INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Interface Gráfica com o Utilizador para Impressoras 3D

JOSÉ VICTOR MARQUES VILELA
novembro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Interface Gráfica com o Utilizador para Impressoras 3D

José Vilela

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Gráficos e Multimédia

Orientador: Dr. Filipe de Faria Pacheco
Supervisor: Eng. Marcos Gomes

Júri:
Presidente:
Dra. Dulce Mota, Professora, DEI/ISEP

Vogais:
Dr. Filipe de Faria Pacheco, Professor, DEI/ISEP
Dr. João Paulo Pereira, Professor, DEI/ISEP

Porto, 5 de Novembro de 2019

Dedicatória

À minha esposa Sónia Cavaco e aos meus pequenos gémeos Samuel e Santiago, pela motivação e boa disposição que me transmitem diariamente, permitindo-me encarar todos os desafios com alegria e confiança.

Resumo

A tecnologia de impressão 3D têm apresentado, nos últimos anos, uma grande evolução. Atualmente a impressão 3D é utilizada para imprimir praticamente tudo, desde braços robóticos inteligentes até habitação acessível para o mundo em vias de desenvolvimento. Neste momento, praticamente todas as impressoras disponíveis no mercado têm uma interface com o utilizador instalada e pronta a usar. No entanto, estas interfaces continuam a ter baixo nível de usabilidade, dificultando a utilização da impressora 3D.

Com esta dissertação pretendeu-se construir uma Interface Gráfica com bons índices de usabilidade e com funcionalidades diferenciadoras que possibilitasse, nomeadamente, reduzir o desperdício de tempo e matéria prima em caso de falha parcial da impressão. Para além disso, pretendeu-se construir um servidor de impressão para comunicar com a impressora 3D e para, através de uma Interface de programação de aplicações *(Application Programming Interface)* (API) comunicar com a Interface Gráfica ou com o software para Computador Pessoal *(Personal Computer)* (PC).

Para isso, foram analisadas várias aplicações de impressão 3D para perceber se a aplicação para PC seria construida de raiz ou se seria um *pluggin* de uma destas aplicações, depois, foram analisadas bibliotecas e *frameworks* python para perceber quais se adequavam ao desenvolvimento, tanto do servidor de impressão como da Interface Gráfica. Seguiu-se a análise, design e implementação dos vários módulos da solução com recurso às ferramentas escolhidas.

Os módulos construídos permitem, imprimir a partir de um PC, de uma drive *Universal Serial Bus* (USB) ou de um ficheiro recente, acompanhar uma impressão (cancelando-a ou pausando e retomando a mesma), remover um objeto da impressão, mudar o idioma da Interface Gráfica do Utilizador(*Graphical User Interface*) (GUI) e trocar de filamento ou extrusor.

Na fase final são apresentados os questionários de qualidade e o *Quantitative Evaluation Framework* (QEF), framework utilizada para avaliar a concretização do projeto.

Palavras-chave: Impressão 3D, Usabilidade, Interface Gráfica com o Utilizador, Servidor de Impressão 3D, Programação Orientada a Objetos

Abstract

3D printing technology has shown a great evolution in recent years. Today 3D printing is used to print just about everything from smart robotic arms to affordable housing for the developing world. Today, virtually all commercially available printers have a ready-to-use user interface installed. However, these interfaces still have a low level of usability, making it difficult to use the 3D printer.

With this thesis we intended to build a Graphical User Interface with good usability indexes and with differentiating features that could, in particular, reduce the waste of time and raw material in case of partial printing failure. In addition, it was intended to build a print server to communicate with the 3D printer and, through its API to communicate with the GUI or PC software.

For this, we analyzed various 3D printing applications to see if the PC application would be built from scratch or if it would be a pluggin of one of these applications, then python libraries and frameworks to analyze which suited the development of both the print server and the Graphical User Interface. This was followed by the analysis, design and implementation of the various solution modules using the chosen tools.

The built-in solution allows you to print from a PC, USB drive or recent file, track a print (canceling or pausing and resuming), remove an object from printing , change the language of GUI and change filament or extruder.

In the final phase are presented the quality questionnaires and the QEF, framework used to evaluate the project's achievement.

keywords: 3D Printing, Usability, Graphical User Interface, 3D Print Server, Object Oriented Programming

Agradecimentos

A realização deste trabalho não teria sido possível sem a colaboração e apoio de todos os que me acompanharam neste processo.

Gostaria de agradecer em primeiro lugar aos orientadores Doutor Filipe Pacheco Paulo, Doutor João Paulo Pereira e Engenheiro Marcos Gomes pelo apoio, concelhos e críticas construtivas.

A toda a equipa da BEEVERYCREATIVE pelo acolhimento e auxílio durante todas as fases deste processo.

A toda a minha família, em especial, aos meus pais, pela educação e valores que me transmitiram e pela liberdade de escolha dada para seguir o meu percurso profissional, e à minha esposa Sónia Cavaco, pela paciência, apoio e preocupação demonstrada nos últimos anos.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

3MF	*3D Manufacturing Format*.
AHP	Método de Análise Hierárquica*(Analytic Hierarchy Process)*.
AMF	*Additive Manufacturing File Format*.
API	Interface de programação de aplicações *(Application Programming Interface)*.
ASCII	*American Standard Code for Information Interchange*.
BBB	*Beagle Bone Black*.
BEE2B	*BEEVERYCREATIVE To Business*.
BEE2BServer	Servidor de Impressão da *BEEVERYCREATIVE To Business* (BEE2B).
BEECura	pluggin para o Ultimaker Cura da BEEVERYCREATIVE.
BEEStats	Servidor de Telemetria da BEEVERYCREATIEVE.
BPIMB	*Banana Pi M2 Berry*.
CAD	Desenho Assistido por Computador *(Computer Aided Design)*.
CNC	*Computer Numerical Control*.
CRUD	*Create, Read, Update and Delete*.
DAE	*Digital Asset Exchange*.
DIY	*Do It Yourself* .
DRF	*Django REST Framework*.
ESA	*European Space Agency*.
FEI	*(Front End of Innovation)*.
FFF	Fabricação de Filamento Fundido *(Fused Filament Fabrication)*.
FIFO	*First In, First Out*.
FPS	*Frames per Second*.
GRASP	*General Responsibility Assignment Software Patterns*.
GUI	Interface Gráfica do Utilizador(*Graphical User Interface*).

HSV	*(Hue, Saturation, Value)*.
HTTP	Protocolo de Transferência de Hipertexto *(Hypertext Transfer Protocol)*.
MELT	*Manufacturing of Experimental Layer Technology*.
MVC	*Model View Controller*.
NCD	Desenvolvimento de Novo Conceito *(New Concept Development)*.
NPD	Desenvolvimento de Novo Produto*(New Product Development)*.
OBJ	*Wavefront .obj file*.
OPIP2	*Orange Pi PC 2*.
ORM	Mapeamento objeto-relacional (*Object-Relational Mapping*).
PC	Computador Pessoal *(Personal Computer)*.
PEP	*Python Enhancement Proposal*.
PET	Polietileno Tereftalato.
PLA	Poliácido Láctico.
PME	Pequenas e Médias Empresas.
QEF	*Quantitative Evaluation Framework*.
REST	Transferência Representacional de Estado *(Representational State Transfer)*.
RGB	*(Red, Green, Blue)*.
RPI3B+	*Raspberry Pi 3 B+*.
SD	*Secure Digital*.
SOLID	*Single responsability, Open-closed, Liskov substitution, Interface segregation and Dependency inversion*.
SSD	*System Sequence Diagrams*.
STL	*STereoLithography*.
TED	*Technology; Entertainment; Design*.
TSG	Processo de Tecnologia por Etapas*(Technology Stage-Gate)*.
UML	*Unified Modeling Language*.
USB	*Universal Serial Bus*.
WFIRM	*Wake Forest Institute for Regenerative Medicine*.

XML *Extensive Markup Language*.

Capítulo 1

Introdução

Este capítulo apresenta sucintamente vários tópicos que são descritos ao longo deste documento.

Começa com um resumo do contexto geral do tema, do problema encontrado, dos objetivos a atingir e dos resultados esperados. Por fim é apresentado um resumo da análise de valor associada à solução prevista e ainda a sua abordagem preconizada, que irá descrever de uma forma resumida os componentes necessários à implementação da solução prevista.

1.1 Contexto

A *BEEVERYCREATIVE* é uma *startup* portuguesa que desenvolve e comercializa tecnologia de impressão 3D. Foi a empresa responsável pelo lançamento da primeira impressora 3D portuguesa, a *BEETHEFIRST*.

Neste momento a empresa conta já com mais 4 impressoras no seu portefólio de produtos, direcionadas para segmentos de mercado diferentes. Um dos segmentos para o qual ainda não há produto que corresponda às necessidades desse mesmo segmento, é o Profissional e Semi-Industrial.

Este projeto faz parte integrante do projeto da *BEEVERYCREATIVE* BEE2B - Fabricação Aditiva de Precisão para a indústria 4.0, o projeto BEE2B tem como objetivo desenvolver um equipamento de fabricação aditiva por Fabricação de Filamento Fundido *(Fused Filament Fabrication)* (FFF) de precisão que vá ao encontro do conceito de Indústria 4.0, com garantia de fiabilidade, durabilidade e, acima de tudo, qualidade das peças impressas, tornando mais eficaz a execução de protótipos e produção de pequenas séries no menor tempo possível.

1.2 Problema

Apesar da grande evolução da tecnologia de impressão 3D nos últimos anos, a grande maioria dos produtos existentes continuam a ter baixo nível de usabilidade, dificultando a sua utilização. Além disso as interfaces gráficas com o utilizador existentes são ineficientes promovendo, embora indiretamente, baixas rentabilidades e desperdício de matéria prima.

1.3 Objetivo

Com o presente trabalho pretende-se desenvolver uma nova interface gráfica para impressoras 3D, para o mercado profissional, que esteja a par das capacidades da indústria 4.0. Pretende-se

que a interface gráfica com o utilizador a implementar tenha bons índices de usabilidade, mas que implemente funcionalidades diferenciadoras, nomeadamente através do desenvolvimento de software para reduzir o desperdício de tempo/matéria prima em caso de falha parcial da impressão. Um exemplo de um projeto com o mesmo objetivo é o *OctoPrint-Cancelobject* (Paukstelis 2018) ,um *plugin* para o servidor de impressão *OctoPrint* que permite cancelar individualmente objetos a meio de uma impressão. Apesar do seu potencial este *plugin* é difícil de utilizar e requer conhecimentos avançados de software de impressão 3D, algo que o utilizador comum não tem. Pretende-se também que o trabalho desenvolvido sirva de base a futuros desenvolvimentos, que permitam a inclusão de algoritmos de *machine learning* para previsão de falhas em tempo real.

1.4 Resultados esperados

Com a realização deste projeto é esperado aumentar os níveis de usabilidade da impressora 3D BEE2B, ou seja, é esperado um aumento do tempo da impressora efetivamente a imprimir *(uptime)* e uma diminuição do tempo despendido a efetuar tarefas de manutenção *(downtime)*, como substituir filamento ou extrusor.

Também é expectável que a solução implementada promova uma redução de tempo e de matéria prima utilizada em caso de falha parcial da impressão.

1.5 Abordagem preconizada

Construir um servidor para gestão de impressão composto por uma API e uma camada de comunicação com o controlador de impressão aplicando as boas práticas de desenvolvimento de software. Em seguida implementar uma aplicação moderna, atrativa e convidativa, que disponibilize uma interface gráfica com o utilizador fluida e inovadora que permita ao utilizador otimizar o *downtime* e *uptime* da impressora 3D. A avaliação da usabilidade desta interface gráfica com o utilizador será obtida com medições dos tempos de conclusão de determinadas tarefas e com questionários aos utilizadores.

Capítulo 2

Contexto e Análise de Valor

Este capítulo tem como objetivo principal apresentar a contextualização do tema deste relatório. Assim na secção 2.1 é apresentada uma descrição detalhada do problema. A secção 2.2 contém uma análise de valor detalhada à solução proposta. As secções 2.3 e 2.4 apresentam um estudo sobre o estado da arte.

2.1 Descrição do Problema

Esta secção explica de forma detalhada o problema existente, do qual surgiu a necessidade da realização deste projeto.

2.1.1 Propósito

2.1.1.1 Propósito do design

O propósito fundamental deste projeto é desenvolver uma nova interface gráfica para impressoras 3D com bons índices de usabilidade, mas que implemente funcionalidades diferenciadoras, nomeadamente através do desenvolvimento de software para reduzir o desperdício de tempo e matéria-prima em caso de falha parcial da impressão, assim como desenvolver o software para Computador Pessoal *(Personal Computer)* (PC) necessário para a preparação de uma impressão, e ainda um servidor de impressão responsável pela comunicação com a impressora 3D.

2.1.1.2 Oportunidades de Mercado

A *BEEVERYCREATIVE* é uma *startup* portuguesa que desenvolve e comercializa tecnologia de impressão 3D. Foi a empresa responsável pelo lançamento da primeira impressora 3D portuguesa, a *BEETHEFIRST*.

Neste momento a empresa conta já com mais 4 impressoras no seu portefólio de produtos, direcionadas para segmentos de mercado diferentes. Um dos segmentos para o qual ainda não há produto que corresponda às necessidades desse mesmo segmento, é o Profissional e Semi- Industrial.

2.1.1.3 Potenciais clientes

A impressora 3D à qual este projeto se destina terá como potenciais clientes pequenas e médias empresas com departamentos de desenvolvimento interno, e que veem na aquisição de uma impressora 3D um meio de encurtar os tempos de projeto.

2.1.1.4 Satisfação dos requisitos do cliente

Através da utilização deste produto os utilizadores da impressora 3D poderão facilmente gerir a sua impressão através de uma interface gráfica atrativa e intuitiva e ainda realizar tarefas de manutenção rapidamente na própria máquina.

Desta forma, a realização de tarefas como, imprimir um ficheiro recente ou substituir um filamento, são muito mais rápidas e intuitivas, já que com as atuais interfaces gráficas estas operações são lentas e confusas quando efetuadas na própria máquina.

Assim, a interface gráfica proporciona ao utilizador, uma otimização do tempo da impressora 3D despendido em operações de manutenção (*downtime*) e efetivamente a imprimir (*uptime*).

2.1.1.5 Definição de valor para o cliente

A possibilidade de gerir facilmente a impressora 3D e as suas ferramentas, através de uma interface gráfica rápida e intuitiva, diretamente na própria máquina, será o grande valor deste projeto para os seus clientes. A possibilidade de, em caso de falha parcial da impressão, remover um objeto da impressão e com isso poupar tempo e material, também traz uma grande mais valia aos seus clientes.

2.1.1.6 Necessidade de novas tecnologias

Não será necessário a criação de uma nova tecnologia para a concretização deste projeto, é preferível, em vez disso, a utilização de tecnologias já bem desenvolvidas e com anos de utilização no mercado.

2.1.1.7 Adaptação de projetos existentes

No caso da interface gráfica, a adaptação de uma solução já existente no mercado não é possível, pois não foi encontrada nenhuma solução que se ajuste às características pretendidas para a interface gráfica. No entanto, para outros componentes do projeto, como o software de impressão 3D para PC ou o servidor de impressão, é possível a adaptação de soluções existentes no mercado.

No caso do servidor de impressão, decidiu-se não adaptar nenhuma solução existente porque as soluções existentes não cumprem todos os requisitos do projeto no que diz respeito a este componente e o nível de dificuldade em adaptar a solução existente com as funcionalidades necessárias seria muito maior do que construir uma nova solução de raiz.

Já quanto ao software para PC a decisão foi adaptar uma solução já existente, o Ultimaker Cura, este software *Open Source* é multiplataforma e cumpre grande parte dos requisitos para este componente da solução. Adicionar as funcionalidades necessárias será feito com a personalização deste software e com a adição de *plugins* e perfis de impressão.

2.1.1.8 Importância do *time-to-market*

O *time-to-market* deste projeto é de elevada importância devido ao facto deste produto ser inovador no que diz respeito à interface gráfica da impressora 3D. Sendo a *BEEVERYCREATIVE* uma organização que fornece soluções de impressão 3D, este projeto é de máximo interesse, de forma a se antecipar à concorrência.

2.1.2 Ponto de vista pessoal do problema

A solução desenvolvida deverá ser constituída por três módulos fundamentais, os quais se apresentam na lista seguinte:

- Interface gráfica rápida e intuitiva, para que seja possível otimizar os tempos da impressora 3D em *uptime* e *downtime*.
- Software de impressão 3D para PC, para que o utilizador possa preparar as suas impressões e posteriormente imprimi-las, seja pela rede ou diretamente na impressora com uma drive USB.
- Servidor de impressão com a capacidade de comunicar com a impressora 3D e de gerir uma fila de impressão. Este servidor deverá comunicar de uma forma centralizada tanto com a interface gráfica como com o software de impressão 3D para PC.

2.1.2.1 Ponto de vista pessoal das partes interessadas

As partes interessadas no desenvolvimento deste projeto são:

- A *BEEVERYCREATIVE*
- Os clientes da *BEEVERYCREATIVE*

A *BEEVERYCREATIVE* passa a ter mais uma impressora no seu portefólio de produtos, agora para um segmento Profissional e Semi-Industrial, onde ainda não disponibilizava nenhum produto.

Os clientes da *BEEVERYCREATIVE* podem usufruir de uma interface gráfica rápida e intuitiva na sua impressora 3D que lhe confere uma melhor usabilidade.

2.1.2.2 Ponto de vista pessoal dos vendedores/fornecedores

Os vendedores/fornecedores têm interesse no desenvolvimento deste projeto uma vez que é necessária a aquisição do hardware utilizado para a construção da impressora 3D, o que lhes garante uma rentabilidade constante.

2.1.2.3 Ponto de vista pessoal da manutenção

O software da interface gráfica a desenvolver irá servir como auxiliar nas operações de manutenção, nomeadamente na substituição de filamento e extrusores, disponibilizando assistentes que acompanham e auxiliam o utilizador durante todo o processo, permitindo desta forma efetuar essas operações de forma mais rápida e clara o que, consequentemente, leva a um menor *downtime* da impressora 3D.

2.1.3 Pressupostos

2.1.3.1 Pressupostos de condições de operação

A impressora 3D será projetada para ser partilhada entre vários utilizadores dentro da mesma empresa mas, apesar de todos terem acesso às suas funcionalidades de manutenção, assume-se que estas operações são efetuadas por um utilizador técnico responsável, pois acarretam alguns riscos para o utilizador, como queimaduras, devido às elevadas temperaturas a que as ferramentas da impressora 3D se encontram.

2.1.3.2 Nível de maturidade das tecnologias emergentes

Existe um bom nível de maturidade, pois o projeto é constituído por tecnologias bem desenvolvidas e com anos de utilização no mercado, mas também por tecnologias ainda em processo de amadurecimento como é o caso da impressão 3D.

2.1.3.3 Pressupostos assumidos para uma solução ótima

A solução criada deverá ser desenvolvida adotando boas práticas de engenharia de software, como modelação orientada a objetos e adoção de padrões de design, e com bons índices de usabilidade. Deverá ainda disponibilizar funcionalidades inovadoras que permitam melhorar o aproveitamento do tempo e material da impressora no caso de falha parcial de impressão.

Desta forma os critérios que caracterizam a melhor solução são uma interface rápida e intuitiva que otimiza o *downtime* e *uptime* da impressora 3D, e funcionalidades que, em caso de falha parcial da impressão, reduzem o tempo e/ou material utilizado na impressão.

2.1.3.4 Pressupostos na disponibilidade dos materiais

Assume-se a existência ou aquisição de hardware, onde os sistemas a serem desenvolvidos possam ser testados durante o período de testes.

2.1.3.5 Pressupostos das competências técnicas

Poderá haver uma formação dos utilizadores da impressora 3D para que sejam capazes de realizar a configuração inicial e outras operações de manutenção.

Como a interface gráfica é bastante intuitiva e disponibiliza assistentes para as tarefas de manutenção, na maioria dos casos, esta formação não será necessária.

2.1.4 Implicações

2.1.4.1 Fontes de informação

A realização deste projeto irá ter como suporte diversos tipos de material.

A documentação das diversas ferramentas, *frameworks*, linguagens e bibliotecas utilizadas permitem obter informação no que toca à utilização da mesma.

A *BEEVERYCREATIVE* possui elevada experiência no desenvolvimento de tecnologia de impressão 3D, exemplos disso, são os inúmeros prémios internacionais que a empresa portuguesa já ganhou, como o *Best Consumer Printer* 2014 pela 3D *Printshow* (Fabb 2014) ou o recente *TCT Aerospace Application Award* 2018 pela *TCT Awards* (Awards 2018).

2.1.4.2 Informação em falta

Existe informação que é preciso analisar tal como:

- Alteração a fazer ao G-code de forma a permitir a identificação de cada objeto individualmente;

2.1.4.3 Obtenção da informação em falta

É de conhecimento público o formato do ficheiro G-code, desta forma pode-se analisar a especificação do tipo de ficheiro para perceber de que forma ele pode ser alterado ou manipulado para permitir a identificação de cada objeto individualmente.

2.1.4.4 Simulação

Numa fase inicial, a interface gráfica será simulada através de um conjunto de hardware, *Raspberry Pi* 3 B e ecrã de sete polegadas, como é mostrado na Figura 2.1. Para este equipamento foi ainda projetada e impressa uma carcaça, como se vê na Figura 2.2.

FIGURA 2.1: Prova de conceito simulada no conjunto de hardware (Raspberry Pi 3 B e ecrã de sete polegadas).

FIGURA 2.2: Estrutura impressa para acomodar o hardware (*Raspberry Pi* e ecrã de sete polegadas) utilizado para simular a prova de conceito.

Para simular a comunicação com a impressora será utilizado um gabarit de testes (Figura 2.3), construído para o efeito, constituído pelos vários motores da impressora, um extrusor e respetivo *nozzle*, e uma placa eletrónica controladora R2C2.

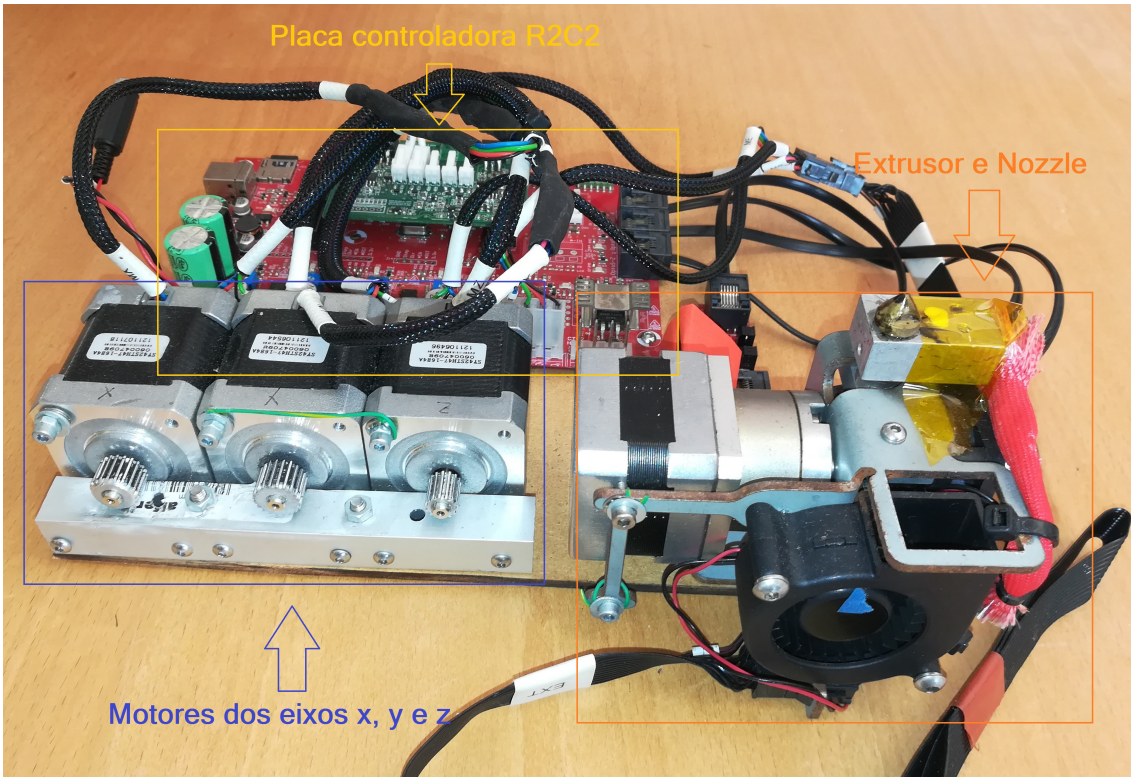


FIGURA 2.3: Gabarit de testes utilizado para simular a comunicação com uma impressora 3D.

2.1.4.5 Testes de componentes

O teste dos componentes do sistema vai ser dividido em vários módulos, nomeadamente:

- Teste de funcionamento do sistema operativo da *Raspberry Pi*;
- Teste de funcionamento da interface gráfica;
- Teste de funcionamento do servidor de impressão;
- Teste de funcionamento do software para PC;
- Teste à consistência do armazenamento de dados;
- Teste à comunicação da interface gráfica e do software de PC com o servidor de impressão;

Todos os componentes desenvolvidos serão alvo de testes unitários.

2.1.4.6 Experiências a realizar

Numa primeira fase, a solução final deve ser testada no *gabarit* e conjunto de hardware definidos no subcapítulo 2.1.4.4 para detetar e corrigir eventuais erros na lógica de funcionamento. Isto é, deve ser efetuada a ligação entre os dois componentes nos quais é simulada uma ligação real de forma a verificar se esta funciona corretamente. Numa segunda fase, a *Raspberry Pi* e o respetivo ecrã devem ser embutidos numa impressora 3D.

2.1.4.7 Estudos, soluções anteriores e problemas

É necessário efetuar um estudo de soluções anteriores de modo a identificar problemas com comunicação da impressora com o software. Outros pontos fulcrais que carecem de atenção e estudo adicional são as falhas de corrente e falhas de rede.

2.1.4.8 Disponibilidade da informação

A informação disponibilizada é suficiente, contudo a documentação de algumas bibliotecas python não estão tão detalhadas como se desejaria.

2.1.4.9 Necessidade de informação

Para poder utilizar os dados necessários para testar o sistema é necessário que o software da impressora 3D tenha um mecanismo de comunicação com o servidor de Telemetria da *BEEVERYCREATIVE*, para que esses dados possam ser transmitidos.

2.1.4.10 Melhor forma para recolher a informação

A melhor forma de recolher os dados é pela utilização da Interface de programação de aplicações *(Application Programming Interface)* (API) do servidor de Telemetria da *BEEVERYCREATIVE*, cada impressora envia os seus dados num intervalo de tempo pré-definido.

2.1.5 Informação necessária para provar a hipótese

2.1.5.1 Conceitos e teorias aplicáveis ao problema

Uma boa usabilidade da interface gráfica é fundamental neste projeto, pois é necessário garantir que o sistema é rápido, simples e intuitivo.

A comunicação sincronizada entre a interface gráfica e a impressora é importante, pois é necessário garantir que a interface gráfica apresenta sempre informação sobre o estado da impressora de forma correta e atualizada.

2.1.5.2 Existência de concorrência

Existem alguns modelos no mercado que competirão com a impressora 3D para a qual a solução final a ser desenvolvida se destina, sendo estes apresentados na secção 3.3 pertencente à capítulo do estado da arte.

2.1.5.3 Tecnologias e teorias apropriáveis

Existem diversas tecnologias e teorias implementadas no mercado que são apropriadas para o desenvolvimento deste projeto.

Linguagens de programação, *frameworks* e bibliotecas bem consolidadas no mercado e com provas dadas serão utilizadas para o desenvolvimento do software.

O desenvolvimento orientado a objetos e a utilização de padrões de design são conceitos necessários para uma melhor arquitetura da solução contribuindo para que esta se torne num sistema modular e escalável.

A utilização de boas práticas de usabilidade e acessibilidade são fundamentais para um design intuitivo, de fácil aprendizagem, eficiente de uso e acessível a todos, que possuam incapacidades ou não.

2.1.6 Reflexão crítica

2.1.6.1 Conjunto de soluções viáveis

Uma vez que o hardware da impressora 3D têm algumas limitações em termos de processamento de dados, todas as soluções candidatas têm de passar necessariamente pela utilização de hardware adicional, neste caso, uma placa controladora Linux (por exemplo uma *Raspberry Pi*) e um ecrã tátil de sete polegadas compatível.

2.1.6.2 Rejeição de outras soluções

Nem todas as soluções possíveis são apropriadas devido a diversas restrições. A lista seguinte apresenta algumas das soluções rejeitadas, bem como a razão para a sua rejeição:

- Preparação da cena 3D a imprimir diretamente na impressora, incluindo manipulação dos objetos 3D e configuração de parâmetros de impressão. Devido ao seu fraco processamento de dados, as placas Linux a utilizar neste projeto, não são adequadas à manipulação de objetos 3D de forma fluída, também o ecrã não favorece esta operação, já que é bastante pequeno para este tipo de funcionalidade.

2.1.6.3 Solução praticável e acessível

A solução é praticável e acessível, uma vez que o hardware necessário (*Raspberry Pi* e respetivo ecrã tátil) tem um preço que ronda 80 euros o conjunto. As restantes despesas estão associadas à manutenção de servidores e base de dados utilizados para recolher e manter a informação transmitida pelas diversas impressoras 3D.

2.1.6.4 Implicações da informação recolhida

Os dados recolhidos são provenientes de diversos modelos de impressoras 3D da *BEEVRYCREATIVE* e devem ser uniformizados de modo a poderem ser processados e guardados no servidor de Telemetria da *BEEVERYCREATIVE*.

2.1.6.5 Implicações das tecnologias não atingirem um nível de maturidade aceitável

De uma forma geral, e uma vez que o sistema utiliza, na sua maioria, tecnologias de hardware e software bastante maduras e desenvolvidas, não existem grandes incertezas em relação à maturidade das tecnologias a utilizar no sistema. No entanto, a capacidade de processamento da placa Linux utilizada pode condicionar a apresentação da cena 3D na interface gráfica sendo, portanto necessário um plano de mitigação deste risco.

2.1.6.6 Importância da sustentabilidade no mercado

O sistema desenvolvido tem de ser suficientemente robusto para não necessitar de substituição ou manutenção de hardware a curto e médio prazo, estando por isso preparado para lidar de forma inteligente com este tipo de situações, nomeadamente, situações de falha de eletricidade, falta de espaço em disco, etc. Espera-se que a manutenção do equipamento só venha a ser necessário em caso de falha do próprio hardware.

A nível de software, o sistema deve ser igualmente robusto, não necessitando de qualquer tipo de manutenção. O software deve possibilitar a sua atualização constante permitindo a correção de falhas das versões anteriores e disponibilizando novas funcionalidades.

2.1.6.7 Possíveis melhorias

Uma vez que o sistema é modular e escalável poderá vir a sofrer melhorias futuras, tanto em termos de software como em termos de hardware. Uma das limitações prende-se com a fraca performance das placas Linux para a apresentação e manipulação de objetos 3D. Uma possível melhoria futura consiste na utilização de um componente com melhor desempenho, como por exemplo uma *Intel NUC* (Intel 2018). Contudo esta melhoria acarreta um acréscimo nos custos da impressora 3D.

Ao nível de software também poderá haver melhorias futuras, como por exemplo, a aplicação de algoritmos de *machine learning* que, obtendo os dados da impressão por visão computacional através da câmara da impressora, poderia analisar a impressão atual de um objeto, compará-la com o objeto que era suposto ela imprimir e analisar os desvios, caso estes fossem suficientemente significativos alertava o utilizador e aconselhava-o a remover o objeto em causa da impressão.

2.1.6.8 Considerações do ciclo de vida

O ciclo de vida do produto corresponde ao ciclo de vida da própria impressora 3D, pois não se prevê a alteração do hardware utilizado durante o seu ciclo de vida. Quanto ao mau funcionamento ou avaria da *Raspberry Pi* ou do respetivo ecrã, a solução passa por uma substituição do equipamento avariado pela equipa de apoio ao cliente da *BEEVERYCREATIVE*.

2.1.6.9 Implicações em caso de falha do produto

A falha do produto pode ser dividida em dois aspetos distintos:

- falha de hardware.
- falha de software.

A falha de hardware passa pela substituição da *Raspberry Pi* ou do respetivo ecrã, sendo necessário, no caso da *Raspberry Pi*, arquitetar uma solução que garanta que os dados não são perdidos.

A falha de software pode implicar uma falha do software da interface gráfica, do servidor de impressão ou do software para PC. Em caso de falha da interface gráfica, é necessário identificar o problema e utilizar medidas preventivas que garantam a estabilidade da interface. No caso de falha no servidor de impressão ou no software de para PC é necessário igualmente identificar o problema e utilizar medidas preventivas que garantam a consistência dos dados, no servidor de impressão são ainda necessárias medidas como a utilização de *backups*, de modo a replicar a máquina o mais rápido possível e com isto normalizar o funcionamento do sistema sem perda de dados.

2.1.6.10 Consequências da alteração de características essenciais

A alteração do sistema operativo instalado na *Raspberry Pi* é a característica de design que, quando alterada, afeta todos os outros componentes do sistema (excluindo o software para

PC). As versões das linguagens, bibliotecas e *frameworks* utilizadas, nomeadamente o python, django e PyQt, quando alteradas podem implicar a reformulação de diversas funcionalidades.

2.1.6.11 Funcionalidades insensíveis à mudança

O protocolo e comunicação (Transferência Representacional de Estado *(Representational State Transfer)* (REST) API/Protocolo de Transferência de Hipertexto *(Hypertext Transfer Protocol)* (HTTP)) e a estrutura de dados do servidor de impressão, são independentes de quaisquer alterações no design e arquitetura de outras características do sistema.

2.1.6.12 Potenciais benefícios de subprodutos

Alterando a forma e arquitetura da comunicação com a impressora 3D, o sistema desenvolvido pode facilmente ser aplicado a outras impressoras 3D da *BEEVERYCREATIVE*, tanto para novas versões das impressoras já existentes como para novas impressoras que se venham a desenvolver no futuro.

2.1.6.13 Reações sociais e tratamento de problemas

Uma vez que o sistema proposto não se aplica ao público em geral, mas apenas a pequenas e médias empresas, não é necessário gerir reações e alterações sociais face a este produto.

2.1.7 Conceito básico do problema

Os conceitos mais básicos do sistema proposto englobam conceitos de comunicação, de acessibilidade e usabilidade. A comunicação deve ser efetuada de forma rápida e segura, reduzindo os riscos de segurança para as partes envolvidas, e a acessibilidade e usabilidade deve ser atingida através de um design intuitivo e eficiente, de modo a que os utilizadores percebam a estrutura e navegação da interface por intuição.

Perguntas chave do problema

As questões às quais se pretende responder, isto é, as hipóteses às quais esta tese pretende apresentar soluções através de uma prova de conceito é:

É possível melhorar o design da interface gráfica das impressoras 3D de modo a conseguir obter bons índices de usabilidade e acessibilidade (downtime e uptime) e assim otimizar a utilização da impressora 3D?

É possível, no caso de falha parcial de impressão, através de funcionalidade diferenciadoras, reduzir o desperdício de tempo e/ou material utilizado numa impressão 3D?

2.2 Análise de Valor

O Modelo de Desenvolvimento de Novo Conceito *(New Concept Development)* (NCD) estabelece uma linguagem comum e a terminologia necessária para otimizar o *(Front End of Innovation)* (FEI) (Koen 2004). Este modelo é constituído por três partes fundamentais (ver Figura 2.4):

- O motor (*engine*) é representado pela cultura e liderança da organização e têm como objetivo impulsionar os cinco elementos do FEI.

- Os fatores influenciadores afetam todo o processo de inovação. Estes fatores são os recursos organizacionais, canais de distribuição, clientes, concorrentes, estratégias de negócio, e o desenvolvimento de tecnologias necessárias ao conceito.
- Os cinco elementos chave do modelo:
  - a identificação da oportunidade(*opportunity identification*);
  - a análise da oportunidade (*opportunity analysis*);
  - a geração da ideia e enriquecimento (*idea generation and enrichment*);
  - a seleção da ideia (*idea selection*);
  - a definição do conceito (*concept definition*);

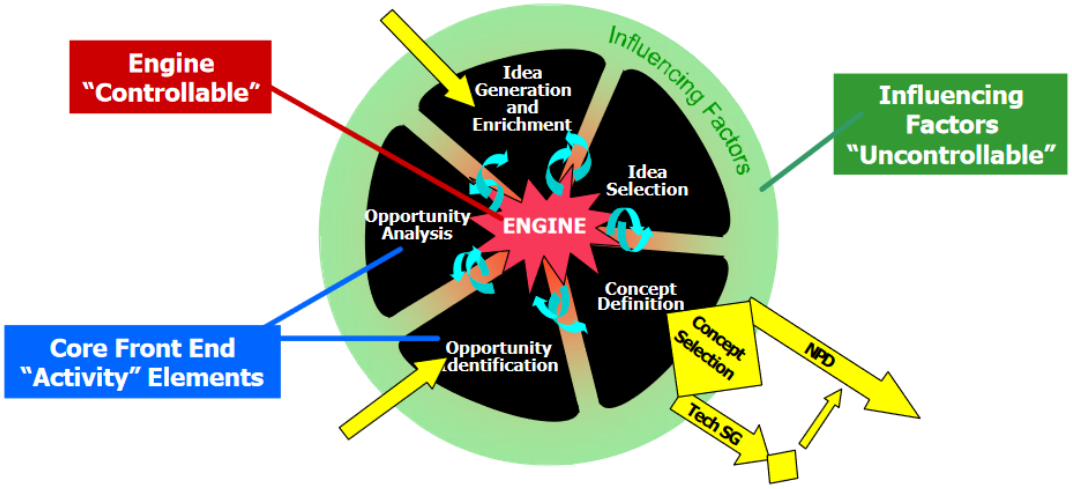


FIGURA 2.4: Modelo de desenvolvimento de novo conceito (Koen 2004).

Como podemos ver na figura 2.4, este modelo contém setas para dentro do modelo que representam o início do projeto e para fora do modelo, que representam a saída do novo conceito para o Desenvolvimento de Novo Produto*(New Product Development)* (NPD) ou para o Processo de Tecnologia por Etapas*(Technology Stage-Gate)* (TSG). A forma circular do modelo tem como objetivo sugerir que o fluxo de ideias seja feito de uma forma desordenada e iterativa entre todos os cinco elementos chave do modelo.

A identificação da oportunidade deste projeto ocorreu quando a *BEEVERYCREATIVE* reconheceu que existiam problemas de usabilidade com as atuais interfaces gráficas das suas impressoras 3D que se traduziam em fracos índices de *downtime* e *uptime* e alguma insatisfação dos clientes. Para a identificação de oportunidades podemos usar ferramentas como:

- *roadmaps*;
- análises e previsões de tendências tecnológicas;
- análises de inteligência competitiva;
- análise de tendências do consumidor;
- estudos de mercado;
- planeamento de cenários;

Para analisar a oportunidade foi efetuado um estudo de mercado onde foi possível concluir que grande parte dos produtos existentes tem fraca usabilidade e não têm funcionalidades que promovam melhores rentabilidades ou redução de desperdício de matéria prima. Para analisar as oportunidades podemos utilizar as mesmas ferramentas utilizadas na identificação da oportunidade, mas refinado e com muito mais detalhe, e ainda outras ferramentas como:

- enquadramentos estratégicos;
- avaliações de segmento de mercado;
- análises da concorrência;
- avaliações dos clientes;

Para a geração da ideia e enriquecimento são usadas várias técnicas e métodos como:

- métodos para identificar as necessidades dos clientes;
- envolvimento precoce dos clientes;
- necessidades de mercado;
- incentivos para estimular novas ideias;

Usando algumas destas técnicas surgiu uma solução, o desenvolvimento de uma nova interface gráfica para impressoras 3D com bons índices de usabilidade e funcionalidades diferenciadoras, nomeadamente através do desenvolvimento de funcionalidades para reduzir o desperdício de tempo/matéria prima em caso de falha parcial da impressão.

Na fase seguinte, a seleção da ideia, é importante decidir que ideias se deve perseguir para tirar o máximo de valor de negócio. Esta seleção pode ser uma simples escolha pessoal, usando por exemplo um portal *web* para reunir as ideias e para dar um *feedback* imediato aos apresentadores das mesmas, ou usando métodos de portefólios baseados em múltiplos fatores como:

- probabilidade de sucesso técnico;
- probabilidade de sucesso comercial;
- recompensas;
- ajuste estratégico;
- alavancagem estratégica;

Para a implementação da ideia selecionada existem diferentes métodos, abordagens e tecnologias, é importante analisar e comparar estes métodos e tecnologias para poder tomar a melhor decisão para a organização.

O último elemento do modelo envolve o desenvolvimento de um caso de negócio baseado em fatores como:

- necessidades do cliente;
- requisitos de investimento;
- avaliação da concorrência;
- potencial do mercado;
- requisitos tecnológicos;

- risco do projeto;

Depois de selecionar os métodos e tecnologias mais adequadas para o projeto foi definido, na *BEEVERYCREATIVE*, através de reuniões com gestores e *stakeholders*, os riscos da solução, as necessidades dos clientes e do mercado, e o esforço e investimento necessário.

Para a definição do conceito podemos ainda recorrer às seguintes técnicas ou métodos:

- abordagens de deliberação de metas;
- definição de critérios que descrevam a atratividade de um projeto;
- avaliação rápida de inovações de alto potencial;
- compreender e determinar o limite de capacidade de desempenho da tecnologia;
- envolvimento precoce do cliente em testes reais ao produto;
- procurar parceiros fora das áreas de competência principais;
- prosseguir abordagens científicas alternativas;

2.2.1 Valor, Valor para o Cliente e Valor Percebido

" A análise de valor pode ser definida como um processo de revisão sistemático aplicado a projetos de produto, com o objetivo de comparar as funcionalidades do produto exigido pelo cliente cumprindo os seus requisitos de uma forma consistente e barata mas com o desempenho e confiabilidade necessária " (Nick Rich 2000).

A definição de valor é imprecisa pois o seu significado depende tanto do cliente como do contexto. As suas características comuns são, o alto nível de desempenho, capacidade, impacto emocional e estilo, relativo ao seu custo. O valor pode ainda ser expresso através da fórmula 2.1, que maximiza a função de um produto em relação ao seu custo (Associates 2016).

$$Valor = \frac{Desempenho + Capacidade}{Custo} = \frac{Função}{Custo} \tag{2.1}$$

O valor do produto é melhorado tanto com a redução de custos como pelo aumento da sua função, desempenho ou capacidade, desde que o custo adicional deste aumento não seja superior (Associates 2016).

Segundo *Tony Woodall*, valor para o cliente é qualquer perceção pessoal do cliente que resulta na sua associação com a oferta de uma organização e pode ter origem numa redução de esforço, na presença de benefícios, ou uma combinação de ambos (Woodall 2003).

" O valor percebido é o valor que um cliente atribui a um determinado produto ou serviço. Normalmente, os clientes desconhecem os fatores envolvidos na atribuição do preço ao produto ou serviço, como os custos reais ou estimados de produção. Assim, os clientes confiam no apelo emocional do produto ou serviço e na avaliação dos benefícios que acreditam que irão receber " (Kenton 2018).

Baseado nos conceitos acima descritos, espera-se que os clientes da nova impressora BEE2B vão beneficiar de uma interface gráfica intuitiva e funcional que lhes permita aumentar os níveis de usabilidade da impressora 3D e ainda em casos pontuais como na falha parcial de uma impressão melhorar o tempo de impressão e reduzir o desperdício de material. Apesar de não ser previsto nenhum custo associado, pois a interface estará embutida na impressora, o

cliente terá como sacrifícios, a configuração inicial da interface e a sua atualização ao longo do tempo caso deseje acompanhar a evolução do software.

Na perspetiva da *BEEVERYCREATIVE* o aumento de funcionalidades, melhoramento da usabilidade da impressora 3D e posicionamento da marca no mercado com uma proposta inovadora, compensam o custo acrescido que será necessário para o produto (custos de desenvolvimento e de hardware).

2.2.2 Proposta de Valor

A interface gráfica *BEE2B* proporciona uma usabilidade intuitiva e inovadora aos utilizadores, permitindo-lhes aumentar a rentabilidade da sua impressora 3D, reduzindo tempos de impressão, desperdício de material e *downtime* da impressora, e aumentando o seu *uptime*.

2.2.3 Modelo Canvas

O modelo *Canvas* é um método para descrever, projetar, desafiar e dinamizar novos modelos de negócio. Este método torna as propostas de valor mais fáceis de discutir e de controlar. O modelo *Canvas* é representado com elementos gráficos que descrevem a proposta de valor do produto ou serviço.

Na Figura 2.5 é possível ver o modelo *Canvas* que foi definido para potencial implementação da solução proposta e que apresento de seguida.

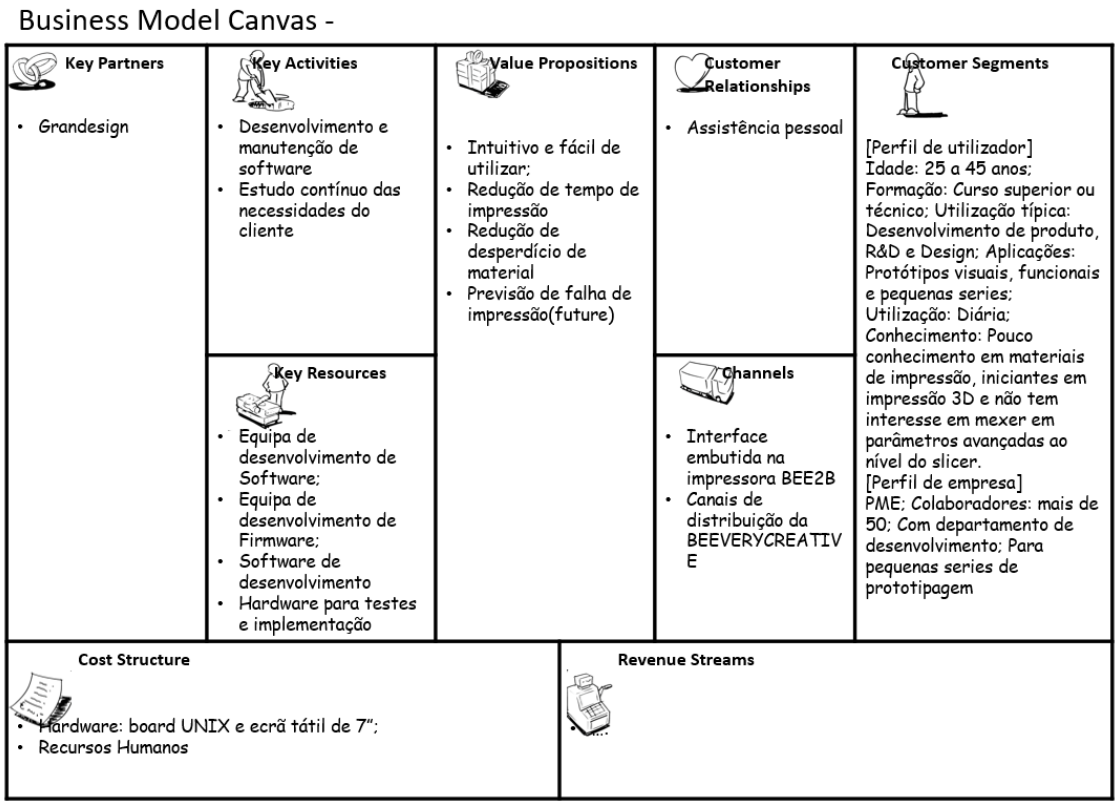


FIGURA 2.5: Modelo *Canvas* da solução proposta.

2.2.3.1 Segmento de Cliente

- **Perfil de Utilizador:**
  - Idade: Entre 25 a 45 anos;
  - Formação: Curso Superior ou técnico;
  - Utilização típica: Desenvolvimento de produto, R&D e Design;
  - Aplicações: Protótipos visuais, funcionais e pequenas series;
  - Utilização: Diária;
  - Conhecimento: Pouco conhecimento em materiais de impressão, iniciantes em impressão 3D e não tem interesse em mexer em parâmetros avançadas ao nível do *slicer*.
- **Perfil de Empresa:**
  - Pequenas e Médias Empresas (PME);
  - Colaboradores: mais de 50;
  - Com departamento de desenvolvimento;
  - Para pequenas series de prototipagem;

2.2.3.2 Proposta de Valor

- Devido aos altos índices de usabilidade a interface gráfica será intuitiva e fácil de utilizar;
- Com recurso às novas funcionalidades possibilita a redução de tempo de impressão e redução de desperdício de material em caso de falha parcial da impressão;
- No futuro, recorrendo a algoritmos de inteligência artificial e visão computacional possibilitar previsão de falhas de impressão em tempo real;

2.2.3.3 Relações com o cliente

- Será feita pessoalmente através da equipa de apoio ao cliente da *BEEVERYCREATIVE*;

2.2.3.4 Canais de distribuição

- A interface gráfica não será vendida em separado pois estará embutida na impressora BEE2B;
- Canais de distribuição da *BEEVERYCREATIVE*;

2.2.3.5 Fontes de receita

- A interface gráfica não terá uma fonte de receita direta devido ao facto de ser vendida embutida na impressora *BEE2B*;

2.2.3.6 Atividades Chave

- Desenvolvimento e manutenção de software necessário à implementação da solução;
- Estudos contínuo das necessidades do cliente;

2.2.3.7 Recursos Chave

- Equipa de desenvolvimento de Software, constituída por um único elemento e responsável pelo desenvolvimento de toda a solução;
- Equipa de desenvolvimento de Firmware, constituída também por um único elemento e que trabalhará muito perto da equipa de desenvolvimento de Software na implementação da camada de comunicação com o controlador de impressão;
- Software de desenvolvimento necessário ao desenvolvimento dos vários componentes da solução;
- Hardware necessário para testes: uma placa controladora Linux e ecrã tátil de 7";

2.2.3.8 Parceiros Chave

- *Grandesign*, empresa especializada em *design* que, através de, *brainstormings*, *design thinkings* e outros métodos e ferramentas irão ajudar a desenhar o *layout* da interface. Este processo será iterativo e nas suas sessões incluirá elementos da Equipa da *Grandesign* (especialistas em *design* e utilizadores de impressoras 3D) e elementos da *BEEVERYCREATIVE* (Elementos das equipas de desenvolvimento de Software, Firmware, Eletrónica e Mecânica).

2.2.3.9 Estrutura de Custos

- Recursos Humanos, equipa de desenvolvimento de Software;
- Hardware necessário: uma placa controladora Linux e um ecrã tátil de 7" por impressora mais algumas unidades adicionais para testes;

2.2.4 Análise de Valor segundo o modelo de Verna Allee

A análise de valor segundo o modelo de *Verna Allee* ou análise de rede de valor " é uma metodologia de modelagem de negócios que visualiza atividades de negócios e conjuntos de relacionamentos a partir de uma perspetiva dinâmica de todo o sistema. Esta metodologia inclui várias abordagens de análise exclusivas e também se integra a outras ferramentas de processo, ferramentas de análise de redes sociais e dinâmicas de sistema " (Allee 2006).

Para analisar a Interface gráfica BEE2B segundo este modelo precisamos de desenvolver um mapa de rede de valor (Figura 2.6), com todas as interações relativas ao produto, este mapa é essencial pois irá facilitar o processo de análise de valor tanto a níveis operacionais como estratégicos. Depois deste mapeamento, definir a sequência dos entregáveis e as *swim lanes*, de seguida vamos sobrepor o mapa de rede de valor nas *swim lanes*. Por fim revemos todo o processo com funções e entregáveis até ficarmos com o mapa de rede de valor detalhado.

2.2.5 Método de Análise Hierárquica

Para a seleção da placa controladora Linux a utilizar foi utilizado o Método de Análise Hierárquica*(Analytic Hierarchy Process)* (AHP).

Os critérios que se considerou para esta análise foram:

- Garantia de disponibilidade do fornecedor;
- Preço;

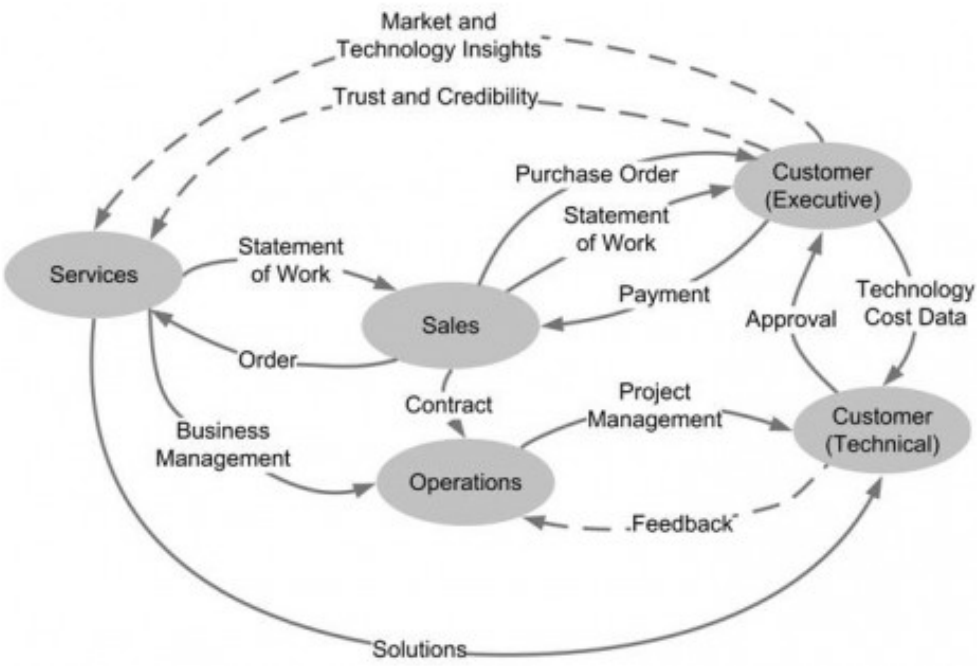


FIGURA 2.6: Mapa de rede de valor (Jarche 2011)

- Performance;

As alternativas disponíveis no mercado mais adequadas foram:

- *Raspberry Pi 3 B+* (RPI3B+);
- *Orange Pi PC 2* (OPIP2);
- *Banana Pi M2 Berry* (BPIMB);
- *Beagle Bone Black* (BBB);

2.2.5.1 Fase 1: Construção da árvore hierárquica da decisão

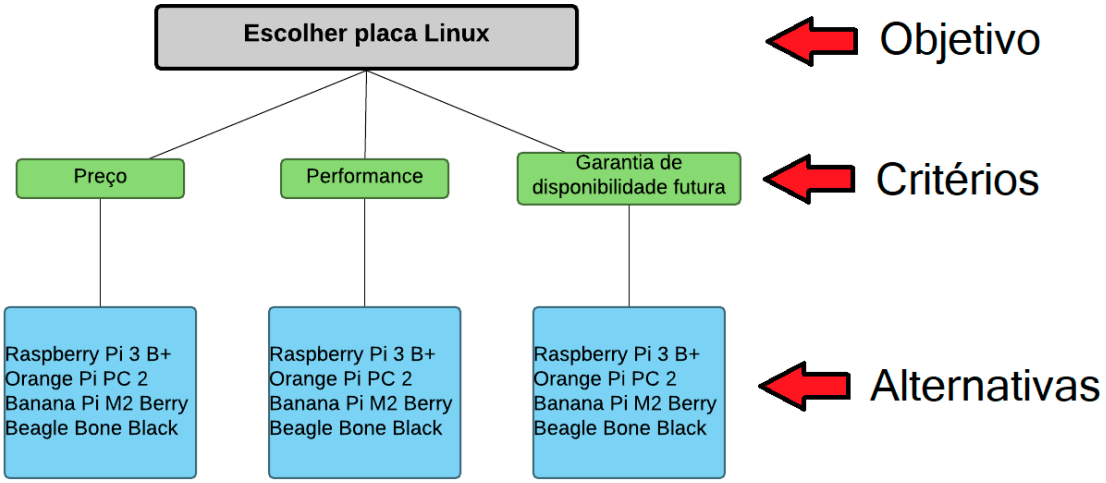


FIGURA 2.7: Árvore AHP - Escolha da placa controladora linux

2.2.5.2 Fase 2: Comparação entre elementos da hierarquia

TABELA 2.1: Matriz de Comparação de critérios.

	Preço	**Preformance**	**Disponibilidade**
Preço	1	$\frac{1}{2}$	$\frac{1}{3}$
Preformance	2	1	$\frac{1}{2}$
Disponibilidade	3	2	1

2.2.5.3 Fase 3: Prioridade relativa de cada critério

TABELA 2.2: Matriz de Comparação de critérios com soma de colunas.

	Preço	**Preformance**	**Disponibilidade**
Preço	1	$\frac{1}{2}$	$\frac{1}{3}$
Preformance	2	1	$\frac{1}{2}$
Disponibilidade	3	2	1
Soma	6	$\frac{31}{2}$	$\frac{15}{6}$

TABELA 2.3: Matriz Normalizada de critérios.

	Preço	Preformance	Disponibilidade	Prioridade Relativa
Preço	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{5}$	0.16
Preformance	$\frac{2}{3}$	$\frac{2}{7}$	$\frac{2}{7}$	0.30
Disponibilidade	$\frac{1}{2}$	$\frac{4}{7}$	$\frac{5}{9}$	0.54

Peso dos critérios:

- Preço 0.16
- Performance 0.30
- Disponibilidade 0.54

2.2.5.4 Fase 4: Avaliar a consistência das prioridades relativas

$$A \cdot X \cong \delta_{max} \cdot X \tag{2.2}$$

$$\begin{bmatrix} 1 & 0.5 & 0.33 \\ 2 & 1 & 0.5 \\ 3 & 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.16 \\ 0.56 \\ 0.12 \end{bmatrix} \cong \delta_{max} \cdot \begin{bmatrix} 0.16 \\ 0.56 \\ 0.12 \end{bmatrix}$$

$$\begin{bmatrix} 0.49 \\ 0.89 \\ 1.62 \end{bmatrix} \cong \delta_{max} \cdot \begin{bmatrix} 0.16 \\ 0.56 \\ 0.12 \end{bmatrix}$$

$$\delta_{max} = \frac{\frac{0.49}{0.16} + \frac{0.89}{0.56} + \frac{1.62}{0.12}}{3} = 3.01$$

$$IC = \frac{\delta_{max}^{-n}}{n-1} \tag{2.3}$$

$$IC = \frac{3.01 - 3}{3 - 1} = \frac{0.01}{2} = 0.005$$

De acordo com a tabela das prioridades relativas:

TABELA 2.4: Tabela das prioridades relativas.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

2.2.5.5 Fase 5: Construção da matriz de comparação paritária para cada critério, considerando cada uma das alternativas selecionadas

Tabela 2.5: Matriz Normalizada do critério Preço.

	RPI3B+	OPIP2	BPIMB	BBB	Prioridade Relativa
RPI3B+	$\frac{1}{2}$	$\frac{5}{9}$	$\frac{1}{2}$	$\frac{3}{8}$	0.48
OPIP2	$\frac{1}{4}$	$\frac{2}{9}$	$\frac{1}{3}$	$\frac{1}{3}$	0.29
BPIMB	$\frac{1}{6}$	$\frac{1}{7}$	$\frac{1}{6}$	$\frac{1}{4}$	0.18
BBB	$\frac{1}{14}$	$\frac{1}{64}$	$\frac{1}{61}$	$\frac{1}{19}$	0.05

Tabela 2.6: Matriz Normalizada do critério Performance.

	RPI3B+	OPIP2	BPIMB	BBB	Prioridade Relativa
RPI3B+	$\frac{1}{7}$	$\frac{1}{3}$	$\frac{2}{7}$	$\frac{1}{8}$	0.23
OPIP2	$\frac{5}{98}$	$\frac{4}{33}$	$\frac{4}{17}$	$\frac{35}{223}$	0.14
BPIMB	$\frac{3}{98}$	$\frac{1}{33}$	$\frac{1}{17}$	$\frac{20}{223}$	0.05
BBB	$\frac{75}{98}$	$\frac{16}{33}$	$\frac{7}{17}$	$\frac{140}{223}$	0.57

Tabela 2.7: Matriz Normalizada do critério Disponibilidade.

	RPI3B+	OPIP2	BPIMB	BBB	Prioridade Relativa
RPI3B+	$\frac{280}{411}$	$\frac{4}{7}$	$\frac{2}{3}$	$\frac{30}{41}$	0.66
OPIP2	$\frac{35}{411}$	$\frac{1}{14}$	$\frac{1}{21}$	$\frac{2}{41}$	0.06
BPIMB	$\frac{40}{411}$	$\frac{1}{7}$	$\frac{2}{21}$	$\frac{3}{41}$	0.10
BBB	$\frac{56}{411}$	$\frac{3}{14}$	$\frac{4}{21}$	$\frac{6}{41}$	0.17

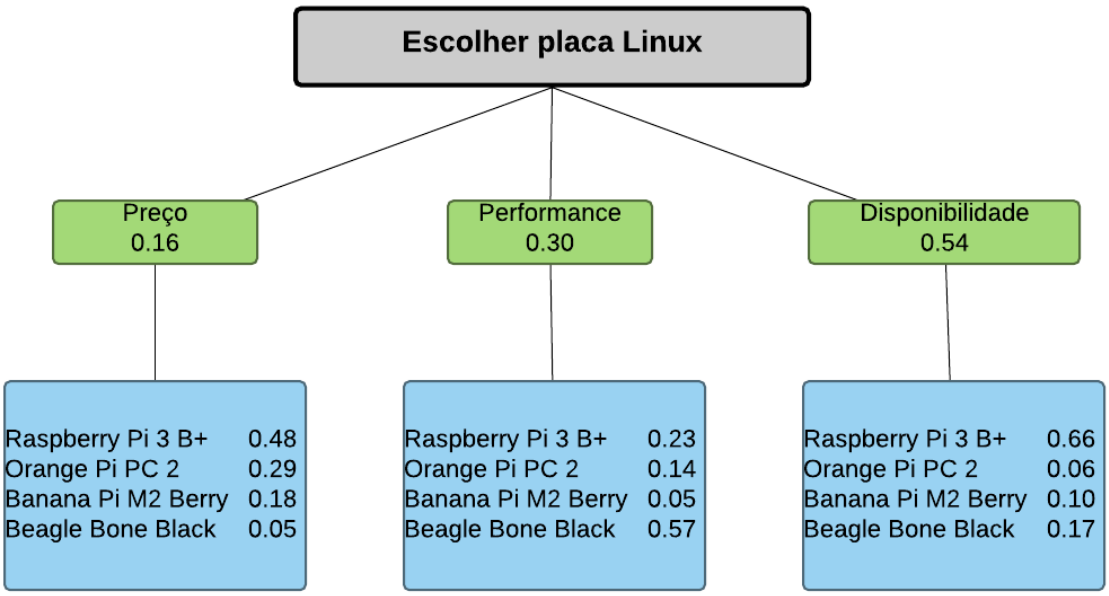


FIGURA 2.8: Árvore AHP com matrizes de comparação paritária de cada critério, considerando cada uma das alternativas selecionadas.

2.2.5.6 Fase 6: Obter a prioridade composta para as alternativas

$$\begin{bmatrix} 0.48 & 0.23 & 0.66 \\ 0.29 & 0.14 & 0.06 \\ 0.18 & 0.05 & 0.10 \\ 0.05 & 0.57 & 0.17 \end{bmatrix} \cdot \begin{bmatrix} 0.16 \\ 0.30 \\ 0.54 \end{bmatrix} = \begin{bmatrix} 0.50 \\ 0.12 \\ 0.10 \\ 0.27 \end{bmatrix}$$

2.2.5.7 Fase 7: Escolha da alternativa

A RPI3B+ aparece como a mais indicada para adquirir, em função dos critérios referidos e das suas respetivas importâncias.

Capítulo 3

Estado da Arte

Neste capítulo é apresentado o estado da arte. Na secção 3.1 é apresentado uma breve história da impressão 3D. Na secção 3.2 é mostrado a evolução das Interface Gráfica do Utilizador(*Graphical User Interface*) (GUI)s das impressoras 3D. Depois, na secção 3.3 vamos ver alguns softwares de impressão 3D. No final do capítulo, na secção 3.4, é apresentado alguns formatos de ficheiros 3D.

3.1 Impressão 3D

A impressão 3D, também conhecida por prototipagem rápida ou manufatura aditiva, é o processo de construir um objeto físico a partir de um modelo tridimensional digital, normalmente depositando sucessivas camadas finas de material.

O início da impressão 3D remonta a 1976 quando a impressora de jato de tinta é inventada. Em 1984, Charles Hull, cofundador da *3D Systems*, inventa a estereolitografia, um processo de impressão que permite que um objeto tangível seja criado a partir de dados digitais. A tecnologia é usada para criar um modelo 3D a partir de uma imagem e permite que os utilizadores testem um projeto antes de investir num programa de manufatura maior(Daly 2013).

Em 1992, O primeiro aparelho estereolitográfico do mundo, é produzido pela 3D Systems, esta máquina possibilitou a fabricação de peças complexas, camada por camada, em muito menos tempo(Charles W. Hull 1986).

A tecnologia estava na sua infância e não era perfeita, tinha vários problemas como, deformação do material enquanto ele endurece ou preço proibitivo das máquinas, mas o seu potencial era inegável. Décadas depois, a história da impressão 3D mostrou que muito do seu potencial ainda está por descobrir.

Em 1999, a impressão 3D começa a dar sinais de maturidade quando, no *Wake Forest Institute for Regenerative Medicine* (WFIRM), é implementado em humanos o primeiro órgão impresso em 3D. Os cientistas deste instituto imprimiram estruturas sintéticas de uma bexiga humana e depois revestiram-na com células de pacientes humanos. Este tecido foi então implantado nos pacientes com pouca ou nenhuma chance de que o seu sistema imunológico os rejeitasse, pois eram feitos das suas próprias células(Moon 2014).

Os seguintes anos da história da impressão 3D foram bastante positivos para a medicina, cientistas de diferentes instituições e *startups* conseguiram marcos importantes, A *Organovo* bio imprimiu os primeiros vasos sanguíneos usando apenas células humanas (Organovo 2011), *Anthony Atala*, diretor do WFIRM fabrica um rim em miniatura durante uma *Technology;*

Entertainment; Design (TED) *Talk*, *Luke Massella*, o paciente que tinha recebido a primeira bexiga impressa em 3D também esta presente na apresentação (Atala 2011) , a *Bespoke Innovations* constrói prótese de uma perna com componentes complexos impressos dentro da mesma estrutura (Danaylov 2012).

Foi também nesta década que a impressão 3D iniciou o seu movimento de *Open Source*. Em 2005, o Dr. Adrian Bowler lança o projeto RepRap, uma iniciativa *Open Source* para criar uma impressora 3D que se auto construi-se ou pelo menos fosse capaz de imprimir grande parte dos seus próprios componentes (RepRap 2019b). Este objetivo é conseguido em 2008 pela Darwin, uma impressora que consegue imprimir todos os seus (RepRap 2015a).

Em 2008 é criado a Shapeways, uma *Startup* que fornece uma plataforma onde os utilizadores podem carregar os seus ficheiros 3D, obter *feedback* dos clientes e de outros designers e mandar fabricar os seus objetos a preços acessíveis à Shapeways ou outras entidades registadas (Shapeways 2019). No ano seguinte entra em cena a MakerBot com uma oferta inovadora de kits *Do It Yourself* (DIY) *Open Source* para os fabricantes construírem as suas próprias impressoras (Makerbot 2019).

A Kor Ecologic e a Stratasys criam o Urbee em 2010, o primeiro carro do mundo cujo corpo foi totalmente impresso em 3D (Quick 2010). No ano seguinte, Engenheiros da Universidade de Southampton no Reino unido, imprimem a primeira aeronave não tripulado do mundo (Gordon 2011).

Em 2014, a *Made in Space* apresenta a primeira impressora 3D para trabalhar sem gravidade com o objetivo de ser usada na estação espacial internacional para imprimir ferramentas e equipamentos (Aeronautics e Administration 2014). Também a *European Space Agency* (ESA) fez a sua própria impressora 3D para imprimir no espaço, o Projeto *Manufacturing of Experimental Layer Technology* (MELT), um consórcio entre SONACA Space, BEEVERY-CREATIVE, OHBSystem e Active Space Technologies, tinha como objetivo desenvolver uma impressora 3D, capaz de imprimir, em ambiente de microgravidade com polímetros de alta performance, peças e ferramentas úteis para os astronautas" (BEEVERYCREATIVE 2018).

Nos dias de hoje a impressão 3D é utilizada para imprimir praticamente tudo, desde braços robóticos inteligentes, substitutos ósseos (Chatellier 2017) e até habitação acessível para o mundo em vias de desenvolvimento (Story 2019).

No futuro, as crianças irão construir os seus próprios projetos de arte com impressoras 3D nas suas salas de aula, dentistas poderão solicitar prescrições para conjuntos de próteses personalizadas e os astronautas irão construir uma base lunar totalmente impressa em 3D com matéria lunar (ESA 2019).

3.2 Interfaces Gráficas em Impressoras 3D

Hoje quase todas as impressoras 3D vêm com uma interface instalada e pronta a usar, mas nem sempre foi assim, há muito pouco tempo as opções eram limitadas ou inexistentes, tendo apenas uma porta USB para comunicar com o computador. Com a evolução da indústria foram sendo desenvolvidas novas soluções para a impressão 3D ao ponto de praticamente todas as impressoras disponíveis no mercado terem já algum tipo de alternativa ao uso do computador. Durante os últimos anos foram utilizados diferentes ecrãs nas impressoras 3D podendo já distinguir duas gerações de ecrãs, a geração dos ecrãs monocromáticos controlados por Firmware e a geração dos ecrãs táteis já controlados por Software.

Na 1ª geração de ecrãs tudo o que pode ser mostrado, desde as opções disponíveis até ao funcionamento da navegação, é determinado pelo Firmware. As opções habituais são: temperatura dos extrusores e da mesa, multiplicador de velocidade de impressão, tempo decorrido desde o início da impressão e por vezes uma barra de progresso ou percentagem de conclusão.

Os ecrãs *Open Source RepRap Discount Smart Controller* e *RepRap Discont full Graphic Smart Controller* foram dos ecrãs mais utilizados e continuam a estar presentes em algumas impressoras atualmente no mercado. Estes ecrãs são muitos simples, apenas possuem um botão e uma roda que controla tudo, o botão é o reset que cancela todos os movimentos da impressora enquanto a roda controla a navegação e o clique seleciona a opção. Estes ecrãs têm ainda um leitor de cartões *Secure Digital* (SD) para poder imprimir diretamente na impressora. Outra opção usual é o *Viki 2*, um ecrã desenvolvido pela *Panucatt* com tutoriais de instalação para várias placas de controlo. Ao contrário dos dois anteriores, este ecrã têm um leitro microSD.

FIGURA 3.1: RepRapDiscount Full Graphic Smart Controller (RepRap 2015b).

A usabilidade dos ecrãs nas impressoras 3D melhoraram bastante com a inclusão de ecrãs táteis, ambientes como salas de aula beneficiaram destes ecrãs pois esta interface é muito mais omnipresente para uma pessoa sem grandes conhecimentos da tecnologia. No entanto, as funcionalidades disponibilizadas continuaram a ser as mesmas melhorando apenas a sua apresentação. Embora os ecrãs não sejam essenciais para a funcionalidade da impressão 3D, estes facilitam muito a execução de várias tarefas e operação de várias impressoras, cada uma com o seu próprio cartão SD em vez de estar dependente do computador.

3.3 Software de impressão 3D

Neste subcapítulo serão apresentados alguns softwares de impressão 3D. Inicialmente será feita uma breve explicação do *Slicer* (3.3.1), software essencial para o processo da impressão 3D. De seguida serão apresentados o Ultimaker Cura (3.3.2), *Simplify3D* (3.3.3), *OctoPrint* (3.3.5) e o *Slic3r* (3.3.4).

3.3.1 Slicer

O *Slicer* é uma ferramenta que converte um modelo 3D digital, geralmente um ficheiro *STereoLithography* (STL) (3.4.2), *3D Manufacturing Format* (3MF) ou *Wavefront .obj file* (OBJ),

FIGURA 3.2: Ecrã tátil de 5' da impressora 3D Ultimaker S5.

em instruções de impressão (ficheiro G-Code (3.4.1)) para que uma impressora 3D possa criar o objeto pretendido. O *Slicer* corta o modelo em camadas horizontais com base nas configurações escolhidas e calcula a quantidade de material e de tempo que a impressora precisará para extrusão. Toda esta informação é agrupada num ficheiro G-Code personalizado para a máquina que se está a usar, para posteriormente ser enviado para a impressora. As configurações do *slicer* afetam muito a qualidade da impressão, por isso é importante ter o software e as configurações corretas para obter a melhor qualidade de impressão possível. Assim, a utilização de um bom *slicer* é de extrema importância, pois os resultados da impressão, nomeadamente o tempo de impressão e a sua qualidade, serão muito influenciados pelo método de *slicing* utilizado.

De seguida são apresentados os aspetos mais importantes para um bom *slicer*:

- **Capacidades de visualização 3D:** Não tendo acesso a nenhum software CAD, provavelmente só vemos o modelo pela primeira vez já no software *slicer*, um bom *slicer* deve possibilitar a capacidade de visualizar o modelo, rodá-lo e fazer *zoom* de forma rápida e sem falhas;
- **Resultados de impressão:** O software deve conseguir bons resultados de impressão;
- **Reparação de modelos 3D:** é importante que o software seja capaz de reparar erros nos modelos 3D de forma automática ou não.
- **Usabilidade:** O software deve ser fácil de usar, intuitivo e com processos simples. Deve ainda ter definições para iniciantes e opções para utilizadores mais avançados. Não deve restringir a acessibilidade para pessoas incapacitadas.
- **Cálculo de estimativas:** o software deve ser capaz de produzir estimativas para a duração de impressão e quantidade de material utilizado.
- **Ajuda:** o utilizador deve ter bastantes opções de ajuda, tanto para utilizadores principiantes como para utilizadores avançados.

Alguns aspetos adicionais importantes para este projeto são:

- **Open Source:** para este projeto é importante que o *slicer* seja *Open Source* pois será necessário personalizá-lo e adicionar funcionalidades, no mínimo deve permitir a criação de *plugins*.

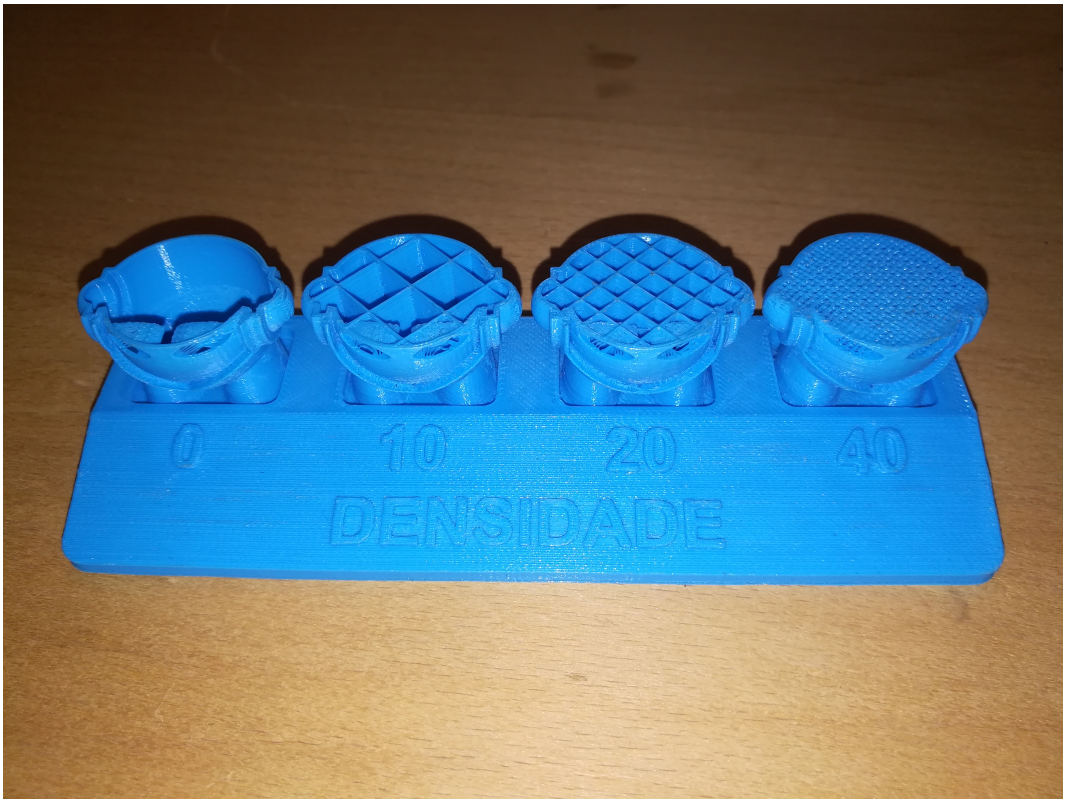


FIGURA 3.3: Objeto impresso para exemplificar a configuração de densidade.

- **Multiplataforma:** é de extrema importância o *slicer* ser multiplataforma, deve estar disponível, pelo menos, para *Windows*, *macOS* e *Linux*.
- **Frequência de Atualização:** o software deve ser atualizado com frequência, para poder usufruir de novas funcionalidades e de novos perfis para as impressoras mais recentes.

3.3.2 Ultimaker Cura

O *Ultimaker Cura* é desenvolvido e mantido pela empresa de impressoras 3D *Ultimaker* e pela sua ativa comunidade de utilizadores. Este software é *Open Source* desde o seu início e até permite a adição de perfis para as impressoras da concorrência.

Ele é capaz de processar e reparar ficheiros STL, 3MF e OBJ. O *Cura* mostra a *toolpath*, tempo de impressão e estimativas de material. Sofre atualizações constantes e permite o desenvolvimento de *plugins*, garantindo assim, que está sempre na vanguarda.

A seguinte lista mostra os pontos fortes e fracos deste software (Locker 2019).

Pontos Fortes:

- É adequado para iniciantes, acesso a apenas as configurações mais importantes;
- É adequado para utilizadores avançados, existem mais de 200 configurações disponíveis;
- A interface gráfica é rápida;
- Permite lidar com impressões de mais de um extrusor;
- Lida com ficheiros grandes de forma rápida;

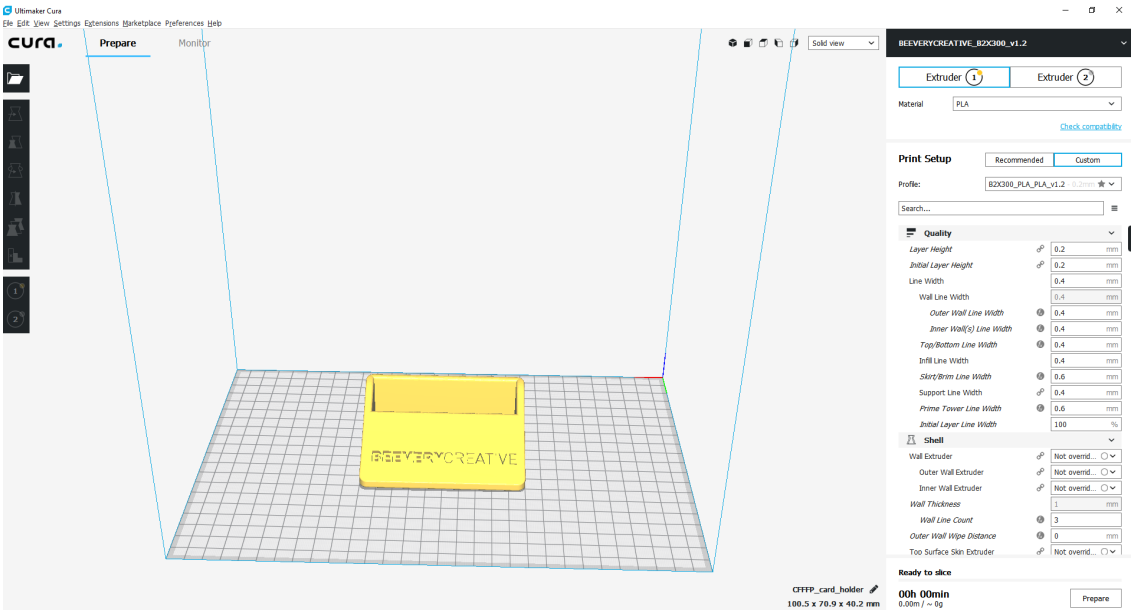


FIGURA 3.4: Interface gráfica do *Ultimaker Cura*.

- Bons resultados de impressão;
- Multiplataforma;
- Permite personalização através de *skins* e *plugins*;
- Atualizações constantes;
- Open Source;

Pontos fracos:

- Algumas funcionalidades não muito importantes ausentes;
- Falhas nas estimativas de impressão que rondam os 10 a 20
- Não muito rápido a processar modelos 3D;

3.3.3 Simplify3D

O Simplify3D é um software *slicer* 3D para profissionais que suporta quase todas as impressoras 3D disponíveis. Para os modelos não suportados, é relativamente fácil adicionar um perfil.

O software permite importar e manipular modelos a partir de ficheiros STL, OBJ ou 3MF. A importação dos ficheiros é muito rápida, mesmo malhas enormes são exibidas em pouco tempo.

Existem imensas configurações disponíveis: extrusores, controlo de camadas, métodos de preenchimento, configurações de temperatura e edição de ficheiros G-Code. Estas configurações podem ser guardadas para facilitar testar várias configurações de *nozzles* ou diferentes filamentos.

De seguida vamos ver uma breve lista de pontos fortes e fracos deste software (Locker 2019).

Pontos Fortes:

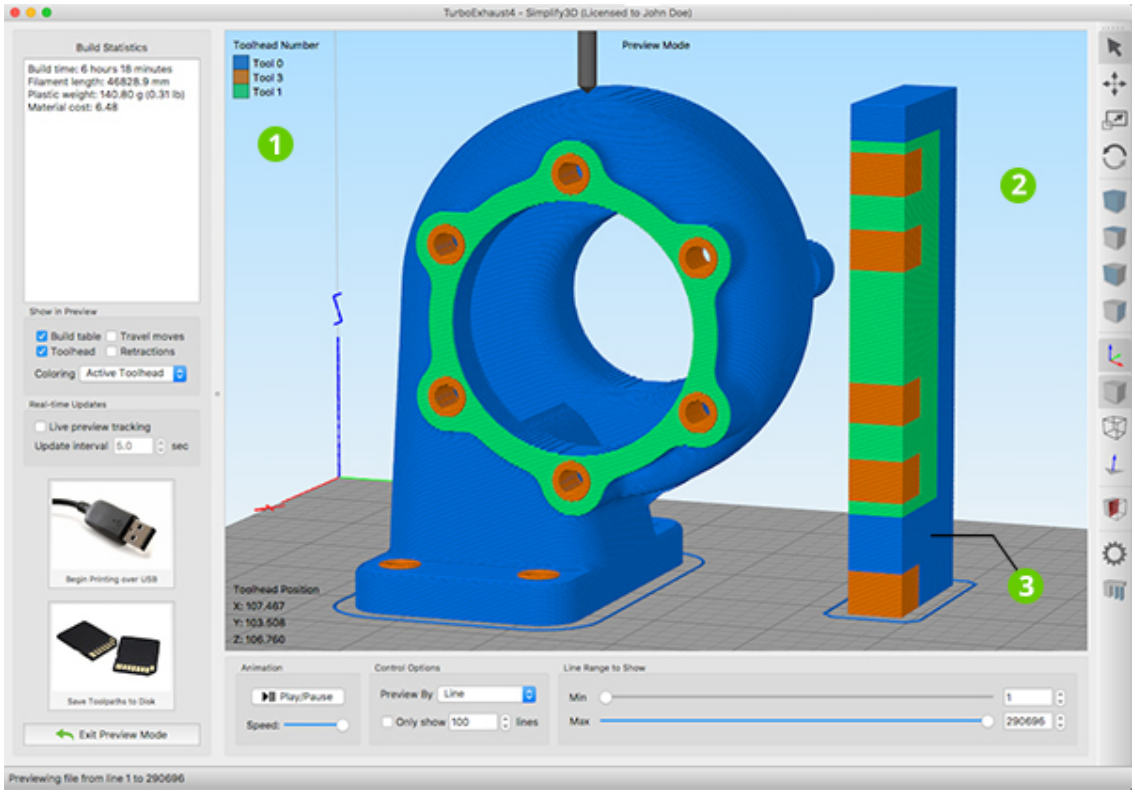


FIGURA 3.5: Interface gráfica de Simplify3D (Simplify3D 2019).

- Excelentes resultados de impressão;
- É adequado para iniciantes, a enorme quantidade de configurações não sobrecarrega o utilizador;
- É adequado para utilizadores avançados devido à enorme quantidade de configurações disponíveis;
- Excelente documentação;
- Suportes editáveis;
- Permite lidar com impressões de mais de um extrusor;
- Multiplataforma;

Pontos fracos:

- Apesar de existir um modo básico, é aconselhado ter alguma experiência com uma impressora 3D;
- Não é *Open Source*;
- Não permite personalização nem adição de *plugins*;
- Atualizações pouco frequentes;
- Preço, 150$ por cada licença de 3 computadores;

3.3.4 Slic3r

O Slic3r é um software de *slicer* 3D *Open Source* com a reputação de adicionar recursos inovadores só ali encontrados. A atual versão do software inclui múltiplas vistas para que o utilizador possa ver melhor como os seus modelos serão impressos. Este *slicer* permite a integração direta com o Octoprint.

O Slic3r acumulou, ao longo dos anos, bastante conhecimento devido à sua vasta comunidade experiente, que foi experimentando configurações, materiais e novas impressoras 3D ao longo dos anos. Aqui foi desenvolvida muita da tecnologia que utilizamos hoje, nomeadamente, múltiplos extrusores, micro camadas e altura de camadas variável.

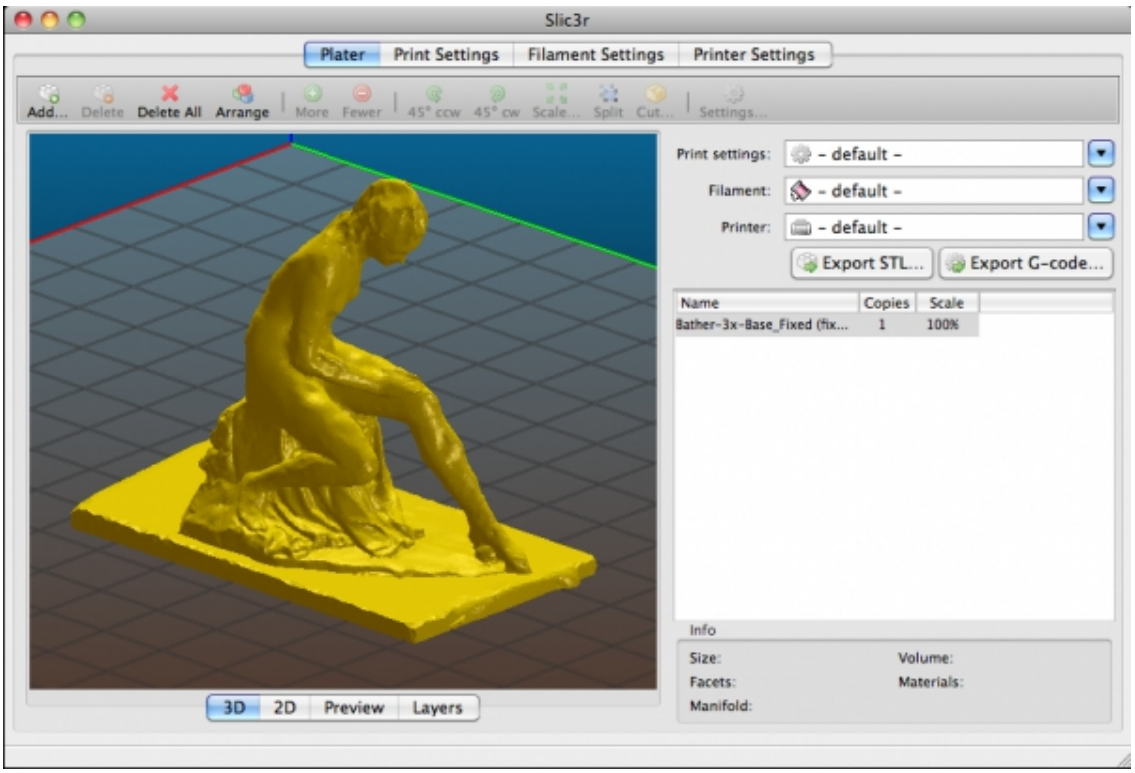


FIGURA 3.6: interface gráfica do Slic3r (Slic3r 2019).

Vamos agora ver uma breve lista de pontos fortes e fracos deste software (Locker 2019).

Pontos Fortes:

- *Open Source*;
- Multiplataforma;
- Razoavelmente rápido devido ao *slicing* em tempo real, mesmo em computadores lentos;
- Alteração de configurações apenas implicam o recalculo das partes afetadas;
- *Slicing* incremental em tempo real;
- Visualização em 3D;
- Visualização de *toolpaths* em 2D e 3D;
- Formato de camada personalizável;

- Integração com OctoPrint;
- Regulação de pressão;

Pontos fracos:

- Não há estimativas de tempo e de material necessários à impressão;
- Não é adequado a iniciantes, pois a sobrecarga de funcionalidades irá sobrecarregálos.
- Não permite a adição de *plugins*;

3.3.5 OctoPrint

O OctoPrint não é um *slicer*, é uma interface web para a impressora 3D, que permite controlar remotamente e monitorar todos os aspetos da impressora 3D e os respetivos trabalhos de impressão, diretamente no *browser*. A aplicação é instalada numa *Raspberry Pi*, pode ter uma câmara instalada, e permite a adição de uma imensa quantidade de *plugins* disponíveis. Assim, os trabalhos de impressão podem ser efetuados diretamente na impressora sem necessidade de um cartão SD. O OctoPrint usa o Cura Engine como *slicer*, o que lhe confere todos os benefícios deste software da Ultimaker.

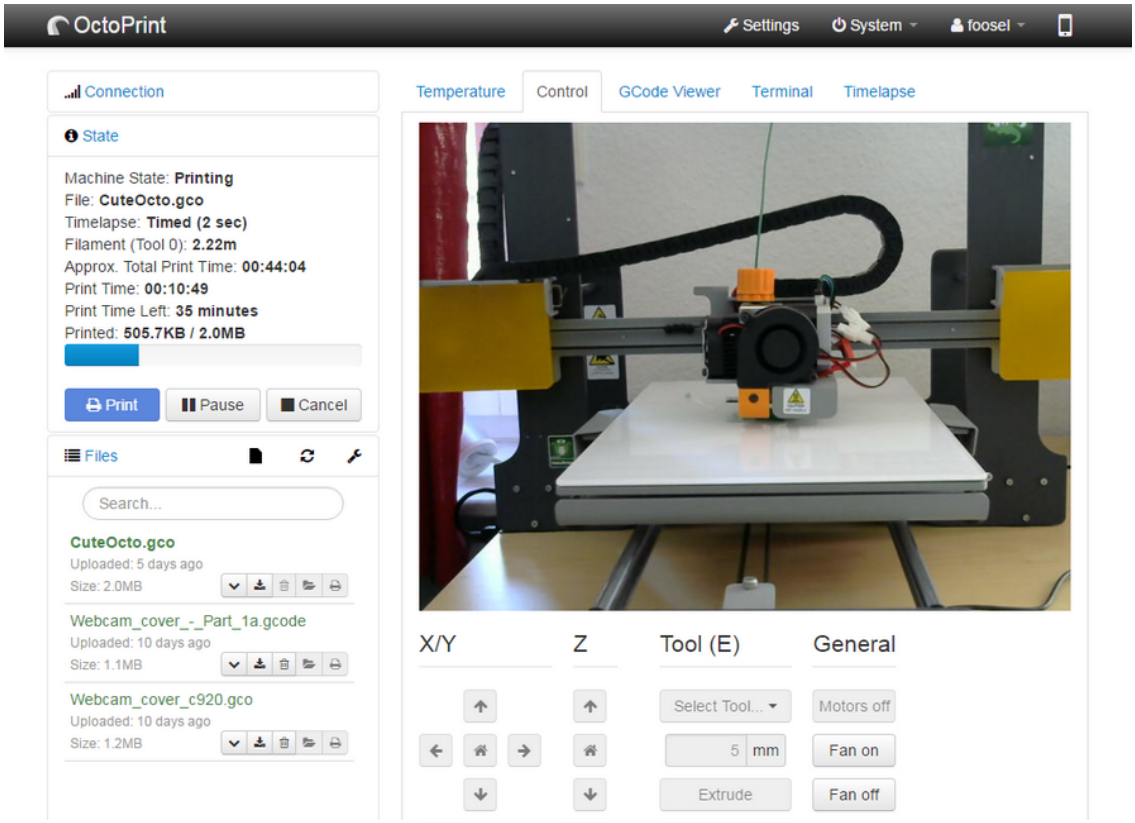


FIGURA 3.7: Interface gráfica de OctoPrint (Octoprint 2019).

A seguinte lista apresenta os seus pontos fortes e fracos (Locker 2019).

Pontos Fortes:

- Open Source;
- Poderoso sistema de plugins;

- Monitorização remota da impressora;
- Visualizador de G-code integrado;
- Aplicação móvel;
- Estatísticas da impressora e dos trabalhos de impressão;
- Multiplataforma;

Pontos fracos:

- Não é adequado a utilizadores iniciantes, expõe demasiadas opções o que sobrecarrega o utilizador;
- Interface gráfica pouco intuitivo;
- Aplicação móvel com demasiados problemas;
- Apesar da qualidade do *slicer* ser boa, demora muito tempo que a realizá-lo, comparativamente aos seus concorrentes, devido à performance da Raspberry Pi.

3.4 Formatos de ficheiros

Para possibilitar a comunicação entre o Software Desenho Assistido por Computador *(Computer Aided Design)* (CAD) que faz o modelo 3D e a impressora são utilizados vários formatos, de seguida fica uma breve descrição dos mais utilizados.

3.4.1 G-Code

GCode é uma linguagem de programação de controle numérico e é utilizada principalmente na imbricação assistida por computador para controlar ferramentas mecanizadas automatizadas (RepRap 2019a). Esta linguagem define as instruções fornecidas ao controlador da máquina que por sua vez informa os motores para onde se devem mover, com que rapidez e qual o caminho a percorrer.

Resumidamente, de 1970 a 2010, os fabricantes de máquinas *Computer Numerical Control* (CNC) enfrentaram dificuldades de compatibilidade porque foram adicionadas extensões e variações de forma independente pelos fabricantes de máquinas e compreender essas disparidades raramente era possível (Buffalo 2008). Em 2010, a sintaxe do GCode era defeituosa e limitada, não possibilitando qualquer tipo de construção, como variáveis com nomes de palavras naturais ou operadores condicionais, mas a sintaxe mais recente traz uma linguagem aprimorada e rica que permite que as máquinas realizem operações como:

- Rápidos movimentos;
- Movimento de avanço controlado em linha reta ou em arco;
- Movimentos de avanço controlados para curvas altamente sensíveis e precisão de fusão de filamentos;
- Sistema de coordenadas de comutação;
- Definir informações da máquina.

O GCode possui um alfabeto em que cada variável é representada por uma letra (de A a Z) com uma descrição da interface que define uma lista de argumentos e um valor de retorno. Por exemplo, alguns códigos representativos são:

- **F** : define a rácio de avanço, ou seja, a velocidade da ferramenta;
- **G** : define um movimento de uma ferramenta;
- **X** : representa a posição absoluta ou incremental no eixo X;
- **Y** : representa a posição absoluta ou incremental no eixo Y;
- **Z** : representa a posição absoluta ou incremental no eixo Z;
- **M** : representam solicitações para obter valores da ferramenta, em tempo real ou estabelecidos anteriormente, como temperatura, coordenadas ou posição inicial.
- G1 F1000 X10Y10Z10 : Executa um movimento de posicionamento rápido com um valor incremental de dez unidades (geralmente polegadas ou centímetros) em cada um dos eixos representados com uma velocidade de 1000 unidades (geralmente polegadas por minuto ou milímetros por minuto).

Vamos considerar, por exemplo, este GCode produzido na aplicação Cura (3.3.2):

```
1  M109 S200
2  M82 ;absolute extrusion mode
3  G28 ;Home
4  G1 Z15.0 F6000 ;Move the platform down 15mm
5  ;Prime the extruder
6  G92 E0
7  G1 F200 E3
8  G92 E0
9  G92 E0
10 G1 F1500 E-6.5
11 ;LAYER_COUNT:1207
12 ;LAYER:0
13 M107
14 G0 F3600 X153.634 Y120.931 Z0.3
15 ;TYPE:SKIRT
16 G1 F1500 E0
17 G1 F1800 X154.144 Y120.328 E0.01486
18 G1 X154.712 Y119.78 E0.0297
19 G1 X155.333 Y119.292 E0.04456
20 G1 X155.4 Y119.249 E0.04606
21 G1 X155.425 Y119.223 E0.04673
```

LISTING 3.1: gcode exemplo

Esse trecho de código possui três séries de comandos singulares: comando de bloco, configuração de parâmetros de comando e um comando de movimento. Em detalhe, o M109 aquece o extrusor da máquina até cerca de 200 graus Celsius, o comando G92 afetam as variáveis de impressão dizendo ao firmware para usar coordenadas absolutas para extrusão e G1 para mover as ferramentas (extrusor e mesa de impressão) para o ponto singular especificado, representado pelas posições absolutas X, Y e Z. Com soluções proprietárias, ao nível do hardware ou mesmo do software, o avanço da tecnologia está a promover departamentos de pesquisa e desenvolvimento para otimizar o código da máquina e criar código especializado.

3.4.2 STL

Este formato define uma superfície geométrica de um modelo 3D sem qualquer representação de cor, texturas ou outro atributo. Esta superfície é separada logicamente numa série de pequenos triângulos (faces). Cada face é descrita por uma direção perpendicular e três pontos que representam os Vértices do triângulo (Burns 1999).

Estes ficheiros podem ser codificados em modo binário ou *American Standard Code for Information Interchange* (ASCII), sendo o primeiro mais comum, seguro e prático, pois é uma representação dos ficheiros mais compacta. O formato STL não é o único formato utilizado, mas é o mais utilizado e confiável quando comparado com outros formatos utilizados pelas empresas de impressão 3D, como o OBJ ou o *Digital Asset Exchange* (DAE).

Independentemente do formato do ficheiro de Input, a estrutura de dados é exportada para a linguagem da máquina, G-code, para ser corretamente interpretada pelo hardware da impressora que executa os comandos sobre as ferramentas da impressora.

3.4.3 AMF

Ao contrário do formato STL, este formato tem suporte nativo para cores, materiais entre outros atributos e é baseado em *Extensive Markup Language* (XML).

Este formato pode representar um objeto ou vários objetos organizados em constelação. Cada objeto é descrito como um conjunto de volumes não sobrepostos. Cada volume é descrito por uma malha triangular que faz referência a um conjunto de pontos (vértices). Estes vértices podem ser partilhados entre volumes pertencentes ao mesmo objeto. Estes ficheiros são compactados usando o formato zip, mas é mantida a extensão do ficheiro ".amf" (*International Organization for Standardization* (ISO) 2016).

Capítulo 4

Avaliar soluções existentes

No âmbito do desenvolvimento deste projeto foi fundamental avaliar as diversas soluções que estão disponíveis no mercado, para adotar a melhor abordagem possível. Este capítulo apresenta toda a informação relacionada com as tecnologias e a abordagem sobre a solução escolhida.

4.1 Soluções existentes

A escolha das tecnologias que serão utilizadas na resolução do problema, foram na sua grande maioria, sugeridas pelo coorientador Marcos Gomes, pela sua experiência no domínio e trabalho anterior na BEEVERYCREATIVE.

A utilização da linguagem *Python* e software para PC *Ultimaker Cura*, ambos amplamente utilizados na comunidade *Open Source* e impressão 3D, foram assim um dado adquirido desde o início do projeto.

Ficaram ainda algumas decisões por tomar como, tecnologia do servidor de impressão e interface gráfico.

Para efetuar estas escolhas foram feitas análises comparativas para cada caso, na secção seguinte vamos detalhar essa análise para a escolha da framework a utilizar no desenvolvimento do servidor de impressão e para a framework a utilizar no desenvolvimento da Interface Gráfica.

4.2 Avaliação de soluções existentes

4.2.1 Servidor de Impressão

Segundo a JetBrains Jetbrains (2019), Django e Flask são de longe as duas *frameworks* web mais populares. Ambas as *frameworks* têm um bom suporte e comunidades maduras, e ambas oferecem abordagens produtivas para o desenvolvimento de aplicações web, permitindo que nos concentremos nas partes únicas da nossa aplicação.

Para o servidor de Impressão foi feita uma análise comparativa entre estas duas *frameworks* web. O Objetivo era escolher a que permitisse um rápido desenvolvimento e que fosse assente em boas práticas de desenvolvimento de software.

O Django e o Flask são ambas grátis, open-source e projetadas para criar aplicações web.

Relativamente à estabilidade, o Django têm ciclos de lançamento mais longos e rígidos que o Flask, assim, as novas versões do Flask vêm com mais recursos novos mas têm uma menor compatibilidade com versões anteriores.

Ao contrário do Flask, o Django adota uma abordagem de "baterias incluídas", ou seja, várias ferramentas, padrões de design, recursos e funcionalidades estão prontas para utilização.

O Flask, por outro lado, é muito mais flexível que o Django. Apesar de o Flask disponibilizar, pronto para uso, roteamento URL, gestão de erros e requests, templates, cookies, suporte para testes unitários e servidor de desenvolvimento, a maioria das aplicações web precisam de um pouco mais, como Mapeamento objeto-relacional (*Object-Relational Mapping*) (ORM) ou autenticação e autorização. Assim, temos flexibilidade a criar a nossa aplicação, podemos decidir usar extensões de terceiros ou escrever o código de raiz.

Também em termos de manutenção, o Flask têm vantagem relativamente ao Django, pois no caso de revisão de código, existe muito menos código para rever.

Base de Dados

O Django inclui um ORM simples, mas poderoso, que suporto diversas base de dados relacionais prontas para uso como, SQLite, PostgreSQL, MySQL e Oracle. Este ORM fornece suporte para criar e gerir migrações de base de dados. Também torna as tarefas de criar formulários, vistas e modelos da base de dados bastante simples, o que é perfeito para o típico *Create, Read, Update and Delete* (CRUD).

O Flask não faz suposições sobre como os dados serão armazenados, mas existem muitas bibliotecas e extensões disponíveis para ajudar nesta tarefa como: SQLAlchemy ou Peewee.

Autenticação e Autorização

O Django disponibiliza esta funcionalidade, juntamente com a gestão de contas de utilizador e suporte para sessões, prontas para uso. O Flask também fornece suporte para sessões baseadas em *cookies*, mas para a gestão de contas, autenticação e autorização é necessário recorrer a bibliotecas que extensões de terceiros.

Administrador

O Django têm um interface de administrador automático, que permite gerir os dados da base de dados baseado nos nossos modelos. Isto permite executar rapidamente operações CRUD sem escrever nenhum código extra. Mais uma vez, o Flask não fornece nada disto, no entanto, oferece as mesmas funcionalidade via extensões e bibliotecas.

Roteamento e Vistas

Ambas as *frameworks* permitem mapear URLs para vistas e oferecem suporte a vistas baseadas em funções e classes.

Formulários

Os formulários são uma parte essencial em muitas aplicações web. O Django têm formulários prontos para uso que incluem funcionalidades como: validações de cliente e servidor, e manipulações de segurança. por defeito, o Flask não suporta formulários, mas como sempre, existem bibliotecas e extensões que podemos adicionar para realizar esta tarefa.

***Templates* e ficheiros estáticos**

Os *templates* permitem injetar dinamicamente informações numa página a partir do *backend*. Flask usa o Jinja2 por defeito, enquanto o Django têm o seu próprio motor de *templates*. Ambos são bastante semelhantes em termos de sintaxe e conjuntos de recursos.

Ambas as *frameworks* também têm suporte para gestão de ficheiros estáticos. O Django traz um comando bastante útil, que agrupa todos os ficheiros estáticos e colóca-os num local central para *deployments* de produção.

Testes

Ambas as *frameworks* têm suporte embutido para testes. Para testes unitários, ambas utilizam a *framework* unittest. Ambos também suportam um cliente de teste para o qual podemos enviar pedidos e, de seguida, inspecionar e validar partes da resposta.

TABELA 4.1: Comparação das caraterísticas das *frameworks* web estudadas.

	Django	**Flask**
Estabilidade	+++	+
Flexibilidade	+	++
Manutenção	+	++
F1: Base de Dados	sim	não
F2: Autenticação e Autorização	sim	não
F3: Administrador	sim	não
F4: Roteamento e vistas	sim	sim
F5: Formulários	sim	não
F6: Templates e ficheiros estáticos	sim	não
F7: Testes	sim	sim

Como podemos verificar na tabela 4.1, apesar do Flask ter maior flexibilidade e melhor manutenção, o Django ganhe nas restantes características, isto deve-se muito, ao facto de trazer "baterias incluídas", e assim, ter diversas funcionalidades prontas para uso desde o início do desenvolvimento de uma aplicação web.

4.2.1.1 REST API

A *Django REST Framework* (DRF) é de longe a biblioteca para implementar REST APIs mais popular (Django 2019). Esta framework traz tudo o que é necessário para implementar uma REST API de forma rápida e fácil: vistas, *serializers*, validações, autenticação, *cache* e controle de versões.

4.2.2 Interface Gráfica

Existem diversas *frameworks* e bibliotecas para a implementação da Interface gráfica, o objetivo é escolher uma *framework* que possibilite o rápido desenvolvimento seja multi-plataforma (obrigatório suportar o sistema operativo da raspberry pi) e open-source.

Segundo a JetBrains Jetbrains (2019), as frameworks TKinter e PyQT são as mais utilizadas, doze e onze por cento respetivamente. Ambas as frameworks são multi-plataforma e open-source, mas o pyQt têm duas vantagens relativamente ao tkinter.

Como é um dado adquirido que o Ultimaker Cura será utilizado para o software de PC, vai ser necessário utilizar PyQT para desenvolver o *plugin* pluggin para o Ultimaker Cura da BEEVERYCREATIVE (BEECura), assim, usando PyQt também na GUI BEE2B podemos nos focar a aprendizagem em apenas numa framework.

A segunda vantagem do PyQt é o seu rápido desenvolvimento, utilizando a ferramenta Qt Designer podemos desenhar os esboços da GUI numa Interface gráfica dedicada e posteriormente converter num ficheiro python com toda o código no "esqueleto"da aplicação. Assim, apenas precisamos construir o modelo de dados do domínio e de ligar os diversos eventos às respetivas ações.

4.3 Conclusão

Para o nosso servidor de impressão vamos valorizar mais aspetos como estabilidade, manutenção, autenticação e autorização, administrador e roteamento e vistas. Relativamente a estas características, o Django mostra-se a melhor solução. Para a implementação da REST API será utilizada a *framework* mais popular para o efeito, a DRF.

Quanto à interface gráfica, foi escolhido PyQt devido ao seu rápido desenvolvimento e pelo facto de partilhar a mesma *framework* que os *plugins* do Ultimaker Cura.

Capítulo 5

Análise e Design da Solução

Este capítulo descreve o design da solução proposta, o modelo de domínio (5.1), a arquitectura da solução proposta (5.2), o seu design com a avaliação de diferentes alternativas (5.3), o modelo de dados, o modelo de componentes e de implementação, e a Navegação da Interface Gráfica (6). Podemos encontrar a identificação dos requisitos de hardware, funcionais e não-funcionais no Apêndice (A).

5.1 Modelo de Domínio

“ O modelo de domínio é uma representação visual das classes conceptuais ou objetos reais num domínio "(Fowler 1996).

“ O termo Modelo de domínio significa a representação de classes conceptuais, não objetos de software, de uma situação real. O termo não significa um conjunto de diagramas que descrevem classes de software, a camada de domínio de uma arquitetura de software ou objetos de software com responsabilidades ”(Larman 2004).

A construção de um modelo de domínio ajuda na diminuição da diferença representacional entre os nossos modelos mentais e de software, pois os nomes dados às classes conceptuais do modelo de domínio vão inspirar os nomes das classes de software.

O modelo de domínio da Figura 5.1 é inspirado nos casos de uso anteriormente descritos. A impressora é gerida pelo Servidor de Impressão e possui uma Lista de Ferramentas, uma Fila de Impressão e uma Lista de Ficheiros.

A Lista de Ferramentas é composta por:

- Extrusores, de um a quatro e contém tipo de material, diâmetro do nozzle e temperatura, cada extrusor pode ter um Filamento carregado, este filamento contém tipo de material, quantidade e cor.
- Mesas, uma ou mais mesas que contém a temperatura (apesar de ser sempre apenas uma mesa esta pode ser dividida em zonas de aquecimento independentes, quando se considera 4 mesas estamos a considerar a mesa dividida em quatro parte iguais).

A Lista de Ficheiros é composta por zero ou mais ficheiros com a cena 3D que contém uma cena 3D e é descrita por uma Descrição do Ficheiro da Cena 3D que contém nome e número de objetos.

A fila de impressão é composta por zero ou mais trabalhos de impressão, cada utilizador pode criar estes trabalhos de impressão, um trabalho de impressão consiste numa cena 3D que é composta por pelo menos um objeto 3D que contém um ID. No final do trabalho de

impressão é gerado um relatório de impressão que contém o tempo de impressão e a quantidade de filamento utilizado.

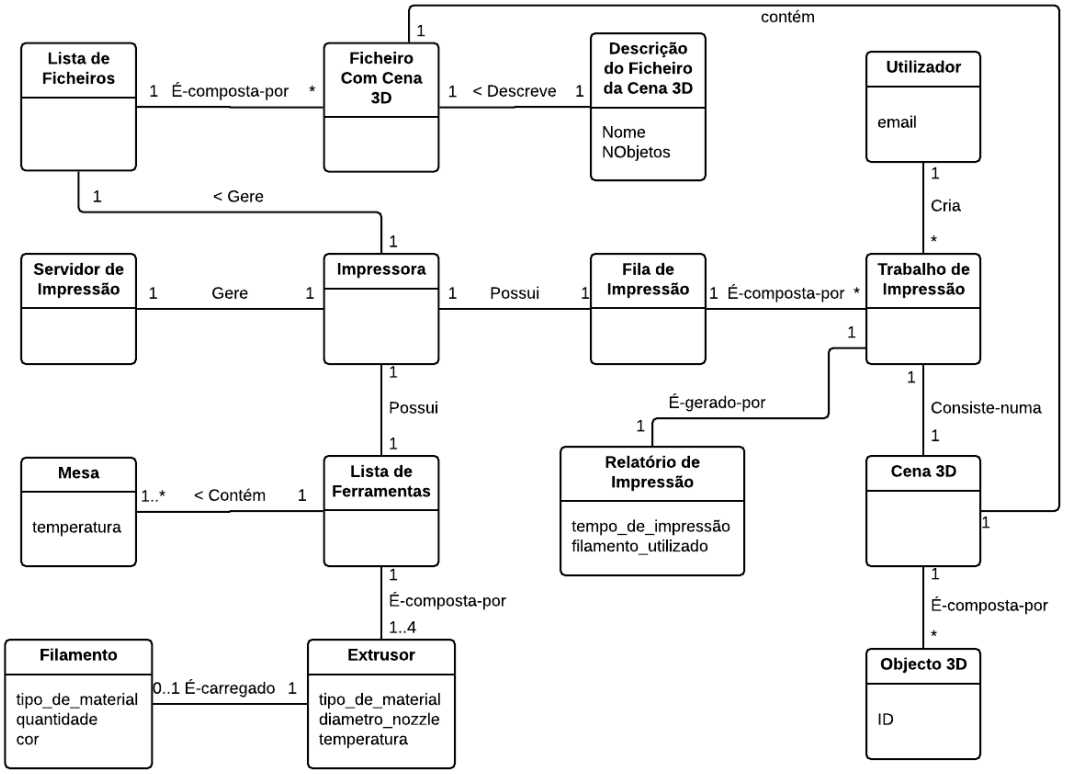


FIGURA 5.1: Modelo de Domínio.

5.2 Arquitetura de Software

Nesta secção é apresentada a arquitetura de software da solução começando pela definição do seu âmbito. De seguida será utilizado o modelo de vistas n+1, constituído pela vista lógica, vista de implementação, vista de fluxo de dados e vista de casos de uso.

5.2.1 Âmbito

Antes de apresentar a arquitetura da solução é importante clarificar o seu âmbito. Assim, a visão geral da solução e o seu âmbito podem ser observados no diagrama de componente de alto nível da Figura 5.1. A impressora BEE2B será uma impressora ligada à rede que irá permitir aos utilizadores profissionais recursos avançados num ambiente de software e hardware cuidadosamente projetado. O utilizador terá de interagir com uma aplicação para PC, onde irá escolher os modelos a serem impressos, um número reduzido de opções adicionais (por exemplo, qualidade, densidade e outras) e mandar imprimir.

A partir desse momento, é desejado ter o menor número possível de interações e diminuir o tempo de inatividade (*downtime*) da impressora.

Como podemos verificar no diagrama anterior o software BEE2B será dividido em dois componentes principais: o software para PC e o software da impressora. O primeiro é instalado

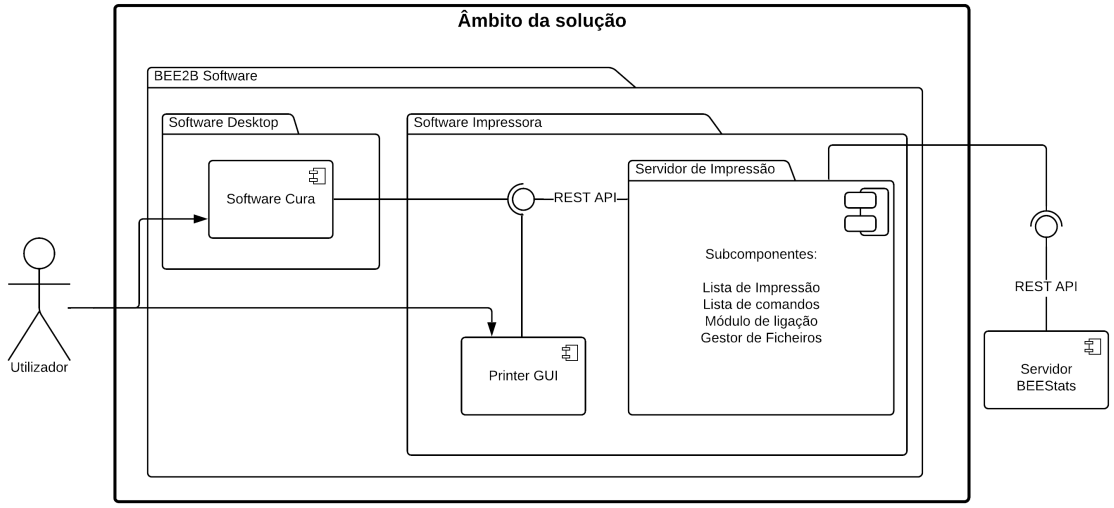


FIGURA 5.2: Âmbito da solução proposta.

no computador do utilizador e é desenvolvido usando software Ultimaker Cura. O último representa todo o software que irá correr no hardware da impressora, que serão o PrintServer BEE2B e a GUI BEE2B. Para o componente de software Cura BEE2B consiste na aplicação para *desktop* Cura, com algumas personalizações e *plugins* para acrescentar algumas funcionalidades a este software, como impressão por rede, informação da impressão atual, perfis de impressão ou emparelhamento da impressora, esta aplicação será executada pelos utilizadores comuns. O PrintServer BEE2B será responsável por todas as interações com o firmware da impressora, pela gestão da fila de impressão e gestão de ficheiros. Este componente disponibiliza ainda uma REST API que fornece métodos para a GUI BEE2B e para o Cura BEE2B. A GUI BEE2B tem como principal objetivo proporcionar uma interface de fácil utilização para as principais funções da impressora, como monitorizar e gerir impressões, realizar operações de manutenção, como substituir filamento e substituir extrusor. Toda a comunicação entre módulos é feita através de uma REST API do Servidor de impressão via protocolo HTTP.

O Servidor Servidor de Telemetria da BEEVERYCREATIEVE (BEEStats) é o servidor de Telemetria da BEEVERYCREATIVE, para onde serão transferidos os dados sobre a forma como o utilizador utiliza a impressora 3D. Este servidor apesar de ser necessário para a implementação da solução fica fora do âmbito da mesma pois já se encontra funcional.

5.2.2 Vista Lógica

Para a vista lógica desenvolveu-se o diagrama de pacotes para a GUI BEE2B representado na figura 5.3.

Podemos verificar que para a GUI BEE2B foi utilizado o padrão de design *Model View Controller* (MVC). Para os pedidos à REST API do PrintServer BEE2B foi utilizada a biblioteca python requests.

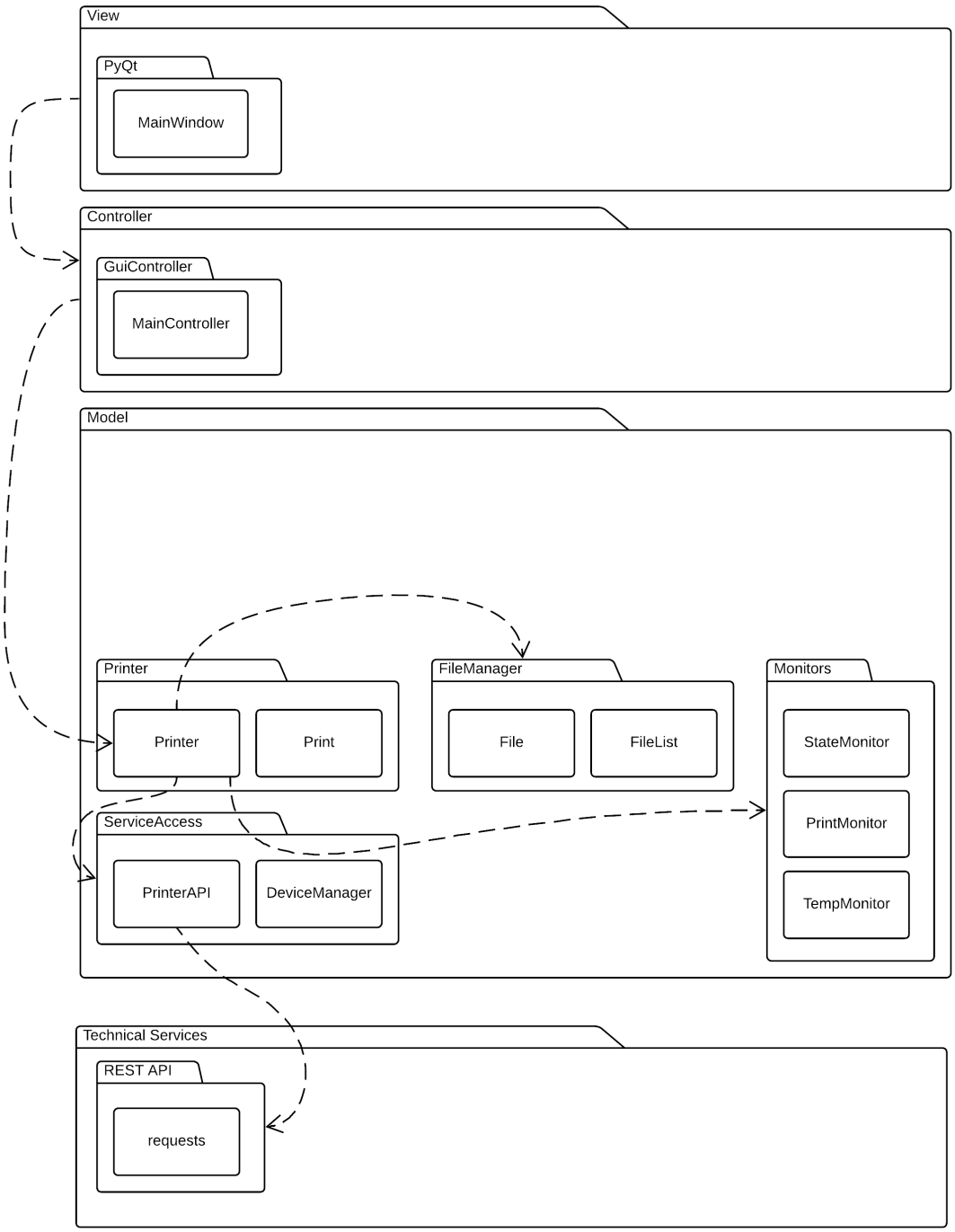


FIGURA 5.3: Diagrama de Pacotes da visão lógica da GUI BEE2B.

5.2.3 Vista de *deployment*

A vista de *deployment* é representada pelo diagrama de *deployment* da Figura 5.4. Esta vista mostra a infraestrutura física onde os componentes irão estar colocados.

O Servidor de impressão PrintServer BEE2B e a GUI BEE2B irão funcionar no mesmo equipamento físico, uma Raspberry pi 3 B+ com um sistema operativo Raspbien. O BEECura estará disponível para as três plataformas (Linux, Window e Mac) e irá comunicar com o PrintServer BEE2B através de uma REST API. A GUI também comunica com o PrintServer através desta API.

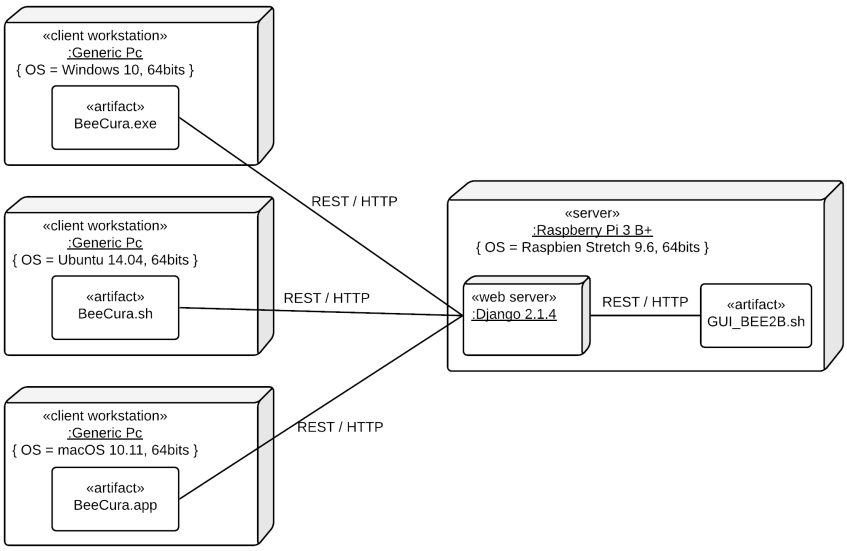


FIGURA 5.4: Vista de *deployment*.

5.2.4 Vista de Dados

A vista de dados faz uma visão geral dos fluxos dos dados e esquemas de persistência de dados. Podem ser usados diagramas de classes (Figura 5.16) e diagramas de atividade (Figura 5.5).

O caso de uso 1, iniciar uma impressão a partir do PC é um bom exemplo para perceber o fluxo de dados entre todos os componentes (5.5). O Utilizador dá início à impressão no software BEECura, o ficheiro é transferido para o servidor de impressão PrintServer BEE2B que dá início à impressão. Quando a GUI BEE2B é notificada do novo estado da impressora (A Imprimir) pede os dados da impressão e apresenta-a no ecrã, quando terminar a impressão é apresentado um relatório e pede ao utilizador para retirar o objeto e limpar a mesa.

5.3 Design

Para a realização dos use cases foram desenvolvidos dois tipos de modelos de objeto, dinâmico e estático. Modelos dinâmicos, como diagramas de sequência ou diagrama de estados, ajudam a desenhar a lógica e o corpo dos métodos. Modelos estáticos, como diagrama de classes ajudam a desenhar a definição dos *packages*, nomes das classes, atributos e métodos.

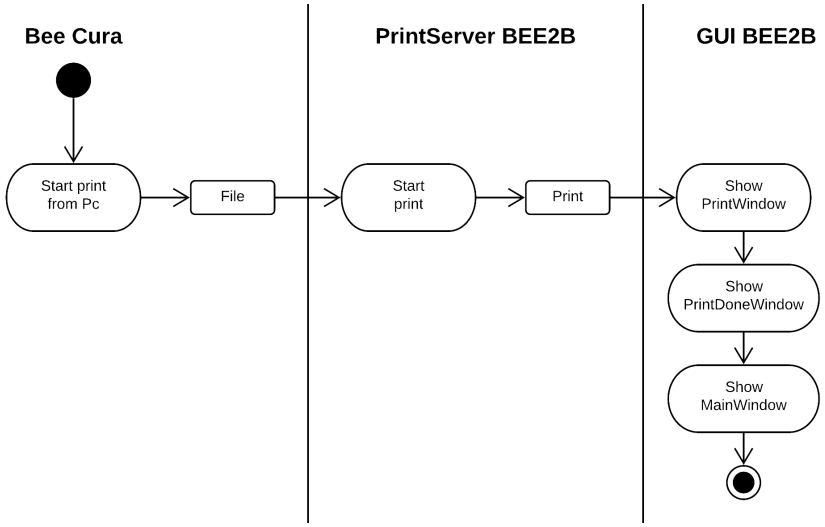


FIGURA 5.5: Vista de fluxo de dados para o UC01: iniciar impressão a partir do PC.

5.3.1 UCR00: Inicio da aplicação

Antes de iniciar a realização dos casos de uso propriamente ditos é necessário perceber o que acontece quando se inicia a GUI BEE2B. No próximo diagrama, figura 5.6, podemos ver a interação dos vários objetos quando iniciamos a aplicação. Começa por criar uma instância da classe *Printer*, figura 5.7, seguindo-se a criação do *MainController* usando o objeto *printer* criado como parâmetro. O *MainController*, por sua vez, vai criar a *MainView* passando novamente o objeto *printer* e o próprio *main_ctr* como parâmetros. A *MainView* cria a janela da aplicação recorrendo ao ficheiro gerado automaticamente no Qt Designer, este ficheiro fornece o método *setupUI* que prepara a janela para a apresentação. A *MainView* vai criar também as ligações com os botões *power_btn*, *settings_btn*, *maintenance_btn* e *home_btn* ligando-os aos métodos *power_off*, *goto_files*, *goto_settings* e *goto_home* respetivamente, e ligar o sinal *progress_value_change* ao método *set_progress_bar_value*. Por fim são chamados os métodos *refresh_files* e *update_filaments*, para atualizar a vista seleção de ficheiro com os ficheiros disponíveis na drive *Universal Serial Bus* (USB) e na impressora, e atualizar a informação dos filamentos na vista manutenção.

No diagrama de sequência da figura 5.7 é criada uma instância da classe Printer, a primeira ação do objeto *printer* é criar uma instância da classe *PrinterAPI*, o *print_server*. Este objeto é responsável por toda a comunicação com a API e só existe uma instância desta Classe durante o ciclo de vida da aplicação, por isso foi desenhada como um *Singleton*. O *print_server* vai então definir o *endpoint* da API a utilizar e a *printer* vai usar o seu método *set_token* para registar a GUI na API do *PrintServer* BEE2B.

Depois de concluir a criação do *print_server* é iniciado o monitor de estado, responsável por monitorizar o estado da impressora através de pedidos à REST API do *PrintServer* BEE2B, figura 5.8. Terminado o processo de inicialização do monitor são criados os objetos utilizados para gerir os ficheiros, tanto USB como Recentes, para uniformizar o acesso a estas classes foi utilizado um *Adapter*.

De seguida são criada as ferramentas da impressora, figura 5.9, aqui é criado o dicionário com todas as ferramentas disponíveis e atualizada a sua informação (figura 5.10) e dos respetivos filamentos, figura 5.11.

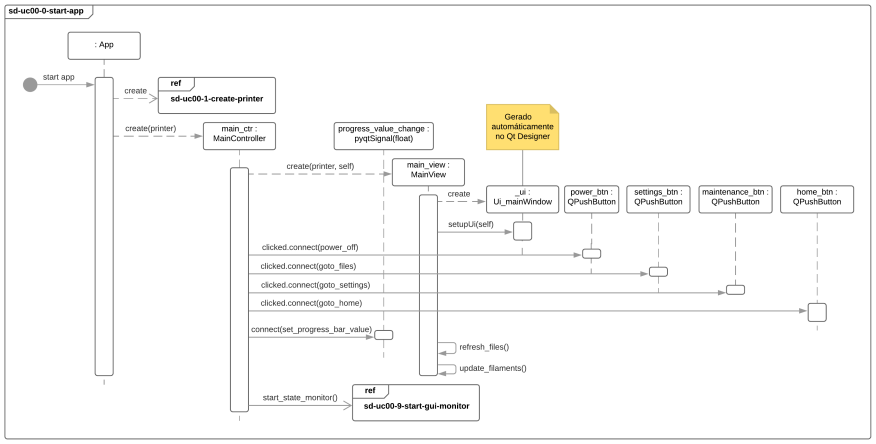


FIGURA 5.6: Início da Aplicação: Diagrama de Sequência do início da aplicação.

Por fim são carregados os ficheiros json com as informações relativas às cores, figura 5.12, aos textos das vistas, figura 5.13 e à lista de filamentos disponíveis para seleção manual, figura 5.15.

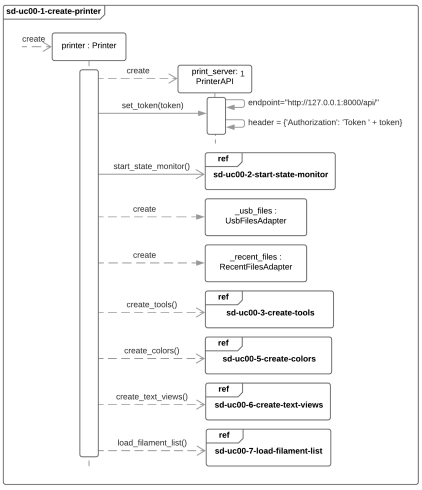


FIGURA 5.7: Início da Aplicação: Diagrama de Sequência criar impressora.

A figura 5.8 mostra a criação do monitor de estado, é criada uma *deamon thread* (*threads* que termina automaticamente com o fecho da aplicação), *state_monitor* e iniciada. Durante toda a aplicação de dois em dois segundos, a variável da _printer_state da *Printer* é atualizada com o estado da impressora, fazendo chamadas sucessivas ao método *get_state* do *print_server*.

As ferramentas são guardadas no dicionário *tools*, na figura 5.9 podemos ver que inicialmente são criados quatro extrusores vazios e guardados os seus ids como chaves ('id00', 'id01', 'id02' e 'id03'). De seguida, são atualizadas as ferramentas, figura 5.10, e os respetivos filamentos, figura 5.11.

No diagrama da figura 5.10,podemos ver como são atualizadas as ferramentas. Primeiro, com o método *get_tools* da *print_server* é obtido o estado atual das ferramentas e guardado no

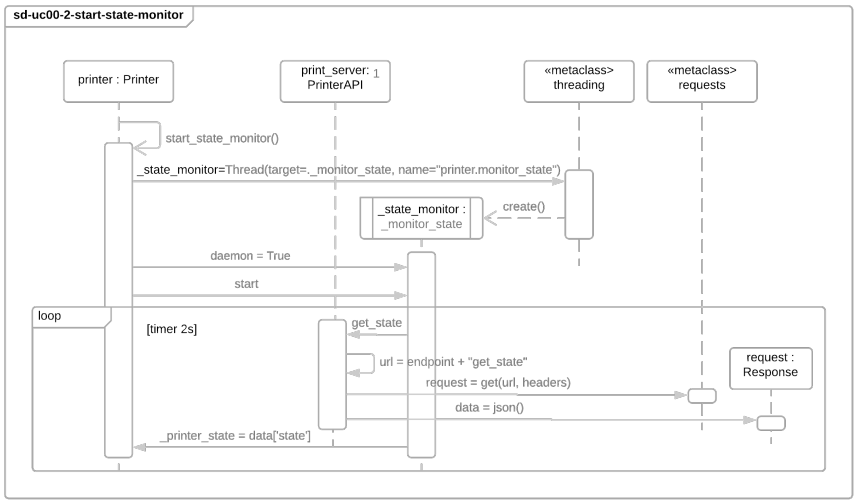


FIGURA 5.8: Início da Aplicação: Diagrama de Sequência Iniciar Monitor de Estado.

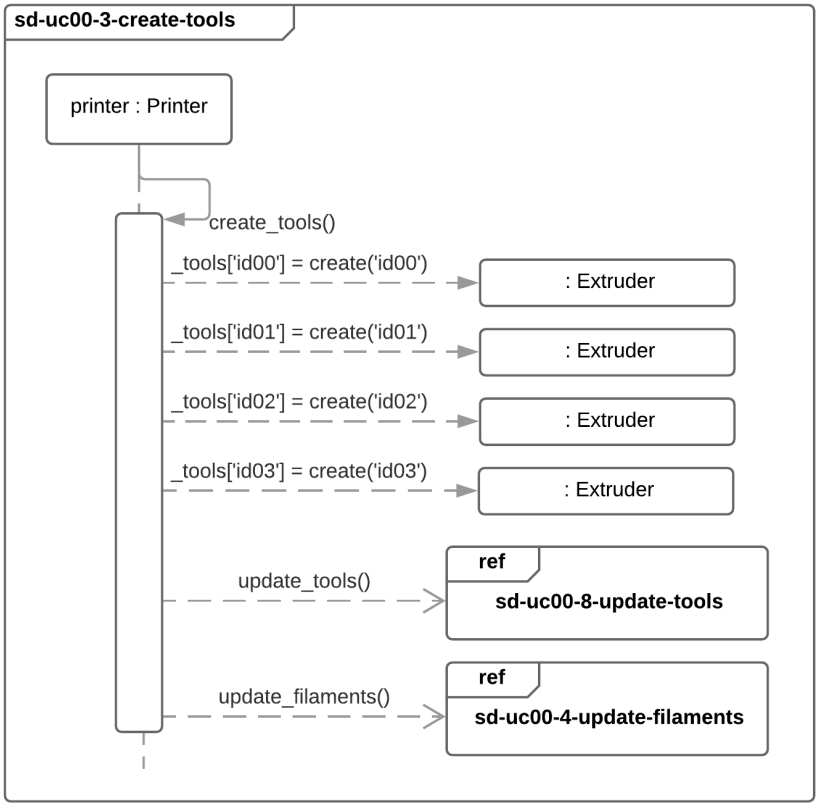


FIGURA 5.9: Início da Aplicação: Diagrama de Sequência Criar Ferramentas.

dicionário *tools_data*, depois percorre este dicionário e para cada ferramenta que não seja nula atualiza a informação da respetiva ferramenta na variável *tools*

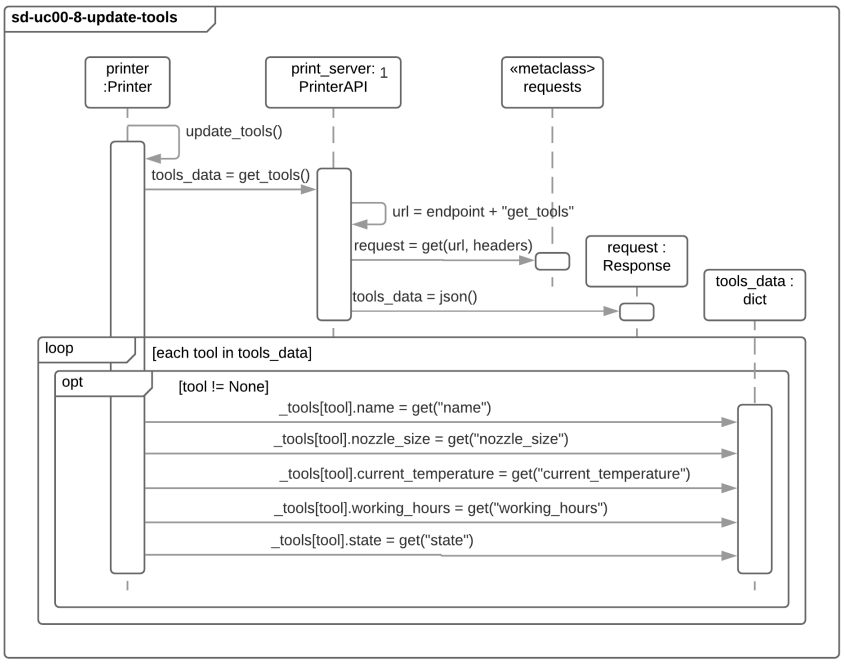


FIGURA 5.10: Início da Aplicação: Diagrama de Sequência Atualizar Ferramentas.

A figura 5.11 mostra como serão atualizadas as informações dos vários filamentos. Inicialmente é criada uma lista com todas as ferramentas disponíveis (*tools_list*) e depois, através do método do *print_server get_filaments*, é obtido um dicionário com a informação dos filamentos. Se este dicionário não estiver vazio (nenhum filamento carregado), para cada ferramenta no dicionário é criado uma instância da classe *Filament* com a informação do filamento dessa ferramenta, e guarda-o na ferramenta correspondente da variável *tools*. É ainda removido da lista *tools_list* cada ferramenta presente no dicionário *filaments_data*. Por fim, para cada ferramenta na lista *tools_list* é definido o filamento como *None*, ou seja, o filamento não está carregado.

Depois de atualizadas todas as ferramentas, são carregados os vários ficheiros json. Para carregar as cores utilizadas nos filamentos, figura 5.12, é criada uma instância da classe *Colors*, esta instância vai inicializar o dicionário *_colors* com as cores a partir de um ficheiro json presente nos *resources*.

Depois são carregados os textos das várias vistas para a língua ativa, figura 5.12, para isso, é criada uma instância da classe *Text*, que inicializa o dicionário *_text* com todo o texto necessário às diversas vistas da GUI BEE2B. O ficheiro carregado é dependente do idioma atualmente selecionado.

É também carregada uma lista de filamentos disponíveis para o utilizador selecionar manualmente durante o use case UC07: Trocar Filamento. Esta lista será guardada na variável *filaments* do objeto *printer*.

Por fim, é iniciado o monitor da GUI, figura 5.15, que será responsável por monitorizar a alteração do estado da impressora e atualizar à GUI sempre que a impressora mudar de estado.

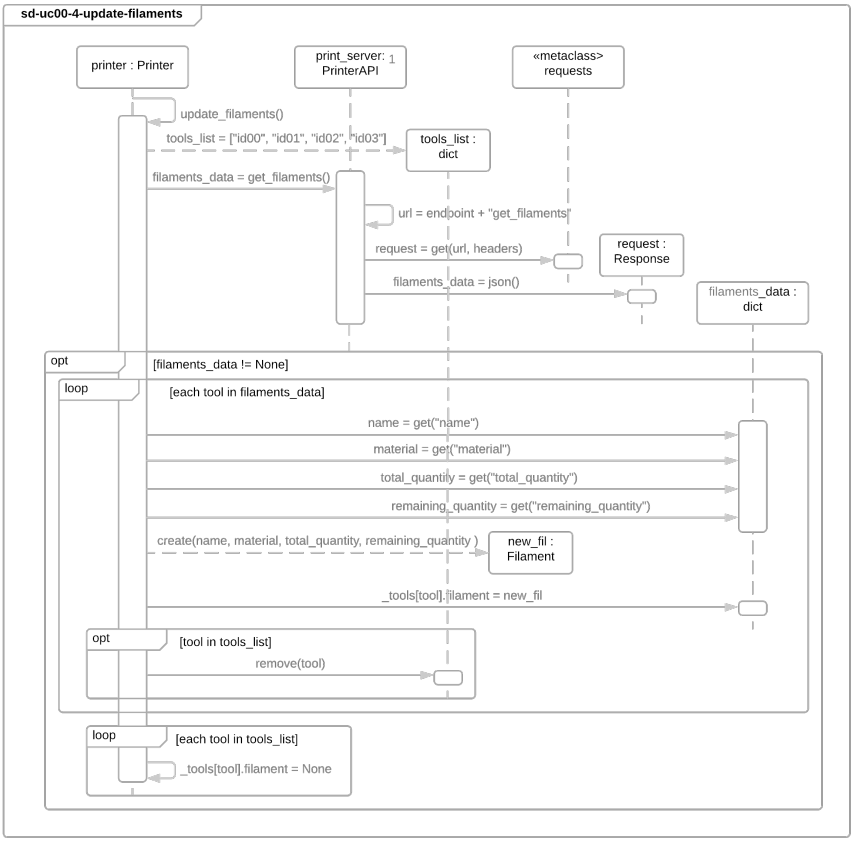


FIGURA 5.11: Início da Aplicação: Diagrama de Sequência Atualizar Filamentos.

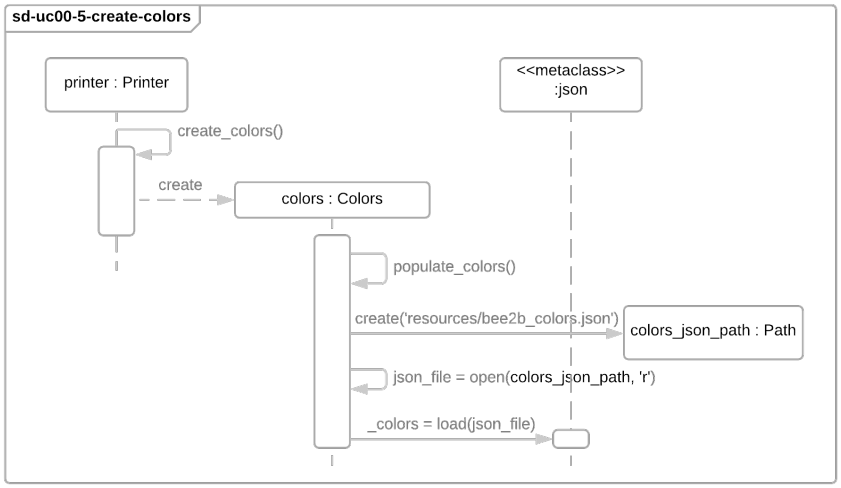


FIGURA 5.12: Início da Aplicação: Diagrama de Sequência Criar cores.

Como podemos verificar no diagrama de classes da Figura 5.16, para a implementação do arranque da aplicação é necessário desenvolver os seguintes métodos da API do servidor de impressão BEE2B:

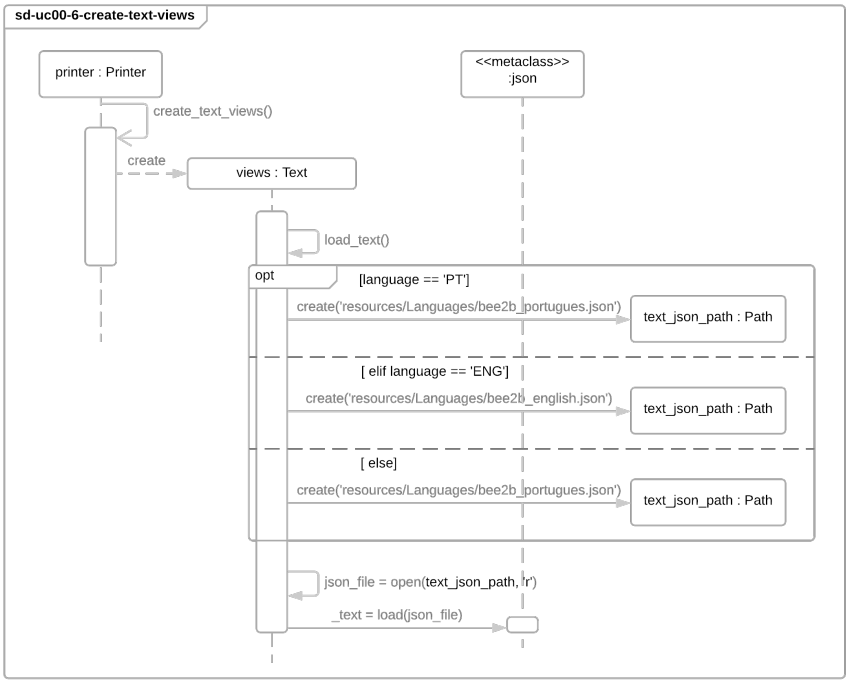


FIGURA 5.13: Início da Aplicação: Diagrama de Sequência Criar textos de vistas.

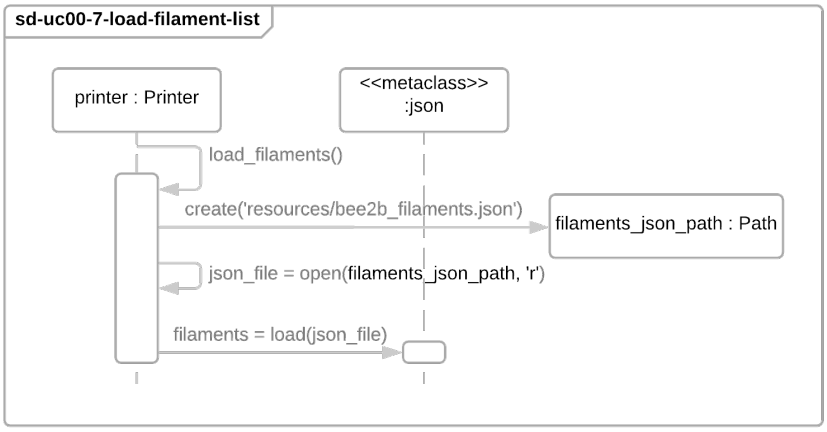


FIGURA 5.14: Início da Aplicação: Diagrama de Sequência Carregar lista de filamentos.

- /get_tools (GET)
- /get_printer_state (GET)

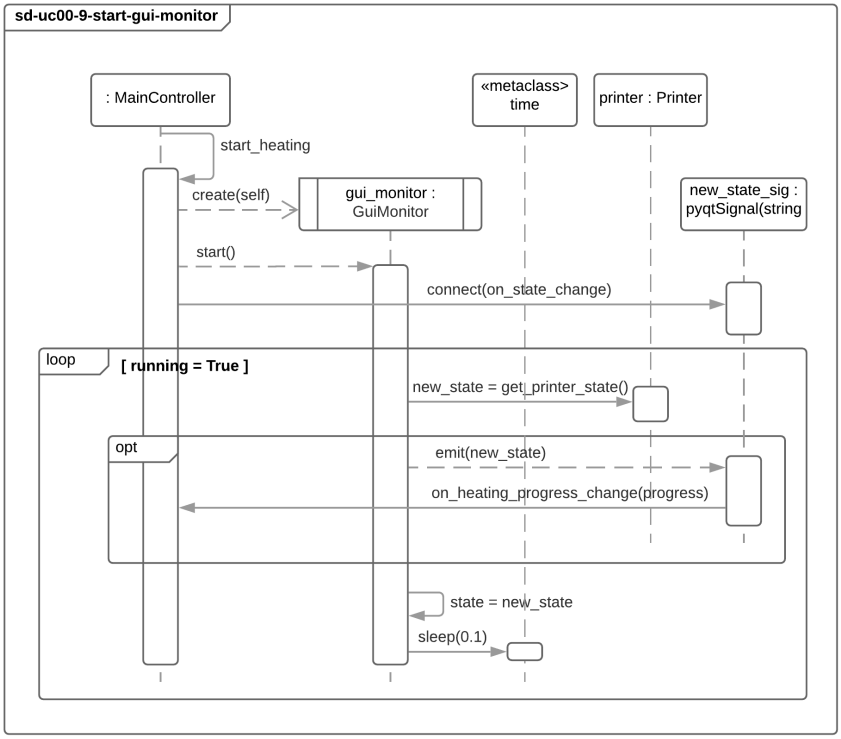


FIGURA 5.15: Início da Aplicação: Diagrama de Sequência Iniciar monitor de GUI.

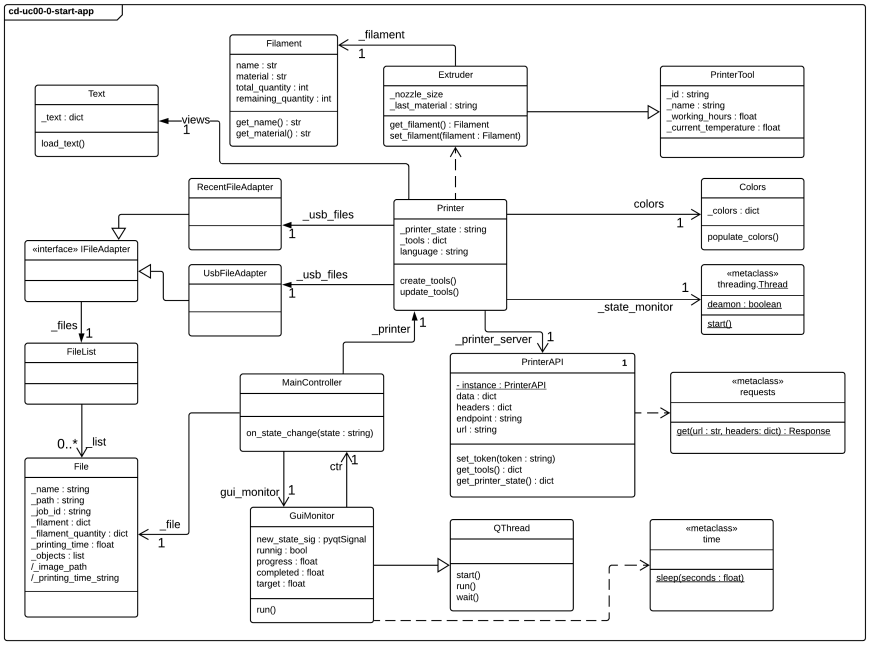


FIGURA 5.16: Início da Aplicação: Diagrama parcial de classes.

5.3.2 UCR02: Iniciar Impressão a Partir de Drive USB

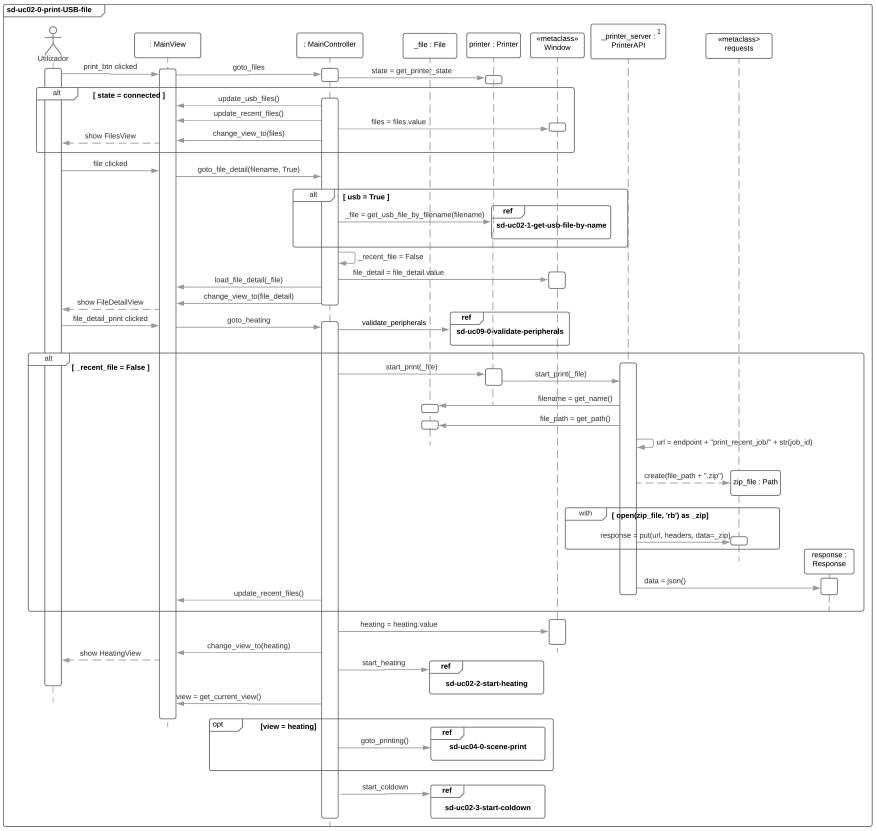


FIGURA 5.17: Diagrama de Sequência UC02: Imprimir a partir de drive USB.

O Diagrama de Sequência da figura 5.17, mostra a interação dos objetos durante todo o processo de impressão. Na vista *home*, o utilizador clica no botão *print_btn* que inicia o método *goto_files* do *controller*. O *controller* vai obter o estado da impressora e, se esta estiver pronta a imprimir (*connected*), atualiza as listas de ficheiros USB e recentes, por fim, muda a vista para a vista files.O método *change_view_to* do *main_view* verifica a vista pedida e prepara-a para seleção do ficheiro a imprimir.

Na vista de seleção de ficheiro o utilizador clica num dos ficheiros e chama o método *goto_file_detail* com o nome do ficheiro (extraído do botão clicado) e a *flag* USB ativa. O *controller* vai obter o ficheiro através do método *get_usb_file_by_filename* da *printer*, na figura 5.18 podemos ver a interação da *printer* com os restantes objetos do domínio. Depois de obter o ficheiro, é passado como parâmetro ao *load_file_detail* da *main_view*, esta vai novamente preparar a janela para a nova vista e apresentá-la ao utilizador.

O utilizador deve agora clicar no botão *file_detail_print* que chama o *goto_heating* do *controller*. Este método vai inicialmente verificar os periféricos de impressão (este processo será detalhado no design do use case UC09: Validar Periféricos), depois chama o método *start_print* da *printer* com o ficheiro como parâmetro que por sua vez chama o mesmo método mas do *_print_server*. Este último método vai obter o endereço do ficheiro e fazer o pedido à API enviando esse ficheiro.

O *controller* vai agora atualizar a lista dos ficheiros recentes e mudar a vista para o Aquecimento, de seguida a impressora inicia o aquecimento como na figura 5.19.

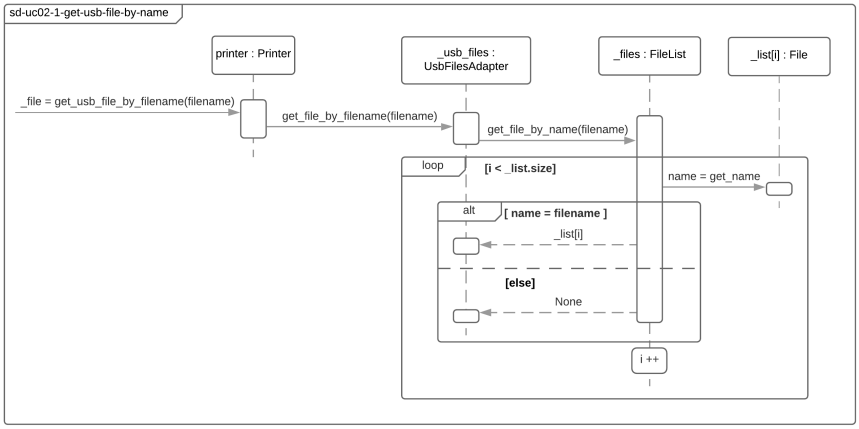


FIGURA 5.18: Diagrama de Sequência UC02: Obter ficheiro USB a partir de nome de ficheiro.

Depois do aquecimento o *controller* pede à *main_view* para mudar para a vista de impressão e inicia-a. O processo de impressão será detalhado no use case UC04: Imprimir cena 3D.

No final a impressora irá arrefecer para poder remover o objeto da mesa, figura 5.20.

Para obter o ficheiro USB a partir do seu nome, figura 5.18, a *printer* vai chamar o método *get_file_by_filename* do *_usb_files* e passar novamente o nome do ficheiro como parâmetro. Este método chama o mesmo método do *_files* que vai percorrer a sua lista interna e retornar o ficheiro com esse nome caso exista.

Quando a impressora inicia o aquecimento o *controller* inicia um monitor de aquecimento com o sinal *progress_sig* ligado ao método *on_heating_progress_change*. Este monitor vai fazer chamadas sucessivas ao *printer_api* para obter a temperatura das ferramentas. A cada chamada é transmitido o valor do progresso ao *controller* que por sua vez atualiza o valor de barra de progresso da GUI.Quando o processo terminar o *controller* vai parar o monitor, caso ainda esteja a correr.

Quando a impressão terminar será mostrada a vista de arrefecimento, nesta vista o *controller* vai criar um monitor de arrefecimento semelhante ao monitor de aquecimento mas com o sinal ligado ao método *on_coldown_progress_change* que irá atualizar a barra de progresso. No final do arrefecimento o *controller* pára o monitor de arrefecimento e pede à *main_view* para mostrar a vista de impressão concluída.

Em paralelo aos diagramas de sequência anteriores foi desenvolvido o diagrama de classes parcial da solução (Figura 5.21).

Como podemos verificar no diagrama de classes da Figura 5.21, para a implementação deste caso de uso é necessário desenvolver os seguintes métodos da API do servidor de impressão BEE2B:

- /get_state (GET)
- /start_print (PUT)
- /get_temperature (GET)

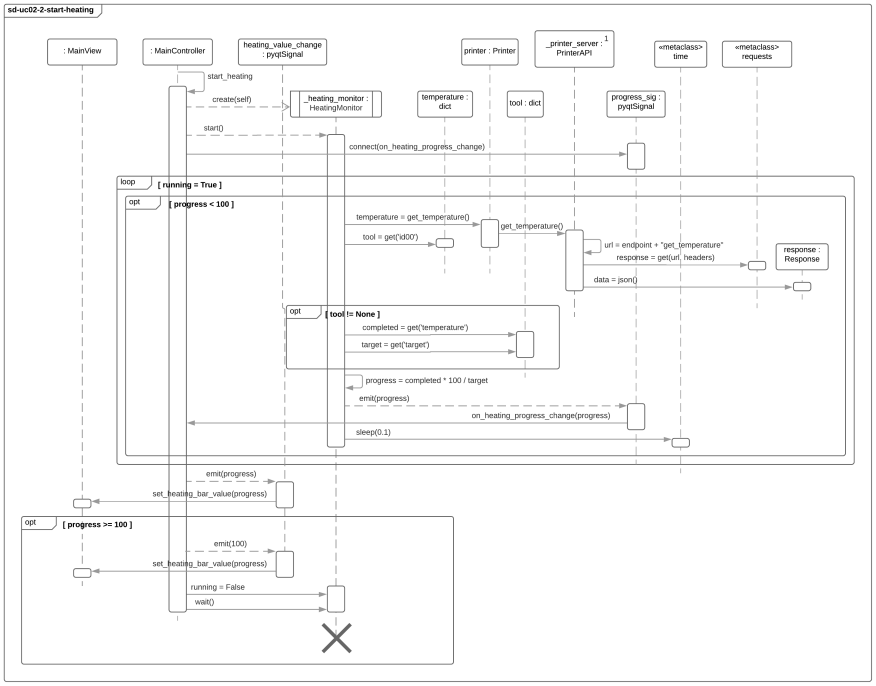


FIGURA 5.19: Diagrama de Sequência UC02: Iniciar Aquecimento.

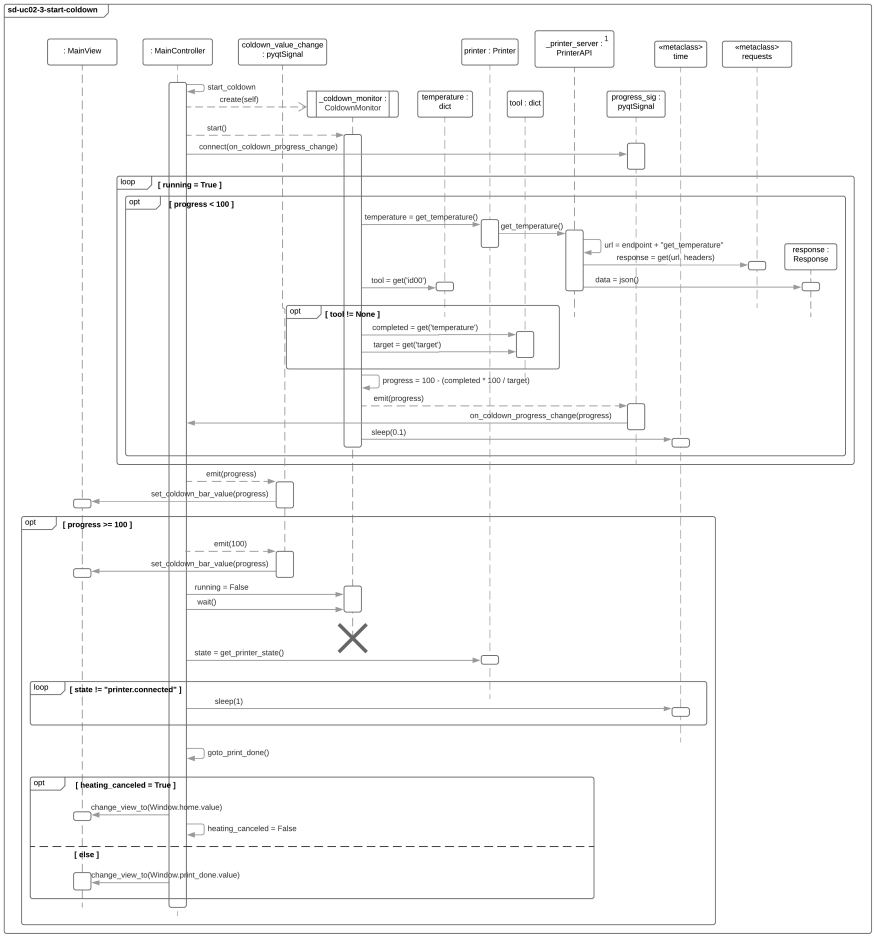


FIGURA 5.20: Diagrama de Sequência UC02: Iniciar Arrefecimento.

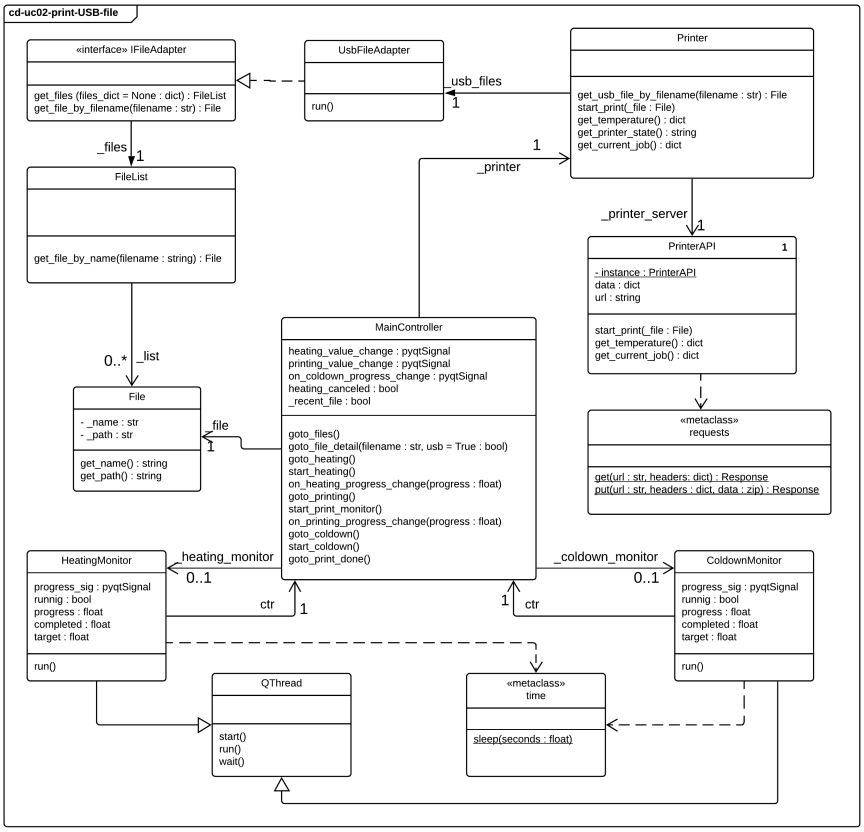


FIGURA 5.21: Diagrama de Classes Parcial UC02: Imprimir a partir de uma drive USB.

5.3.3 UCR03: Iniciar Impressão a partir de um Ficheiro Recente

Este caso de uso é semelhante ao UC02: Imprimir a partir de uma drive USB da secção 5.3.2. Neste caso de uso o utilizador vai escolher um ficheiro recente e com isso irá fazer uma chamada à API diferente, *print_recent_job* em vez de *start_print*, e não precisa de atualizar a lista de ficheiros recentes depois de imprimir. Nos dois seguintes diagramas, 5.22 e 5.23, podemos ver estas alterações refletidas. Nestes diagramas existem referências para as seguintes figuras do UC02: iniciar aquecimento - 5.19 e iniciar arrefecimento - 5.20, e para iniciar impressão, figura 5.25 do UC04.

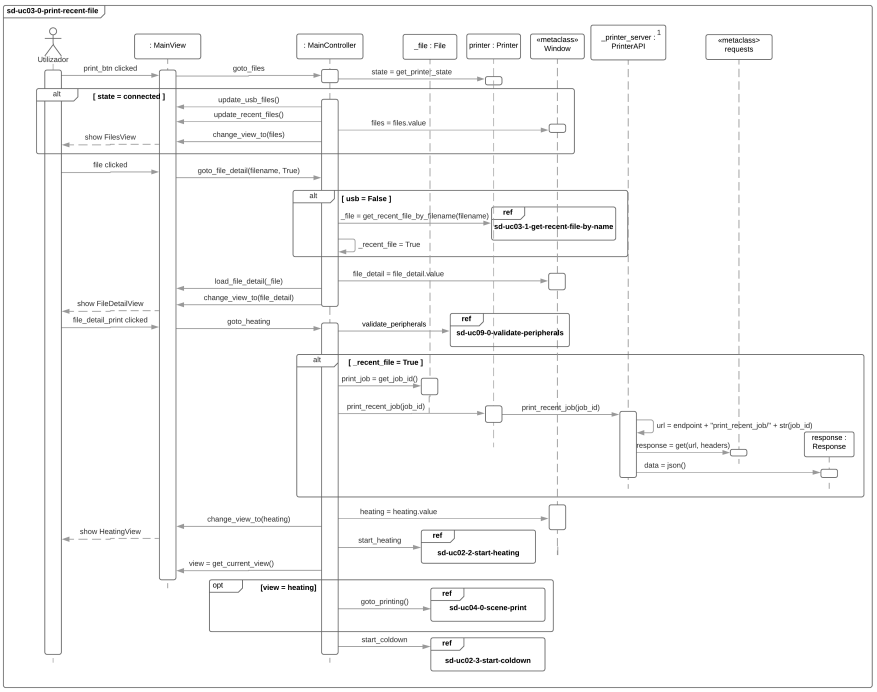


FIGURA 5.22: Diagrama de Sequência UC03: Iniciar Impressão a partir de um Ficheiro Recente.

Estas diferenças também podem ser verificadas no diagrama de classes parcial da solução para este caso de uso (Figura 5.24).Podemos ainda verificar neste diagrama de classes que é necessário desenvolver mais um método da API do servidor de impressão BEE2B:

- /print_current_job (GET)

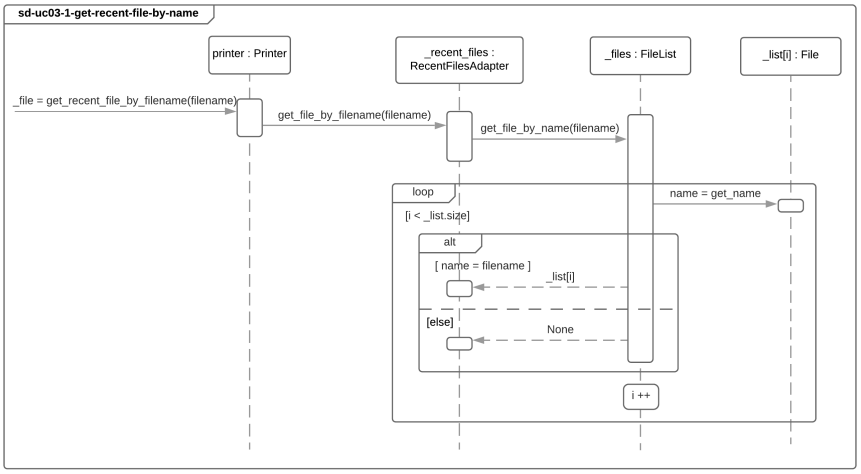


FIGURA 5.23: Diagrama de Sequência UC03: Obter ficheiro recente a partir de nome de ficheiro.

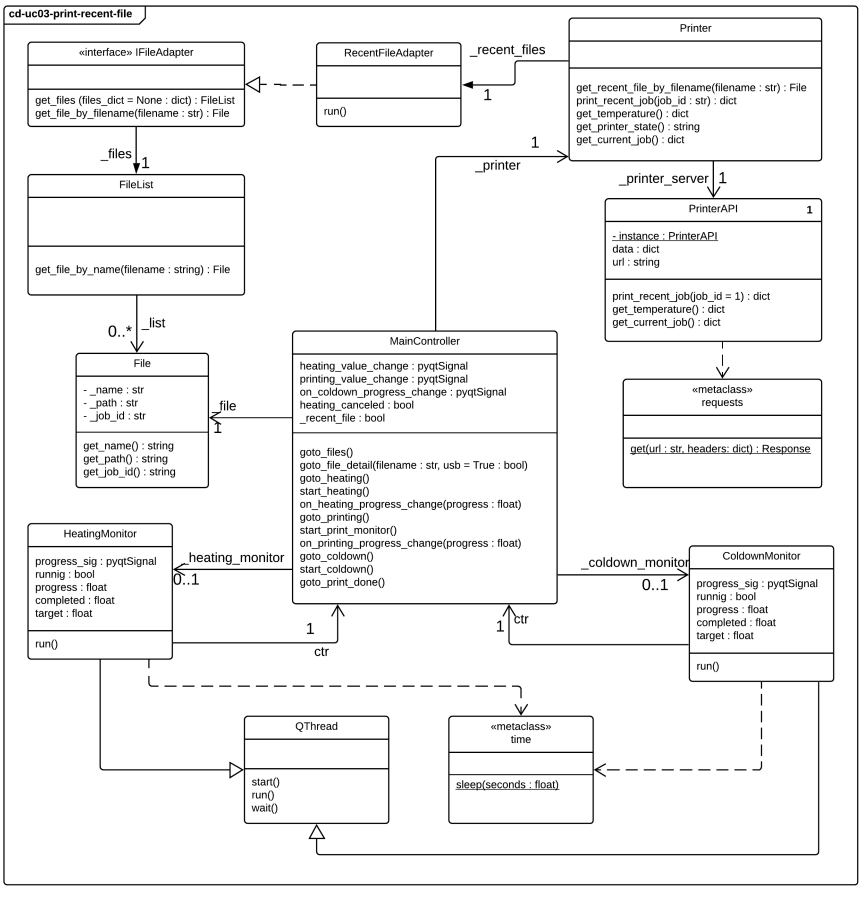


FIGURA 5.24: Diagrama parcial de Classes UC03: Iniciar Impressão a partir de um Ficheiro Recente.

5.3.4 UCR04: Impressão de uma cena 3D

Quando é iniciada a impressão, figura 5.25, o *controller* vai invocar o método *create_board* passando o ficheiro com parâmetro para construir a cena 2D vista de topo e mudar a vista a imprimir é depois iniciado o monitor de impressão detalhado no diagrama da figura 5.26.

A qualquer momento o utilizador pode cancelar, pausar/retomar a impressão.

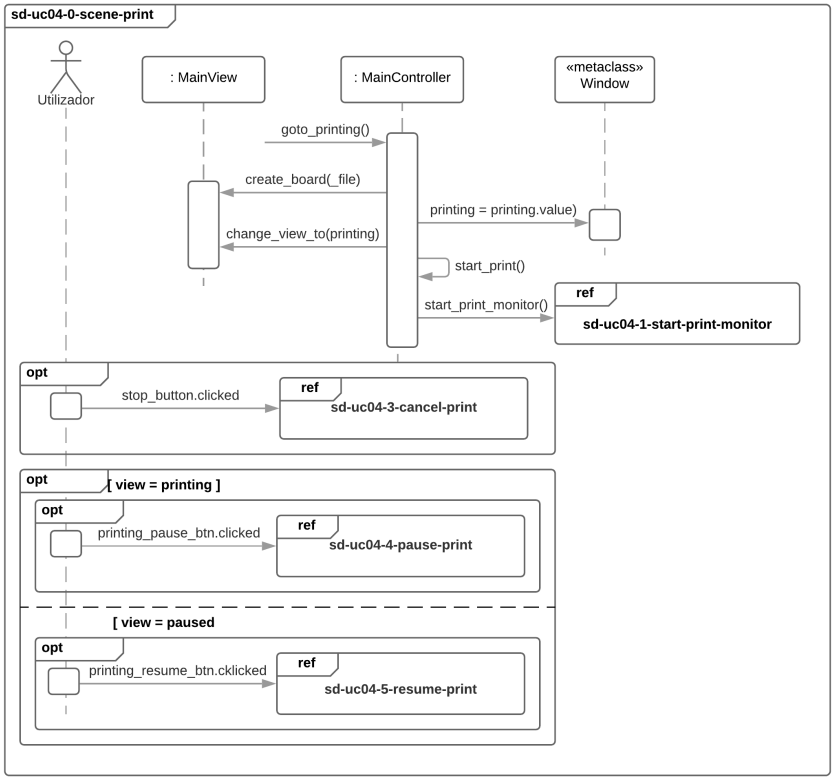


FIGURA 5.25: Diagrama de Sequência UC04: Impressão de cena 3D.

Quando a impressora inicia uma impressão o *controller* inicia um monitor de impressão (figura 5.26) com o sinal *progress_sig* ligado ao método *on_printing_progress_change*. Este monitor vai fazer chamadas sucessivas ao *printer_api* para obter o estado da impressão atual. A cada chamada é transmitido o valor do progresso ao *controller* que por sua vez atualiza o valor de barra de progresso da GUI, figura 5.27.

Quando o utilizar clica no botão *printing_stop_btn* invoca o método do *controller stop_printing*, este, por sua vez, invoca o *toggle_stop_frame* do *main_view* para mostrar a mensagem de confirmação ao utilizador. Se o utilizador confirmar, ou seja, clicar em *printing_stop_yes_btn*, a mensagem é escondida e invocado o método *cancel_print* do *controller*, que invoca o mesmo método da *printer* e esta volta a invocar o mesmo método no *_printer_server*, este método vai fazer o pedido à API e retornar um dicionário com o resultado. Com a impressão parada o *controller* invoca o *clear_board* para limpar a cena 2D e destrói o *printing_monitor*, de seguida o *controller* invoca o *start_coldown* do UC02, figura 5.20.

Ao clicar em *printing_pause_btn*, a GUI mostra ao o utilizador a mensagem de confirmação de pausa, figura 5.29, se o utilizador confirmar esta ação, a *main_view* chama o método *pause_print*, que invoca o mesmo método da *printer* e que por sua vez chama novamente o

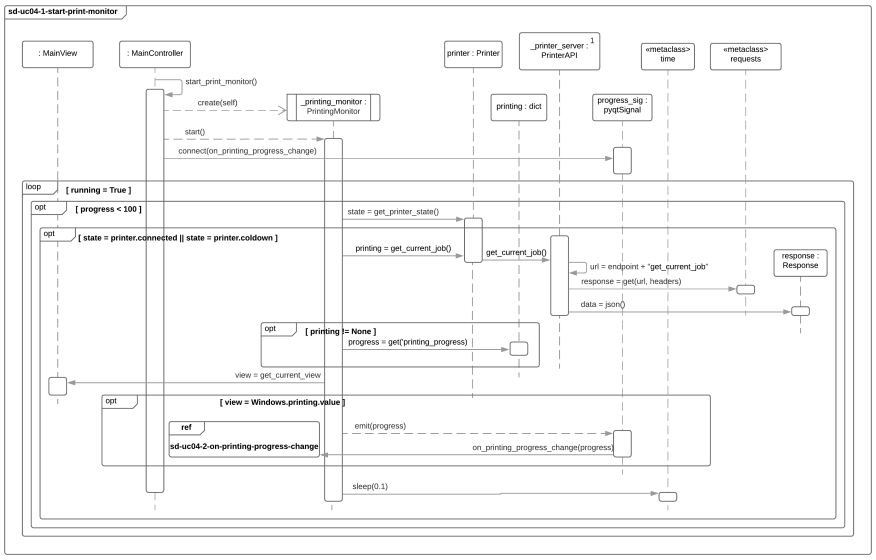


FIGURA 5.26: Diagrama de Sequência UC04: Iniciar Monitor de Impressão.

mesmo método mas agora no *printer_server*, este, como já é habitual, é o responsável pela chamada à REST API, invocando desta vez o método *pause_print* e retornando o dicionário resultante à impressora que por sua vez o retorna ao *controller*. Com a impressora em *pause* o *controller* invoca o *change_view_to* à *main_view* para mostrar a vista em *pause*.

Quando a impressão está em *pause*, o utilizador pode retomar a impressão clicando em *printing_resume_btn*, figura 5.30, esta ação vai chamar o método *resume_print* encadiadamente até ao *print_server*, começando no *controller* e passado pela *printer*. Este método vai efetuar o pedido à API e retornar o resultado até ao *controller*.

Na figura 5.31 vemos o diagrama parcial de classes para este caso de uso.Podemos ainda verificar neste diagrama de classes que é necessário desenvolver mais três método da API do servidor de impressão BEE2B:

- /cancel_print (GET)
- /pause_print (GET)
- /resume_print (GET)

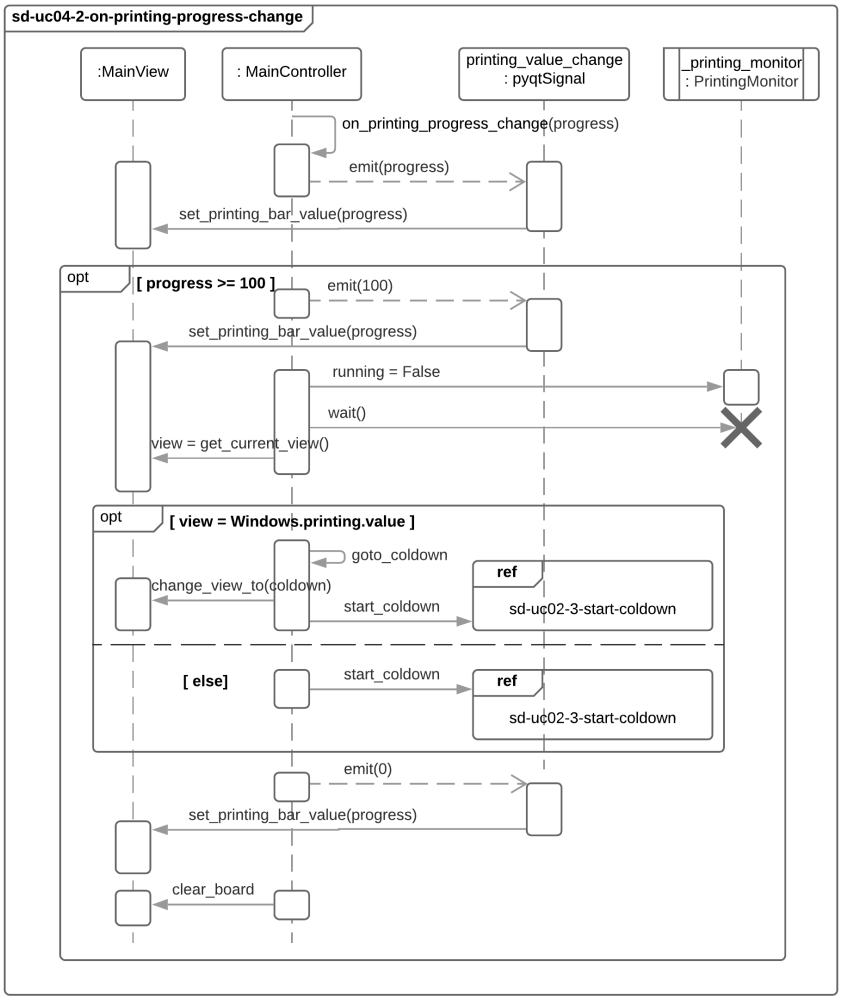


FIGURA 5.27: Diagrama de Sequência UC04: Na mudança de progresso de impressão.

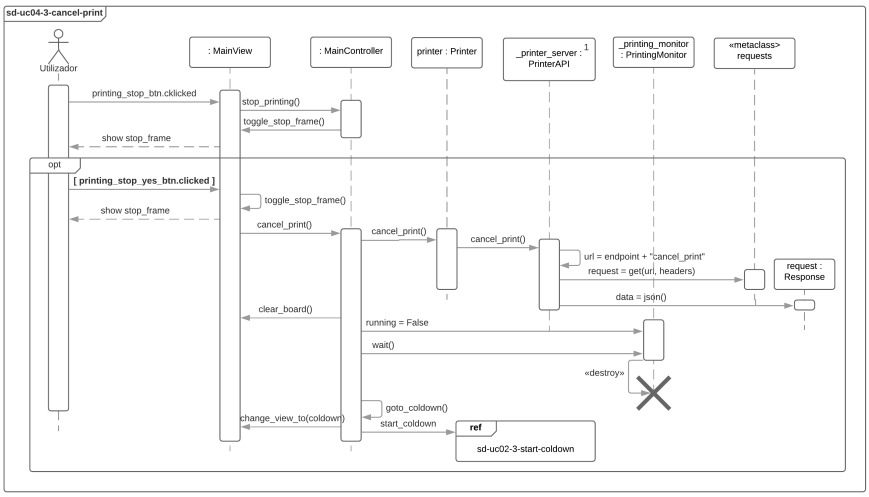


FIGURA 5.28: Diagrama de Sequência UC04: Cancelar a impressão.

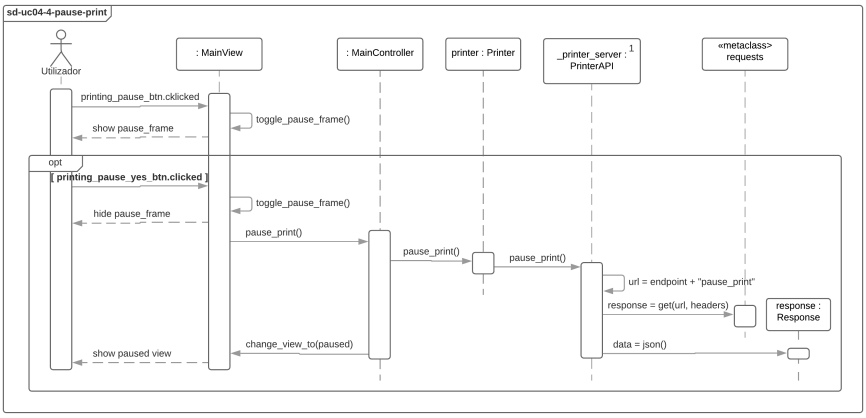


FIGURA 5.29: Diagrama de Sequência UC04: Pausar a impressão.

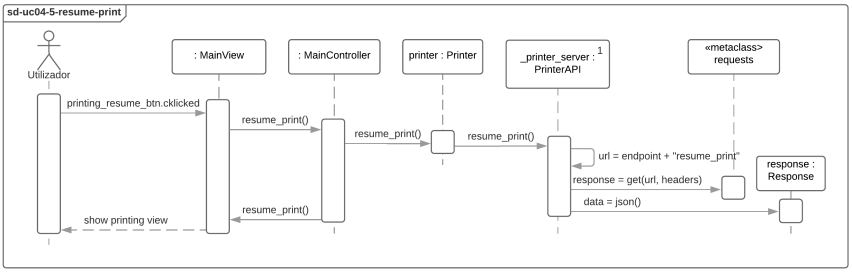


FIGURA 5.30: Diagrama de Sequência UC04: Retomar a impressão.

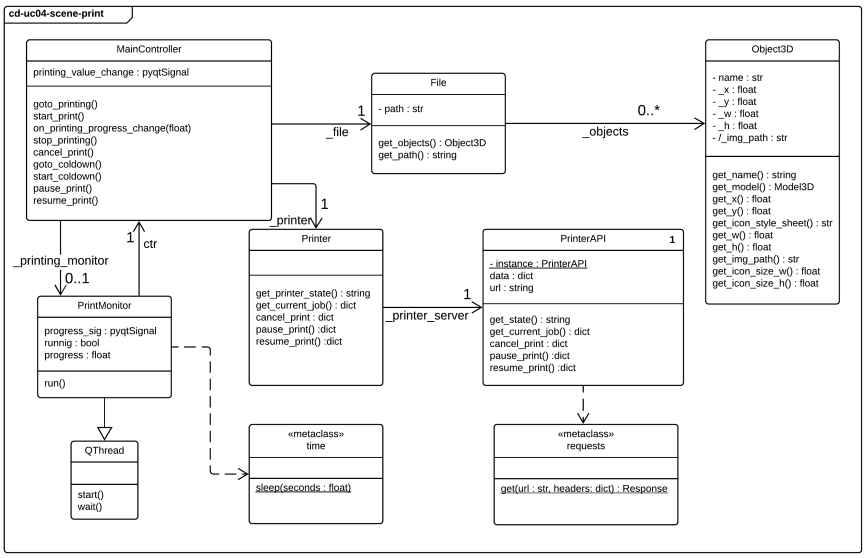


FIGURA 5.31: Diagrama parcial de Classes UC04: Impressão de uma cena 3D.

5.3.5 UCR07: Trocar Filamento

Nos próximos diagramas de sequência, vamos acompanhar a interação dos objetos de software no caso de uso UC07: Trocar Filamento. O utilizador pode mudar de filamento diretamente no menu de manutenção ou na vista dos detalhes do filamento, em ambos os casos, se o extrusor não tiver nenhum filamento carregado o botão muda o texto de trocar para carregar, nessa circunstância começaria na figura 5.34. Para este caso de uso, o extrusor já têm carregado algum filamento e o utilizador inicia-o clicando nos detalhes do filamento.

Como podemos ver na figura 5.32, quando o utilizador clica nos detalhes do filamento é chamado o método *goto_filament* do *controller* com o id da ferramenta selecionada como parâmetro. Para sabermos o filamento que está carregado nessa ferramenta precisamos primeiro obter a ferramenta. A impressora possui um dicionário com todas as ferramentas com o seu id como chave do dicionário, assim, o *controller* pede o dicionário à impressora e depois obtém a ferramenta selecionada através do id. Agora que já temos a ferramenta, podemos obter o filamento carregado com o método *get_filament* da ferramenta. Para consulta futura é guardada a variável *tool_selected* com o id da ferramenta selecionada.

Já com a informação do filamento carregado o *controller* pede à *main_view*, através do método *update_filament_detail* para atualizar a informação do filamento que é apresentado na GUI. Este método têm como parâmetros o objeto filamento e uma *flag* indicando se o filamento está carregado. Depois de atualizada a informação da GUI o *controller* chama o *change_view_to* para mudar para a vista detalhes do filamento.

Nesta vista, o utilizador clica em trocar filamento e confirma essa intenção no aviso seguinte. O *main_view* vai agora chamar o *goto_change_filament* que será detalhado na figura .

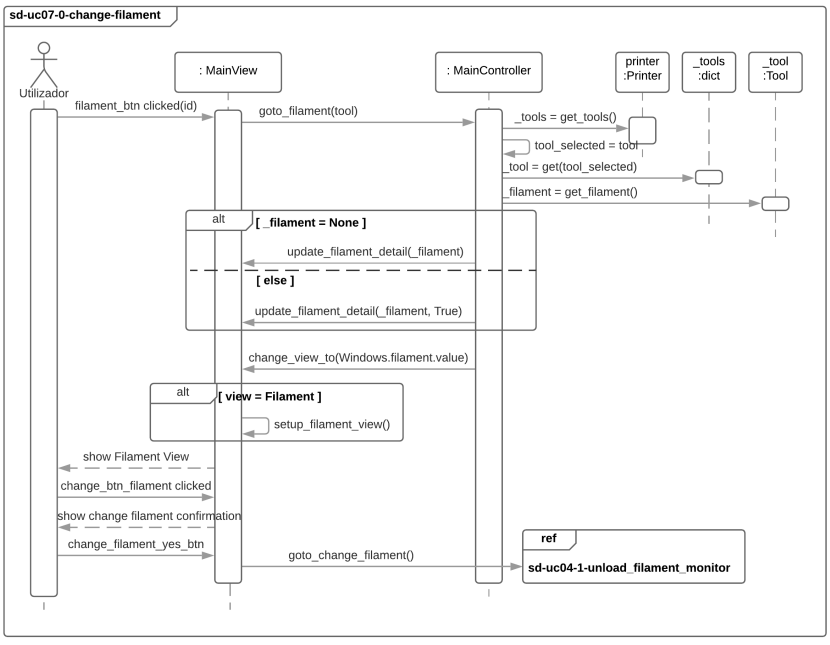


FIGURA 5.32: Diagrama de Sequência UC07: Trocar Filamento.

Quando o utilizador clica em trocar filamento (figura 5.33), o *controller* pede ao *main_view* para mudar para a vista a descarregar filamento e cria um *_unload_filament_monitor* responsável pela monitorização do processo de descarregado de filamento. Este monitor vai fazer

pedidos sucessivos à API através do método *unload_filament* da impressora passando o id da ferramenta selecionada, que por sua vez chama o mesmo método no *printer_server*. Quando o parâmetro de paragem que obtemos na resposta do pedido à API for verdadeiro o monitor emite um sinal para o *on_unload_filament_progress_change* que por sua vez termina o monitor e atualiza a GUI mudando para a vista remover bobine, figura 5.34.

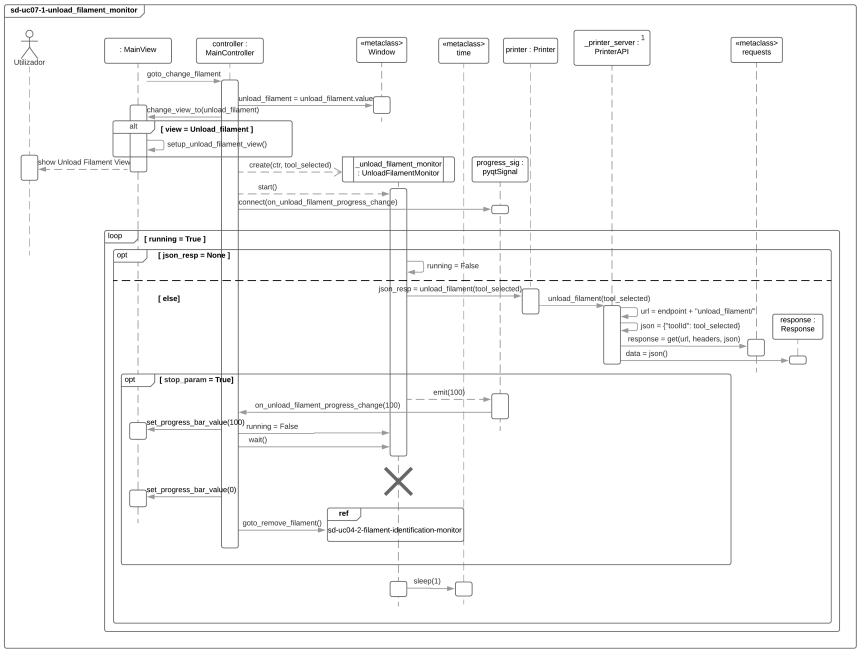


FIGURA 5.33: Diagrama de Sequência UC07: Monitor de Descarregamento de Filamento.

Agora que a impressora 3D já removeu o filamento é necessário que o utilizador remova manualmente a bobine, a GUI fica à espera que o utilizador confirme esta ação. Quando o utilizador confirma a ação o *controller* pede à *main_view* para mudar para a vista novo filamento, cria e inicia um novo monitor, desta feita, para monitorizar a identificação automática do filamento. Por fim o *controller* liga o sinal do monitor ao método *on_filament_identify_progress_change*. O *_filament_identify_monitor* vai fazer chamadas sucessivas à API tal como explicado anteriormente mas agora usando o método *identify_filament* passando novamente o id da ferramenta como parâmetro. Quando a impressora detetar um novo filamento no sensor, o método da API anterior retorna o parâmetro de paragem a verdadeiro juntamente com a informação do filamento.

O utilizador deve agora encostar a nova bobine ao sensor da impressora 3D, quando é detetado o novo filamento a GUI mostra os detalhes do filamento identificado e pede confirmação ao utilizador. Depois de confirmar, se o material detetado for diferente em cor ou tipo, do último material utilizado o utilizador vê uma mensagem de alerta a avisar desse fato e pede nova confirmação. Por fim, segue para a vista adicionar filamento, que podemos ver na figura 5.34

Na vista de adicionar filamento, figura 5.35, o *controller* cria um monitor de adição de filamento que vai fazer chamadas sucessivas do método add_filament à printer_API através da impressora. Este monitor fica à espera que o utilizador coloque a bobine no suporte e empurre a ponta do filamento pelo furo correspondente até que este seja puxado automaticamente. Depois será iniciado o processo de carregamento do filamento, figura 5.36.

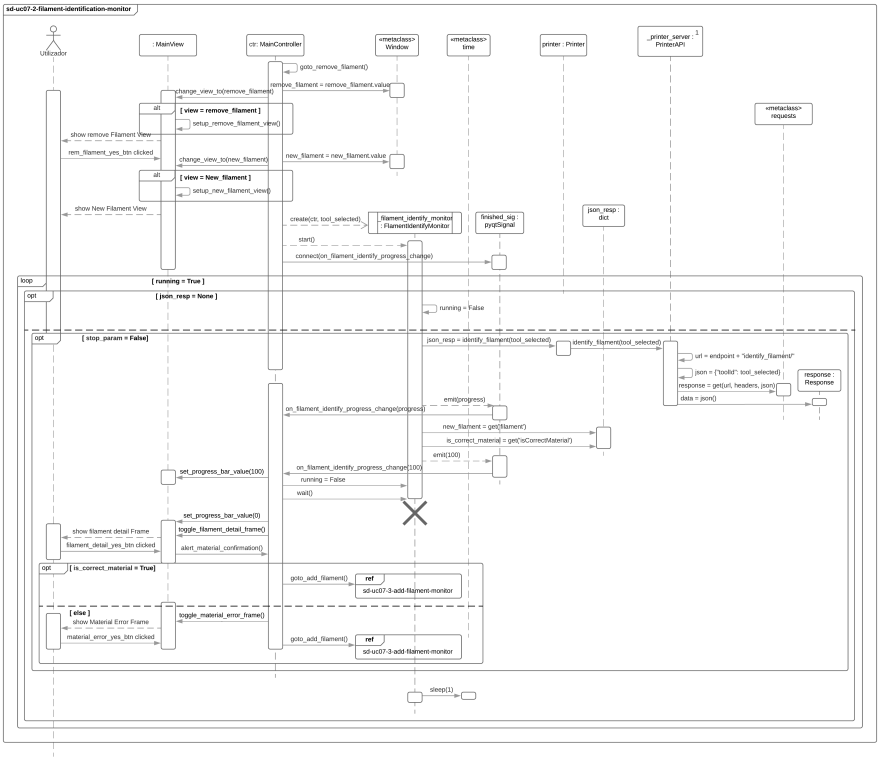


FIGURA 5.34: Diagrama de Sequência UC07: Monitor de Identificação de Filamento.

No diagrama da figura 5.36 vemos o processo de carregamento do filamento, depois do *controller* pedir à *main_view* para mostrar a vista de carregar filamento e criado e iniciado um novo monitor, agora para controlar o progresso do filamento. Este monitor funciona tal como os anteriores para uso o método *load_filament* da *printer_API* e quando concluído irá atualizar os filamentos como mostra a figura 5.37

Como podemos observar na figura 5.37, a primeira coisa que é feita quando é chamado o método *update_filaments* do *controller* é a atualização da informação das ferramentas como na figura 5.10 do Iniciar Aplicação, só depois é atualizada a informação dos filamentos. Para cada ferramenta, obtém-se o filamento que está carregado, esse objeto contém as informações necessárias, como nome, material e cor. Usa-se a cor para obter o estilo do botão conforme a cor e atualiza-se a GUI com o método *update_filament_info* da *main_view* passando como parâmetros os objetos *tool*, *_style* e *material*. Por fim, para cada extrusor sem filamento carregado é usado o estilo pré definido de filamento vazio para atualizar a GUI.

No diagrama de classes da figura 5.38 podemos ver a as classes necessários para a realização deste caso de uso, de destacar a utilização de uma superclasse *filamentMonitor* com os quatro monitores como subclasses. Os vários monitores apenas irão ser diferentes no critério de paragem (*stop_param*) e nos pedidos feitos à *printer_api* (*main_request* e *finish_request*). Podemos ainda verificar neste diagrama de classes que é necessário desenvolver mais quatro métodos da API do servidor de impressão BEE2B:

- */unload_filament* (GET)
- */identify_filament* (GET)

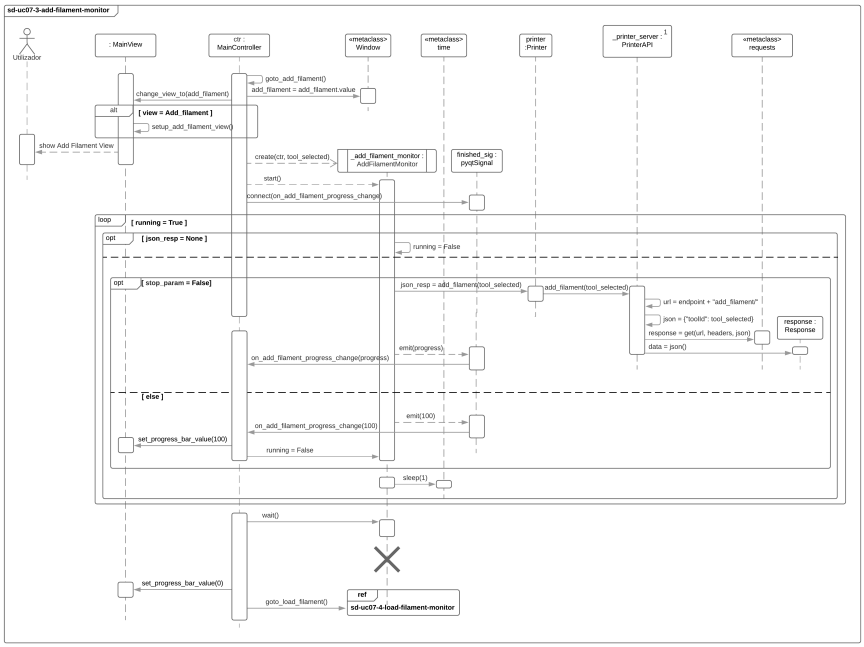


FIGURA 5.35: Diagrama de Sequência UC07: Monitor de Adição Filamento.

- */add_filament* (GET)
- */load_filament* (GET)

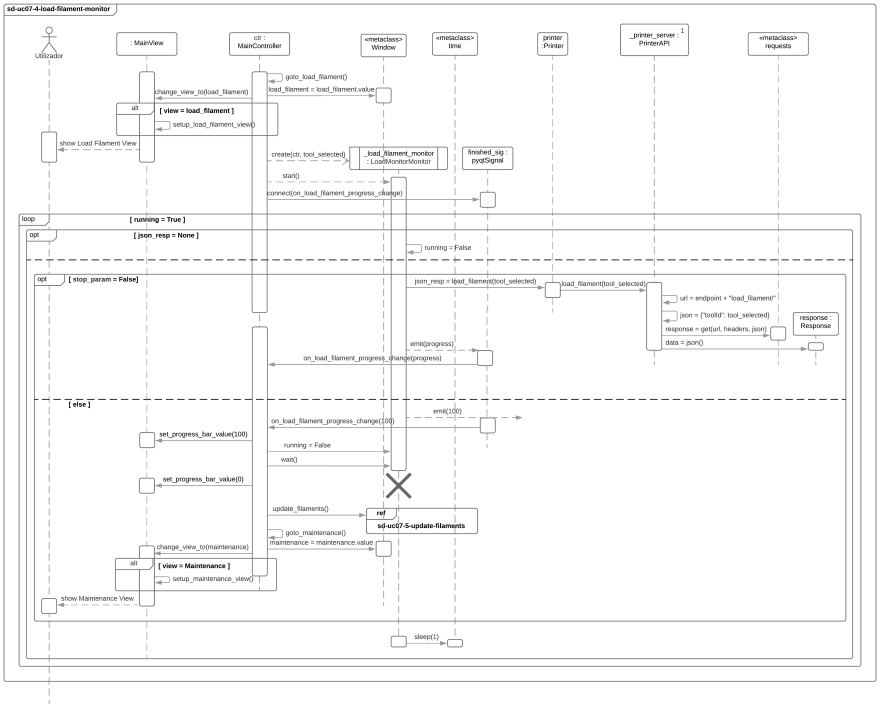


FIGURA 5.36: Diagrama de Sequência UC07: Monitor de Carregamento de Filamento.

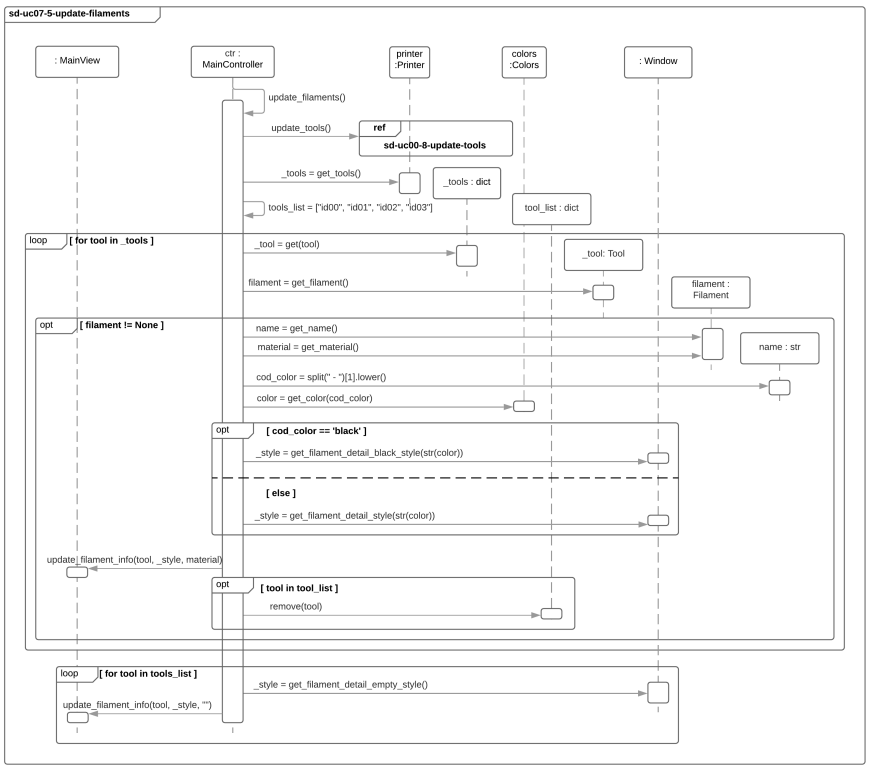


FIGURA 5.37: Diagrama de Sequência UC07: Atualizar Filamentos.

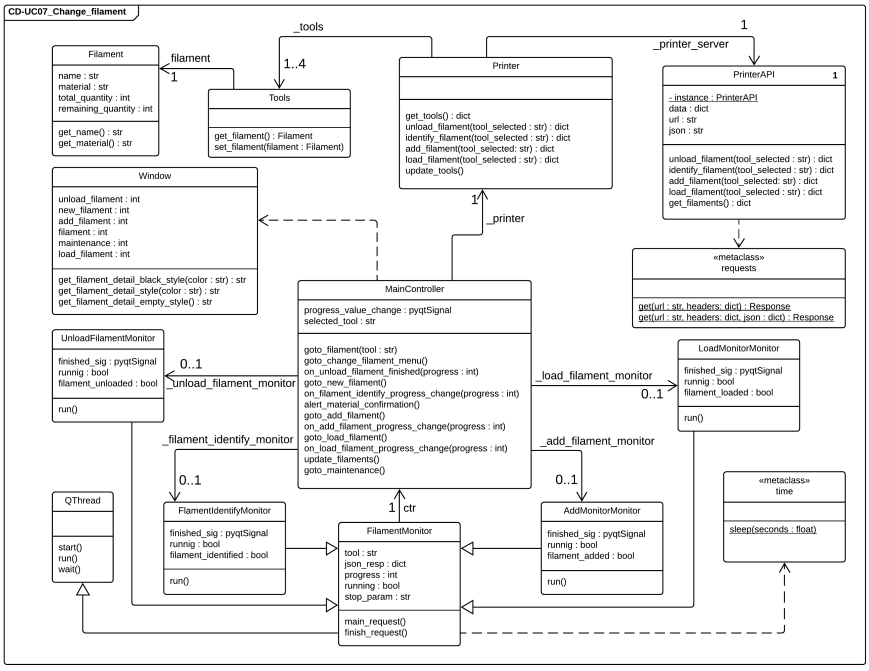


FIGURA 5.38: Diagrama parcial de Classes UC07: Trocar Filamento.

5.3.6 UCR08: Trocar Extrusor

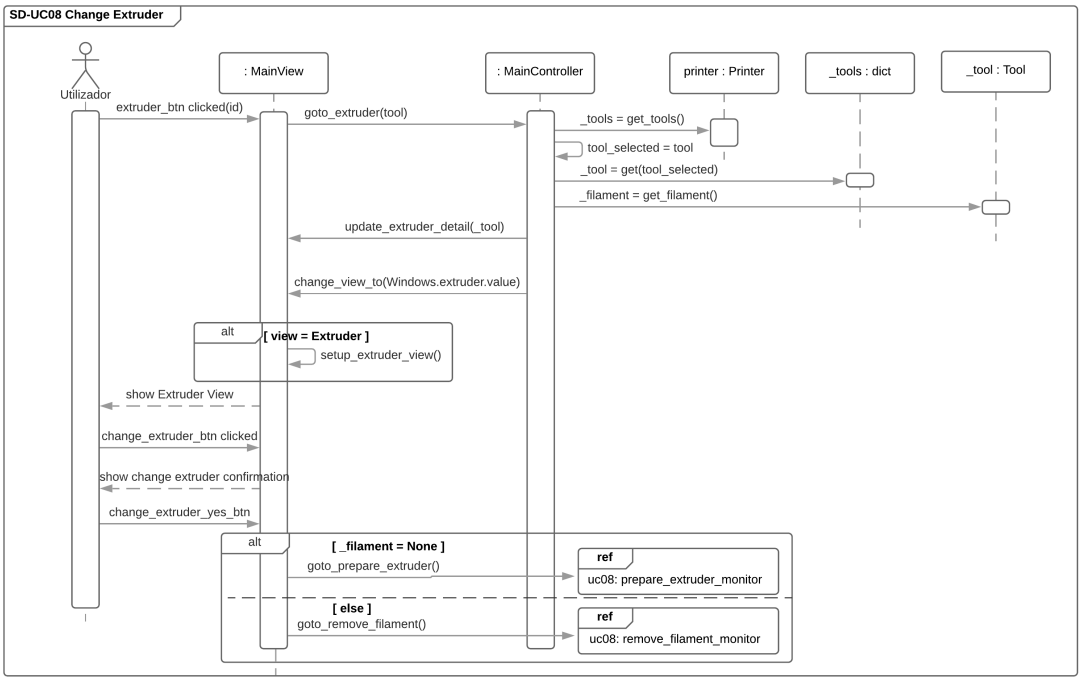


FIGURA 5.39: Diagrama de Sequência UC08: Trocar Extrusor.

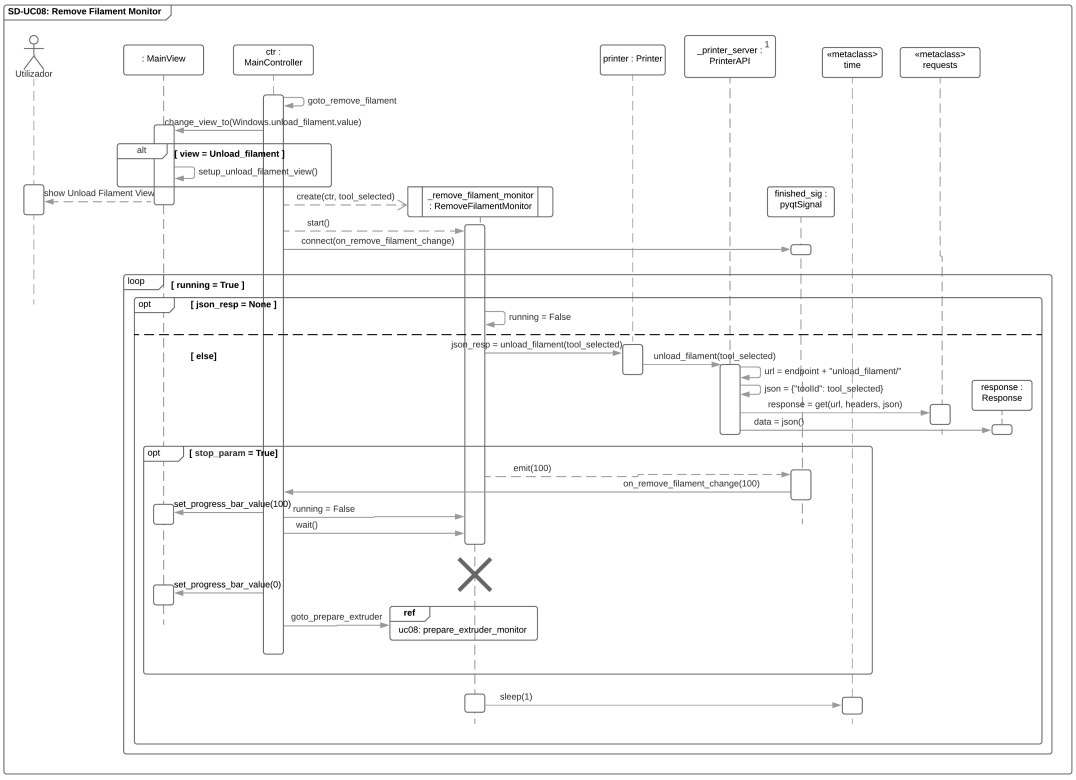


FIGURA 5.40: Diagrama de Sequência UC08: Monitor de Descarregamento de Filamento.

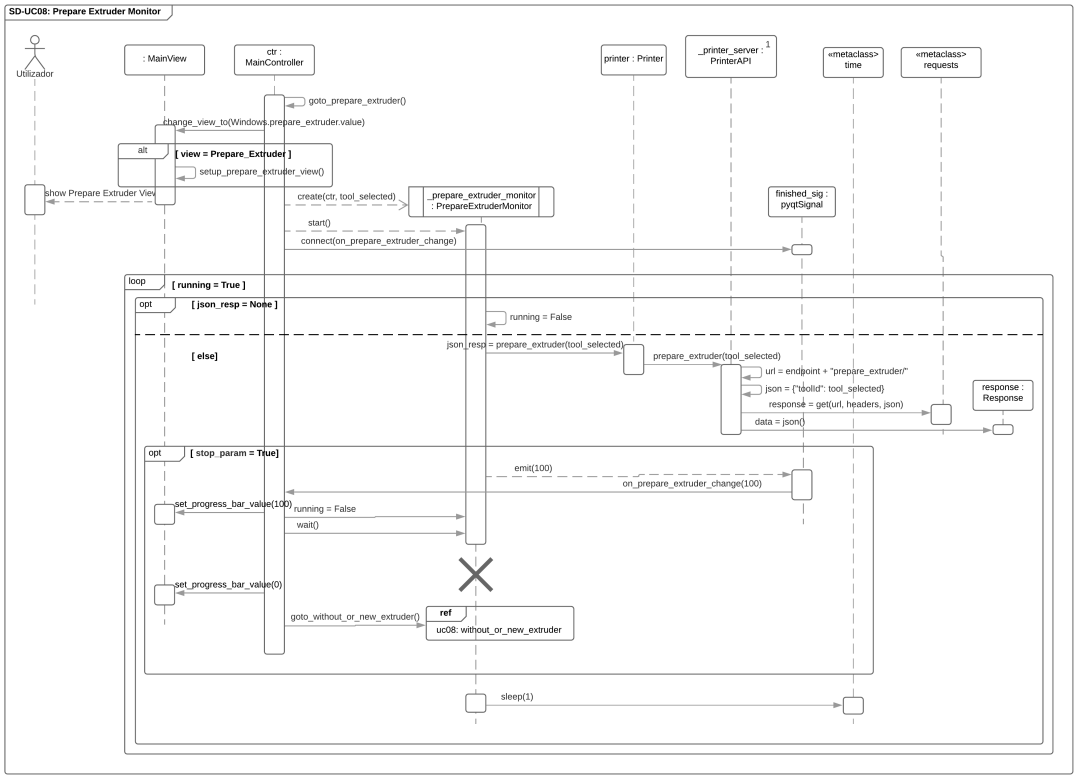


FIGURA 5.41: Diagrama de Sequência UC08: Monitor de Preparação de Extrudor.

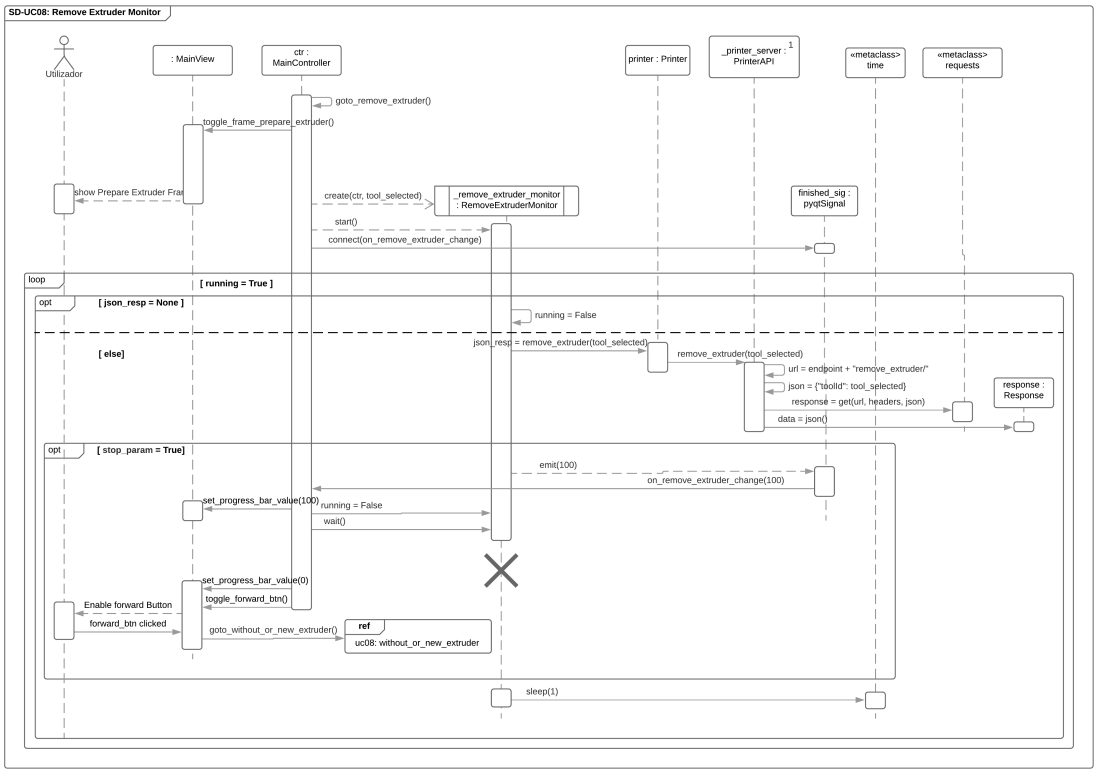


FIGURA 5.42: Diagrama de Sequência UC08: Monitor de Remoção de Extrudor.

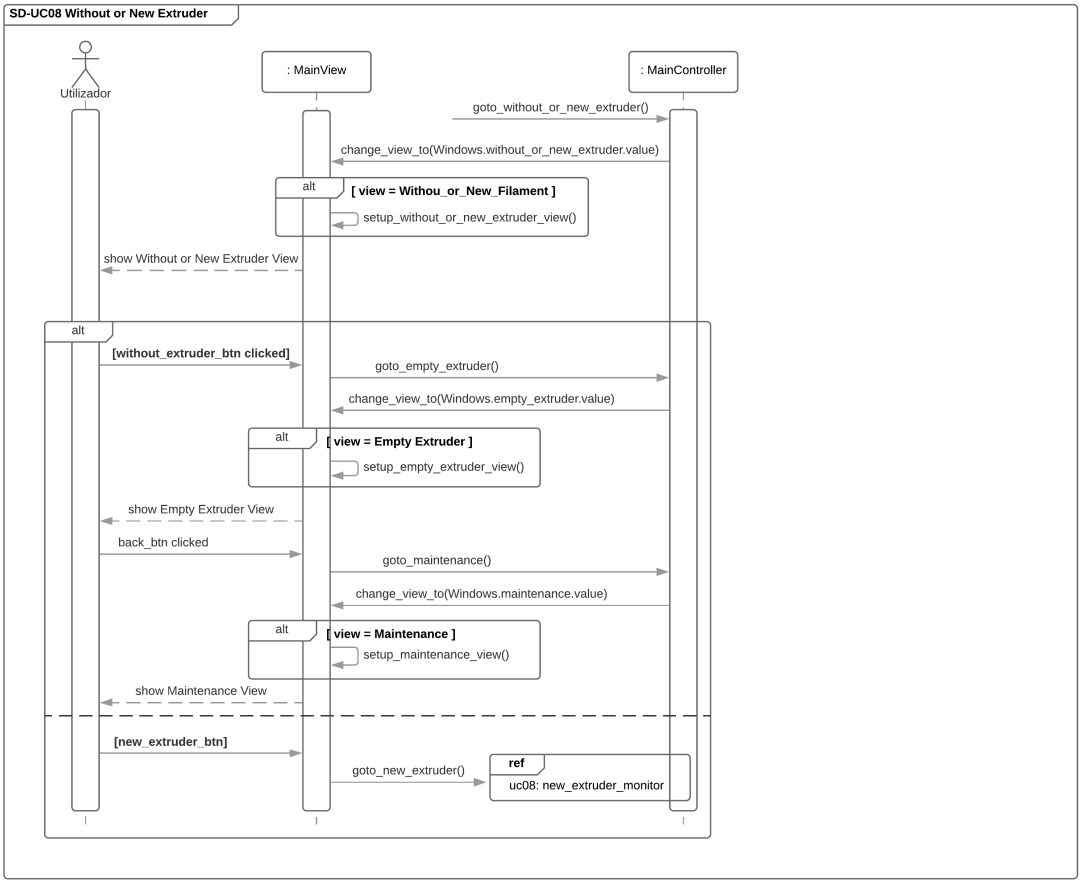


FIGURA 5.43: Diagrama de Sequência UC08: Novo Extrusor ou sem Extrusor.

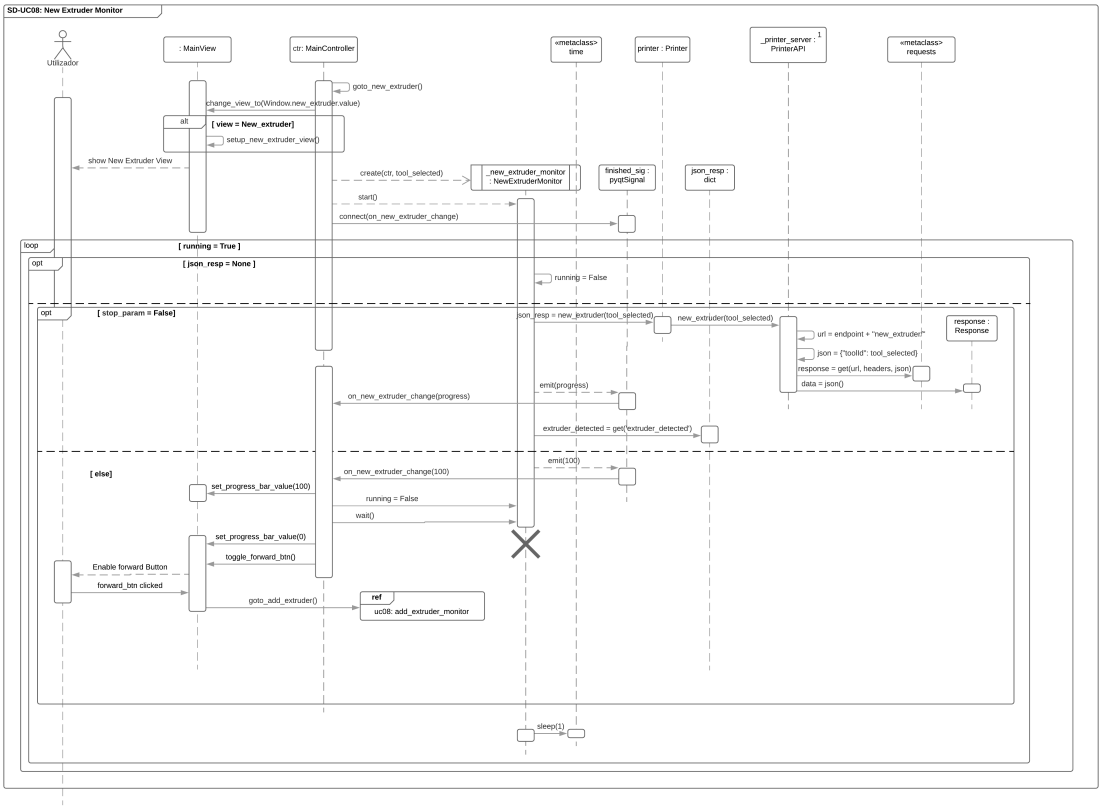


FIGURA 5.44: Diagrama de Sequência UC08: Monitor de Novo Extrusor.

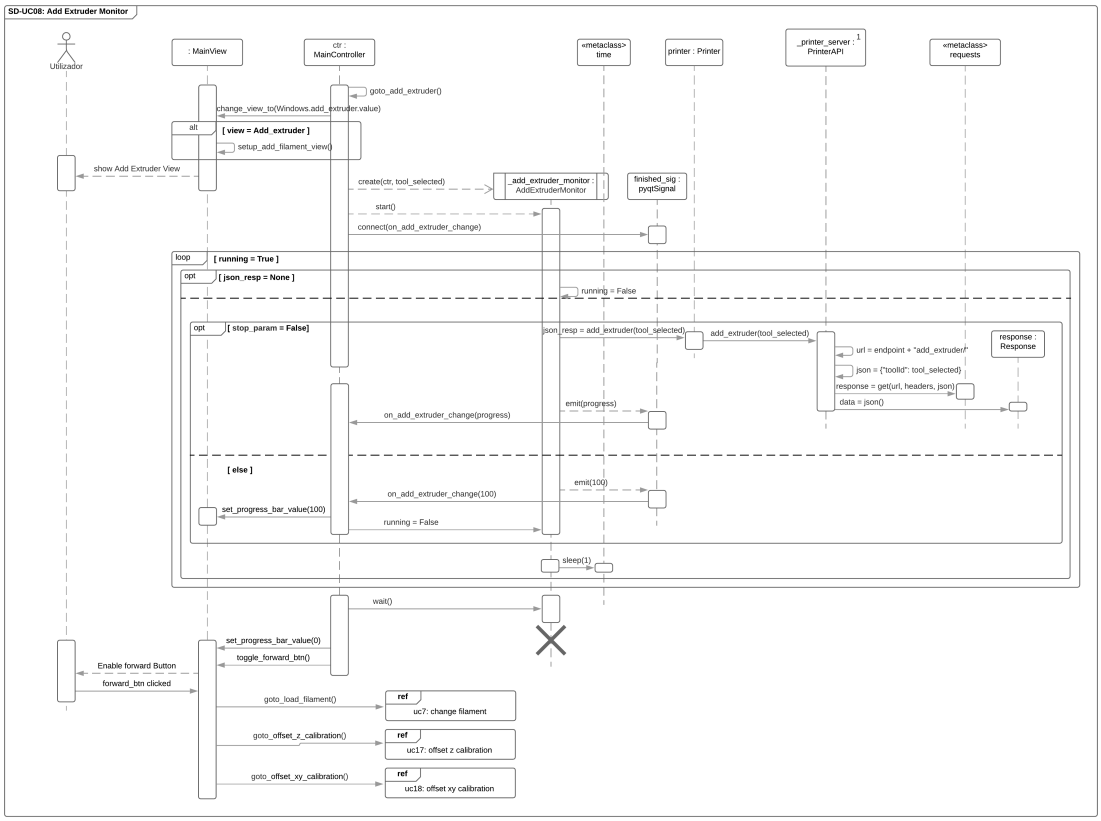


FIGURA 5.45: Diagrama de Sequência UC08: Monitor de Adição de Extrusor.

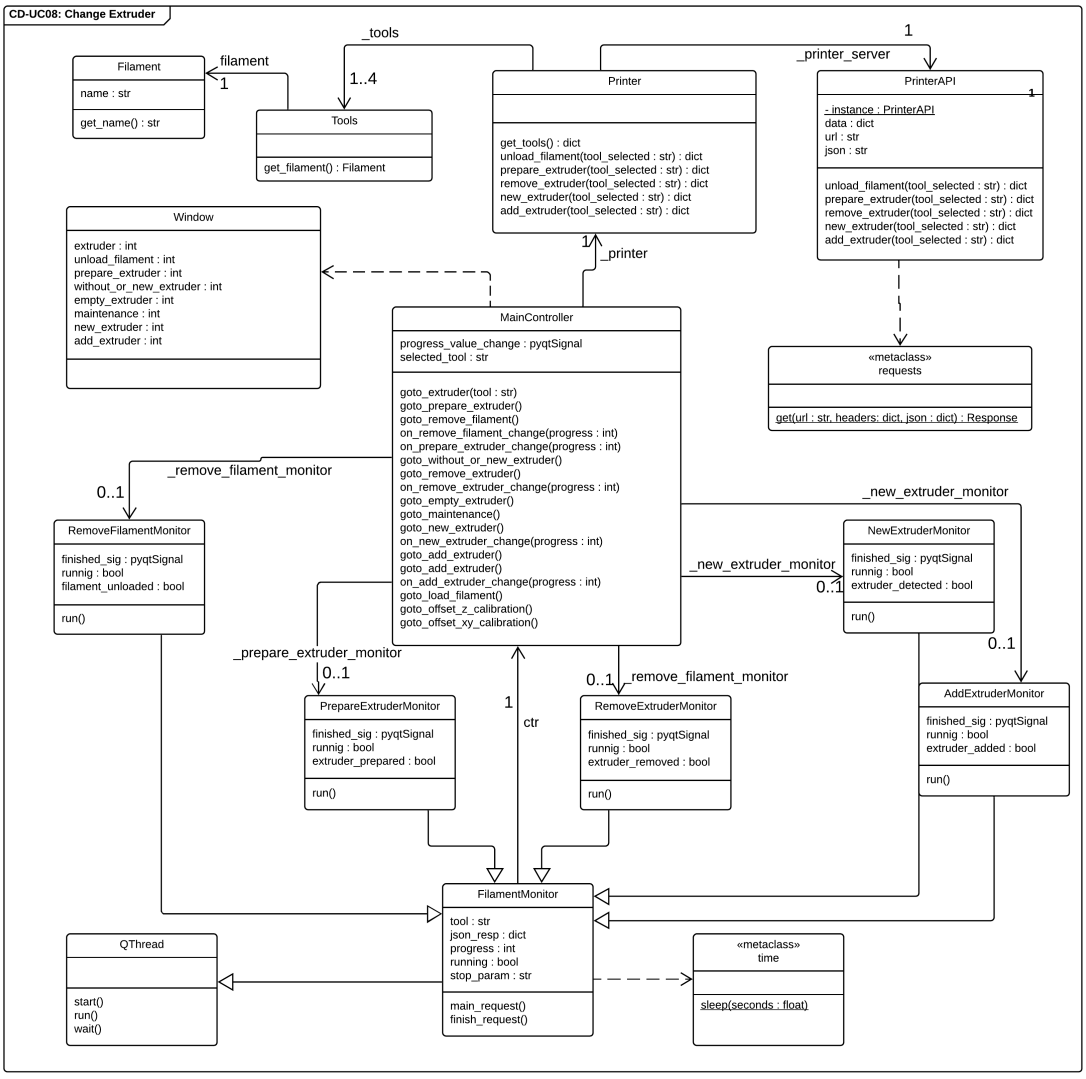


FIGURA 5.46: Diagrama parcial de Classes UC08: Trocar Extrusor.

5.4 BEECura

5.4.1 Criação de um *plugin* Cura

Nos manuais do cura, sugere-se que a criação de um *plugin* seja feita por edição de um já existente, havendo os 4 *templates* referidos em `https://github.com/ultimaker/cura/wiki/plugin-directory`. Contudo, estes exemplos têm várias limitações pois apenas oferecem ferramentas específicas.

Durante a criação do *plugin* BEECura foram elaborados três *plugins* que vão servir de modelo para futuros desenvolvimentos:

- BEE *tool plugin*
- BEE *extension plugin*
- BEE *curastage plugin*

Desta primeira iteração na criação de *plugins* para o Cura verificou-se que os seguintes componentes são obrigatórios para todos os *pluggins* do Cura:

- plugin.json
- ___init___.py
- e pode ainda ter ficheiros adicionais opcionais como *.py, *.qml e *.svg.

5.4.1.1 BEE *tool plugin*

O BEE *tool plugin* , figura 5.47, foi editado de modo a incluir uma ferramenta BEE *tool* no menu do lado esquerdo quando está selecionado um modelo, esta ferramenta apenas mostra um popup com o conteúdo: "/! BEE:<nome-do-ficheiro-seleccionado>".

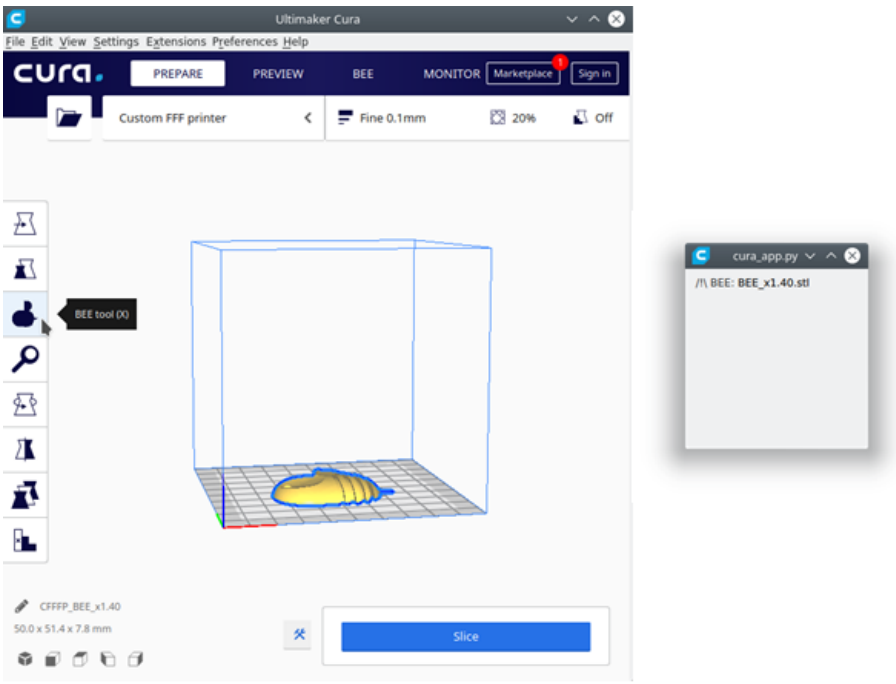


FIGURA 5.47: BEE *tool plugin*.

5.4.1.2 BEE *extension plugin*

O BEE *extension plugin* permite-nos correr qualquer código genérico de python, este *plugin* foi editado de modo a obter o estado da impressora com sucesso.

A vantagem deste *plugin* é o seu carácter genérico, e a desvantagem é que é preciso ir ao menu das *extensions* do cura para o ver em ação.

Este *pluggin* é constituído pelos seguintes ficheiros:

- __init__.py : inicializa o *plugin*;
- plugin.json : dados gerais do *plugin* como versão e título;
- BEE_extension.py : ficheiro principal que coordena o *plugin*;
- hello.qml : definição de um janela de *popup* com uma mensagem básica;
- printer_api.py : classe que contém um elemento que comunica com a API do servidor;
- model_bee2b.py : classe que contem uma thread que recorre à printer_api para obter informação do servidor;
- services : Pasta que contém ficheiros adicionais (obtidos com base no projecto de GUI BEE2B) e os respectivos subficheiros;
- BEE_window.qml : janela principal do *popup* do BEE *extension plugin*: constituída por *header*, *main* e *footer*;
- BEE_maincontent.qml : núcleo do conteúdo incluído na janela principal;
- BEE_service.py : ficheiro onde se definem as funções de interação entre a interface em qml e o código em python;

De seguida vamos descrever alguns dos ficheiros mais importantes:

__init__.py

Faz o import do *plugin*, e regista-o no cura:

```
def register(app):
    return {"extension": BEE_extension.BEE_extension()}
```

Listing 5.1: BEE *extension plugin* - __init__.py

plugin.json

Contem a informação com o título do *plugin*, versão e descrição:

```
{
    "name": "BEE_extension_plugin",
    "author": "BEEVERYCREATIVE",
    "version": "1.0",
    "description": "... ",
    "api": "6.0",
    "i18n-catalog": "... "
}
```

Listing 5.2: BEE *extension plugin* - plugin.json

BEE_extension.py

O ficheiro BEE_extension.py importa a printer_api e o model_bee2b, e inicializa o *plugin*. Foram criadas duas funções, para associar cada uma delas a um dos sub-menus:

```
1
2       #### BEE submenus ####
3       ##  Here, we define a function that will be called for
        each submenu:
4       ##  say_hello() creates a small dialogue window that says
         hello to the user
5       def say_hello(self):
6           self.i+=1
7           if not self.hello_window: #don't create more than one
        .
8               self.hello_window = self._create_dialogue("hello.
        qml")
9           print("self.i:", self.i)
10          self.hello_window.show()
11
12
13      ##  get_printer_state() calls the printer status
14      def get_printer_state(self):
15          if self.model is None:
16              from . import model_bee2b
17              self.model = model_bee2b.Model_BEE2B()
18
19          result = self.model.get_printer_state()
20          print("PRINTER STATUS:", result)
21          return "PRINTER STATUS: "+result
```

Listing 5.3: BEE *extension plugin* - BEE_extension.py

hello.qml

O ficheiro hello.qml contem a informação sobre a janela gráfica criada aquando da chamada do primeiro submenu.

BEE_window.qml

Neste ficheiro define-se a janela principal, define-se um cabeçalho, importa-se o BEE_maincontent.qml, e define-se um rodapé.

BEE_maincontent.qml

Este é o ficheiro onde se define todo a aspeto gráfico.

ligação à impressora na linha de comandos

Para utilizar o PrintServer API é necessário inicialmente obter um *token* de autenticação, para testes usou-se a ferramenta Postman. Com esta ferramenta foi feito um pedido ao servidor por HTTP ao seguinte endereço `http://127.0.0.1:8000/api/request_auth` e enviou-se com o método POST um json com o client_id, como por exemplo:

```
1 {
2     "client_id": "BEE-client"
3 }
```

Listing 5.4: json enviado ao método request_auth

De seguida, recebe-se a resposta, que irá incluir um *token*, como por exemplo:

```
{
    "token": "1cbe81075a05908dd8baa9534a90e967a04ed1f9"
}
```

LISTING 5.5: resposta parcial do método request_auth

O BEECura tem de usar um *token* válido, tal como este acima mencionado. A definição do *token* de autenticação faz-se no ficheiro printer_api.py Na figura 5.48 observa-se à esquerda a janela do cura, à direita em cima o terminal do servidor, e à direita em baixo o terminal do cura que inclui o cliente que se liga com sucesso ao servidor. O estado printer.connected é retornado pelo servidor que está a trabalhar com uma impressora virtual, e é recebido pelo cliente do cura.

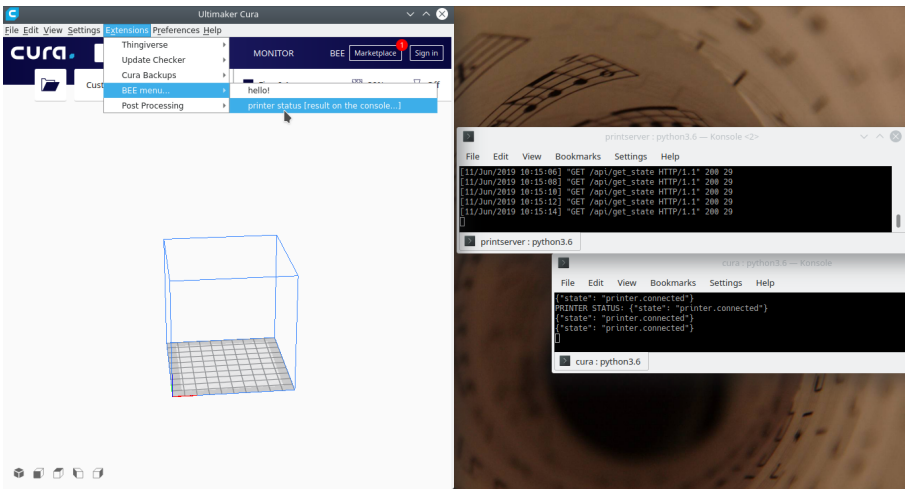


FIGURA 5.48: BEE *extension plugin*.

Interação entre a linha de comandos e a GUI

Foi ainda criado um terceiro sub-menu para este *plugin* que abre um *popup* (figura 5.49) com a informação da impressora. Usou-se este *popup* para testar três cenários:

1. a chamada de uma função genérica quando o primeiro botão **go!** é clicado;
2. a chamada da função da impressora BEE2B que faz ligação ao o servidor, e retorna o resultado para a interface gráfica;
3. a manipulação de um *input* genérico do utilizador dado em qml, processado em python, e retorno do resultado para o qml.

No primeiro cenário testou-se a chamada de uma função python ao clicar num botão qml e o retorno para o qml, usando neste caso uma função genérica. De um modo semelhante, no segundo caso, substituiu-se a chamada a essa função pelo código respetivo da chamada get_state. Com isto testámos as respostas a eventos e a passagem de dados de python para qml. De seguida, desenvolveu-se um código que permite passar informação em ambas as direções: o utilizador insere uma string numa textfield de qml, a qual é enviada para o python, é processada, neste caso apenas se faz um uppercase(), e o resultado é enviado para o qml, testando assim a transmissão de dados bidirecional.

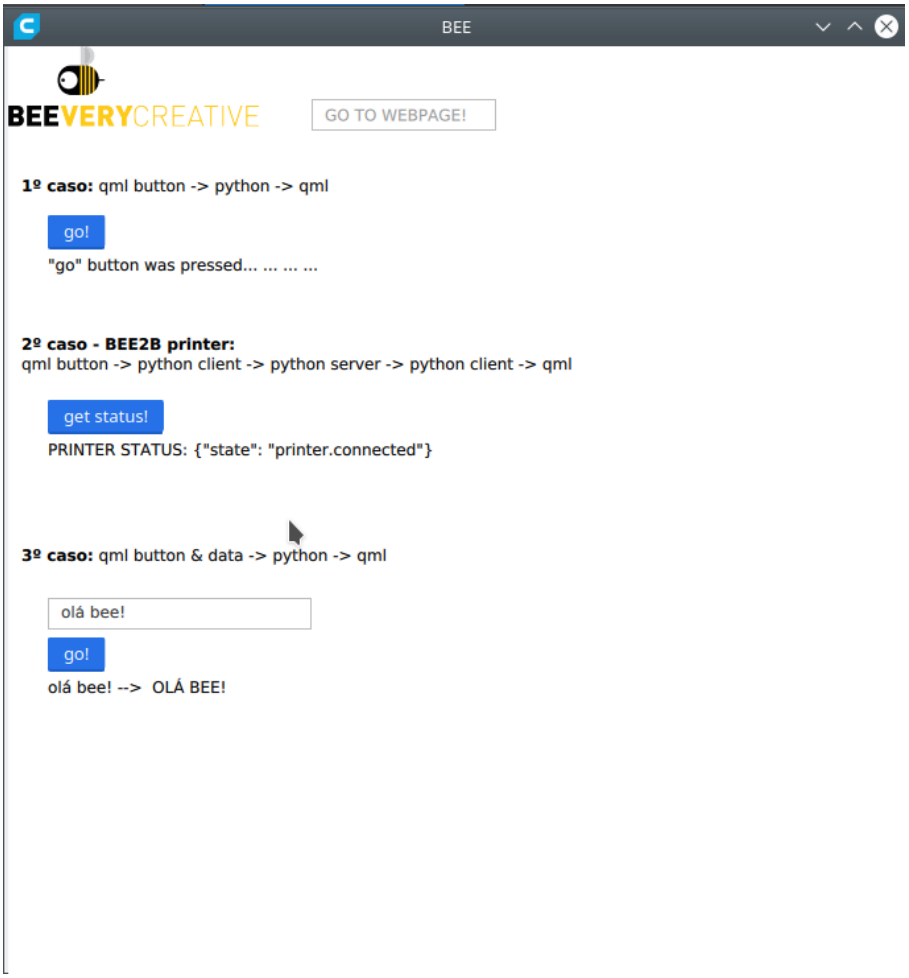


FIGURA 5.49: BEE *tool plugin - popup*.

5.4.1.3 BEE *curastage plugin*

Neste último *plugin*, figura 5.50, tentou-se criar uma ligação entre os comandos corridos na consola do cura e na interface gráfica, este *plugin* foi inspirado num dos *plugins* intrínsecos dos cura: o MonitorStage, presente na pasta de *plugins* criada pela instalação do cura.

5.4.2 Cena 3D - Fizheiro Zip

Inicialmente estava previsto a *rendering* 3D da cena a imprimir na GUI BEE2B, mas depois de criar os primeiros zips com o GCode e modelos STL observou-se que os ficheiros zips eram demasiado grandes para o envio por rede, principalmente no caso da ligação Wi-Fi. No entanto, este problema não é muito grave porque como a impressora 3D têm de aquecer para a impressão, pode fazer as duas tarefas em simultâneo, minimizando assim o tempo de espera.

Outro problema verificado foi no próprio *rendering* da cena 3D, com cenas com muitos objetos a *Frames per Second* (FPS) da GUI baixa demasiado tornando-a pouco fluída, assim, para minimizar estas situações optou-se por uma representação 2D da cena 3D a imprimir vista de topo. Com esta solução permite-se na mesma a visão da cena onde o utilizador pode selecionar um objeto e removê-lo da impressão, não comprometendo assim um dos objetivos deste projeto.

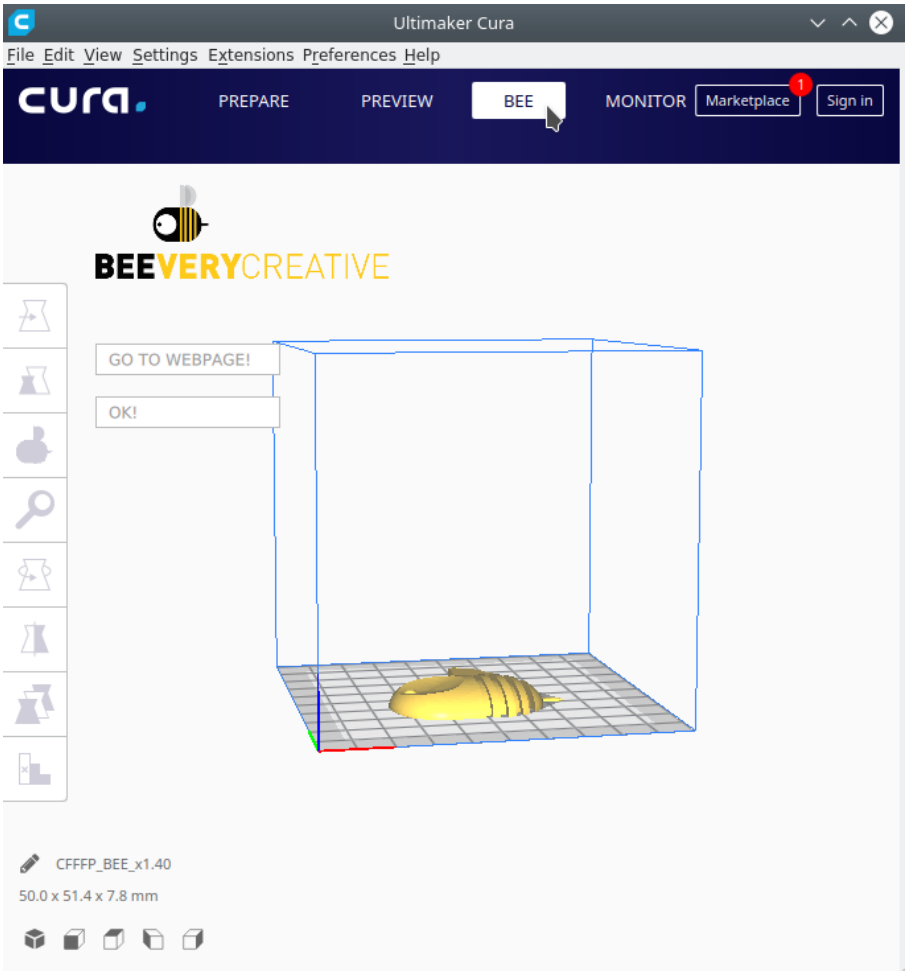


FIGURA 5.50: BEE *curastage plugin*.

Para esta representação 2D é necessário que o zip gerado pelo cura inclúa os seguintes ficheiros:

- gcode alterado com os ids dos objetos da cena 3D;
- uma imagem da cena 3D para mostrar na escolha dos ficheiros;
- uma imagem vista de topo de cada modelo da cena 3D para a construção da cena 2D;
- uma imagem vista de topo de cada modelo da cena 3D de cor diferente da anterior para na construção da cena 2D mostrar como objeto selecionado;
- json com as seguintes informações:
  - nome da impressão;
  - tipos de filamento utilizados;
  - quantidades de filamento utilizadas;
  - tempo previsto de impressão;
  - para cada objeto na cena: nome do objeto, id, modelo, pontos x e y do centro do objeto, largura e altura do objeto.

Todos os zips gerados pelo BEECura tem terminação [nome da impressão]_BEE2B.zip.

5.4.3 Manipulação do G-Code

Para alterar o gcode de modo a que seja possível remover um objeto da impressão criou-se um script em python que para cada linha do gcode do tipo:

```
;MESH:eiffeltower\_fixed.stl
```

é substituída por:

```
;MESH:eiffeltower\_fixed.stl (s\_i)
Mxx s\_i
```

obtendo um gcode modificado. Para testar esta novo gcode filtrou-se os comandos entre "Mxx s_i" e "Mxx <seguinte>", tal que i era o valor do id de um objeto que o utilizador escolheu remover, comentou-se as linhas indesejadas e ficou-se com um novo gcode.

Durante a pre visualização deste gcode no Cura, observou-se que, apesar de os objetos em x e y estarem a ser corretamente impressos, por vezes, em determinadas situações os objetos não cresciam. Depois de voltar a analisar o gcode original, verificou-se que o problema estava a ser causado porque as alterações em Z estavam a ser feitas dentro das MESHs, assim, sempre que um objeto removido tivesse uma instrução do tipo G0 ... ... Z... ia também ignorar a mudança em Z, causando assim os resultados verificados anteriormente.

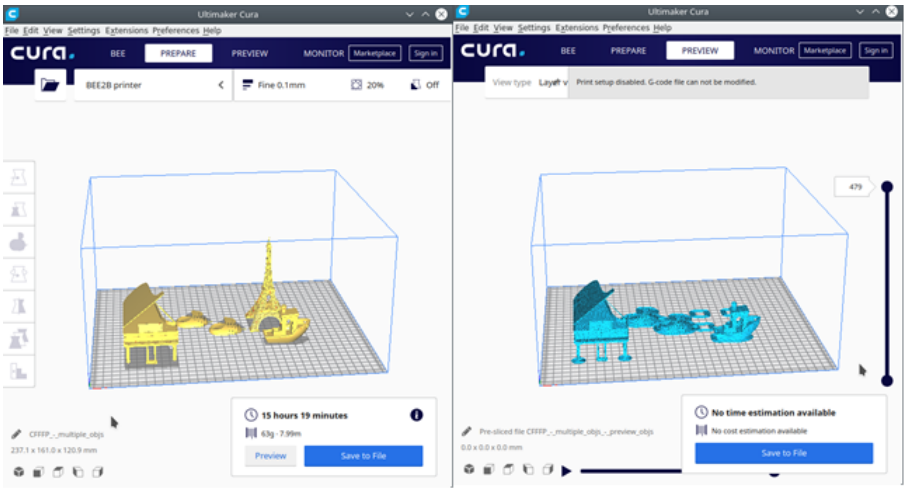


FIGURA 5.51: exemplo no caso de remover o objeto torre eiffel, de uma cena com vários objetos: esquerda) cena original; direita) cena importada do gcode modificado.

Para corrigir o problema com os comentários no local ideal para a preparação da remoção de um objeto, fez-se o seguinte tipo de alteração:

de:

```
;MESH:eiffeltower\_fixed.stl
;LAYER:153
;MESH:eiffeltower\_fixed.stl
G0 X168.771 Y143.771 Z15.6
```

para:

```
;;; objectivo
;LAYER:153
;MESH:nonmesh
G0 Z15.6
;MESH:eiffeltower\_fixed.stl (... )
Mxxx Iii L<linha c/ ;MESH seguinte>
G0 X168.771 Y143.771
```

Capítulo 6

Navegação

Neste capítulo são apresentados esboços do layout da GUI BEE2B e BEECura. A evolução do layout da Interface gráfica é apresentada na secção 6.1, de seguida são apresentados os seus principais menus na secção 6.2 e por fim os vários casos de uso, durante alguns casos de uso também são mostrados esboços do layout do BEECura.

6.1 Evolução da Interface Gráfica

Foram realizadas sessões, onde se utilizaram ferramentas e métodos como, *brainstorms* e *design thinkings* para projetar o *layout* da GUI BEE2B. A equipa destas sessões foi multidisciplinar e contou com utilizadores comuns de impressoras 3D, especialistas em design (Grandesign), elementos das equipas de software, firmware, mecânica e marketing da BEEVERYCREATIVE. Este processo foi uma vez mais iterativo.

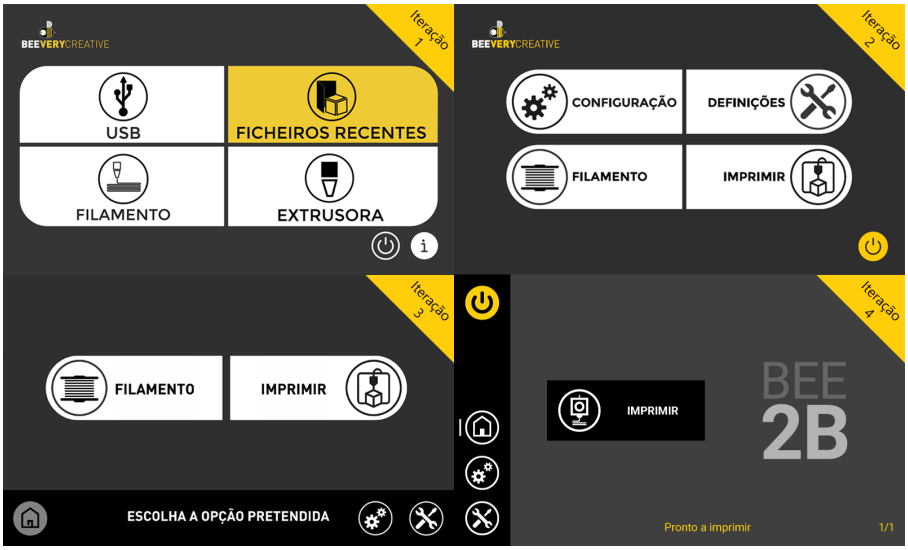


FIGURA 6.1: Evolução do Layout da Interface Gráfica BEE2B durante as várias iterações.

Na figura 6.1 podemos ver a evolução da vista *home* ao longo das quatro iterações realizadas. Nas duas primeiras iterações foram analisados os caso de uso relativos à impressão (a partir de USB e Recentes). As principais alterações que emergiram da primeira para a segunda iteração foram:

- menus USB e Ficheiros Recentes uniram-se formando o botão Imprimir, o seu sub-menu passa a apresentar as opções USB e Recentes em separadores;

- à medida que os restantes requisitos foram sendo refinados surgiram dois novos menus, Configuração e definições;
- o menu Extrusor é removido mas é mantido o menu Filamento para dar ao utilizador um acesso rápido a uma das principais e mais utilizadas tarefas de manutenção, trocar de filamento.

Nas últimas duas iterações foram analisados os restantes casos de uso relacionados com a manutenção de ferramentas e definições da aplicação. Durante a análise destes casos de uso surgiram as alterações da terceira iteração:

- nova barra de menus (barra preta no fundo da aplicação), nesta barra ficam disponíveis os menus *home*, definições e manutenção. Estes menus estão disponíveis praticamente durante toda a utilização permitindo ao utilizador aceder a estes menus de uma forma mais rápida;
- menu configuração muda de nome para manutenção;
- nova zona de informação (título da vista e informação sobre o estado da impressora) no centro da nova barra de menus.

Na última iteração foram refinados mais alguns aspetos da interface gráfica como:

- a barra de menus muda de posição, passa agora a ocupar o canto esquerdo da aplicação fornecendo assim mais espaço em altura para as restantes vistas;
- A zona de informação divide-se, o título passa a ser apresentado no topo da aplicação enquanto a informação sobre o estado é apresentada na mesma posição mas agora com mais espaço mostrando ainda o número de mensagens atuais;
- menu filamento é removido, depois de refinado o menu de manutenção concluiu-se que o menu filamento não acrescentava valor, visto o menu manutenção já apresentar a mesma informação;
- o menu atual é agora indicado por uma barra na vertical situada à esquerda do botão correspondente;
- são adicionadas notificações importantes como: novas atualizações ou aviso de nível baixo de filamento.

No final das várias iterações chegou-se ao layout da figura 6.2 que será utilizado em toda a aplicação. Este layout define claramente cada zona da GUI BEE2B.

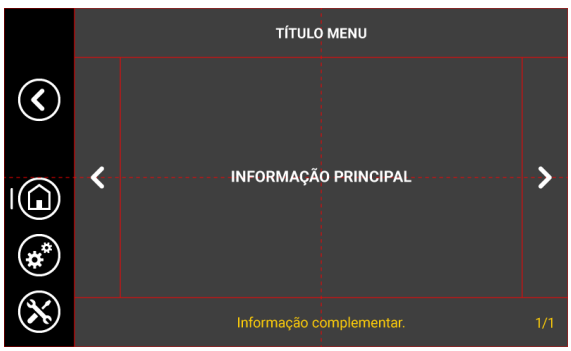


FIGURA 6.2: Layout da GUI BEE2B.

- **barra de menus:** situada no lado esquerdo da aplicação onde estão presentes os botões **desligar**(apenas presente na vista *home* e quando não está em processo de impressão, nesse caso irá aparecer o botão ***stop*** durante os processos da impressora), ***home***, **definições** e **manutenção**. Nesta barra estará ainda presente, quando necessário, um botão ***back*** e uma barra de progresso;
- **zona de informação complementar:** situada no fundo da aplicação apresenta mensagens relativamente a eventos e estado da impressora. Pode existir mais do que uma mensagem, a indicação do canto inferior direito mostra a mensagem atual e o número de mensagens totais, o utilizador pode alternar entre elas clicando na mensagem;
- **título de menu:** título do menu atual situado no topo da aplicação;
- **zona de informação principal:** no centro da aplicação fica a zona de conteúdo exclusivo de cada vista;
- **setas de navegação:** quando necessário as setas de navegação são apresentadas nas margens do conteúdo da vista, como no menu de escolha de ficheiro ou manutenção.

6.2 Principais Menus

Nesta secção vamos analisar com mais detalhe os principais menus da GUI BEE2B, começamos pela **página principal (*home*)** na secção 6.2.1, seguido do menu **definições** na secção 6.2.2 e terminamos com o menu da **manutenção** na secção 6.2.3.

6.2.1 Home

FIGURA 6.3: GUI BEE2B - Vista *Home*, pronto a imprimir.

A vista *home* alterna entre um e dois botões conforme o estado da impressora. Se a impressora não estiver a realizar nenhuma impressão apenas temos o botão **imprimir** disponível (figura 6.3), aqui iniciamos o processo de impressão tanto por uma drive USB (caso de uso UC02, secção 6.4) como de um ficheiro recente (caso de uso UC03, secção 6.5). Durante uma impressão o botão **imprimir** é substituído pelos botões **estado de impressão** e **adicionar impressão** (figura 6.4), e mostrada uma barra de progresso com o estado do processo atual (aquecimento, a imprimir ou arrefecimento) e respetivo ***stop*** (exceto em caso de arrefecimento).

Se o utilizador clicar em **estado de impressão** tem acesso à vista correspondente ao estado da impressão, por exemplo, se a impressora estiver a imprimir é mostrada a vista A imprimir, como o da figura 6.20 do caso de uso UC04: Impressão de cena 3D.

FIGURA 6.4: GUI BEE2B - Vista *Home*, a imprimir.

Com o botão **adicionar impressão**, o utilizador pode adicionar um ficheiro à fila de impressão. A GUI mostra a vista Selecção de Ficheiro como na figura 6.13, onde o utilizador pode adicionar um ficheiro USB ou Recente à fila de impressão. Nesta vista continuará presente a barra de progresso e respetivo ***stop*** para que o utilizador consiga acompanhar ou cancelar o processo de impressão. O botão **imprimir** passa agora a ser **adicionar**.

Se por alguma razão a GUI não estiver a comunicar com o servidor de impressão o botão **imprimir** fica desativo e a zona de informação complementar avisa o utilizador que o sistema está a inicializar como na figura 6.5.

FIGURA 6.5: GUI BEE2B - Vista *Home*, a inicializar sistema.

6.2.2 Definições

FIGURA 6.6: Vista Definições.

Na vista Definições o utilizador tem acesso aos botões: **configurar rede**, **emparelhar com PC**, **atualizar software**, **restaurar software** e **selecionar idioma**.

O botão **configurar rede** têm sempre uma indicação a mostrar se existe ou não alguma rede configurada na impressora. Na figura 6.7, podemos ver as várias notificações e indicações utilizadas ao longo da GUI. Este botão permite configurar uma nova rede ou mostrar a informação da rede atual (caso de uso UC11, secção 6.11).

FIGURA 6.7: Vista Definições.

Para o utilizador ter acesso ao botão **emparelhar com PC** é necessário ter uma rede configurada na impressora e, tal como no botão **configurar rede**, é apresentado uma notificação relativa ao estado do emparelhamento da impressora. Este botão permite emparelhar um novo PC (caso de uso UC14, secção 6.14) ou mostrar informação relativa aos PCs atualmente emparelhados com a impressora.

O botão **atualizar software** permite realizar o caso de uso UC13 (secção 6.13), que atualiza, não só a GUI, mas também o servidor de impressão. Este botão apresenta um ícone de notificação quando existem novas atualizações disponíveis.

O botão **restaurar software** serve para colocar o software no estado inicial (caso de uso UC12, secção 6.12), perdendo o histórico da impressora (exceto informação relativa à utilização da mesma como o número de horas de trabalho).

o utilizador pode ainda utilizar o botão **selecionar idioma** para alternar entre os idiomas disponíveis (caso de uso UC16, secção 6.15), neste momento, apenas estão disponíveis os idiomas português e inglês.

6.2.3 Manutenção

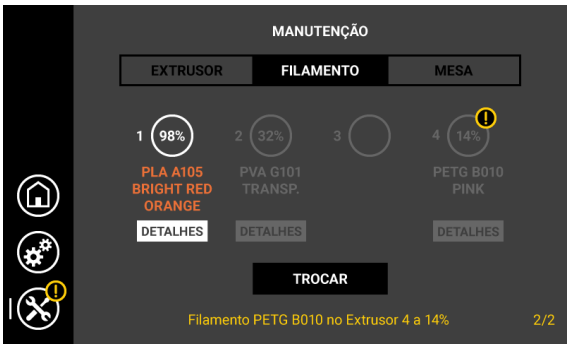


FIGURA 6.8: Menu Manutenção - Tab Filamento, filamento 1 selecionado.

A vista Manutenção apresenta todas as ferramentas da impressora, esta informação é separada em três separadores: Filamento, Extrusor e Mesa

No separador Filamento (figura 6.8), são mostrados os vários filamentos carregados nos respetivos extrusores, para cada filamento é mostrado a quantidade restante de material em

percentagem, o código completo do filamento (este código resume o tipo de material, código simples e cor do material) e um botão **detalhes** para aceder a informação mais detalhada do filamento selecionado. Para selecionar um novo filamento o utilizador apenas tem de clicar sobre ele. Para cada filamento é mostrado um botão **Trocar** (caso de uso UC07, secção 6.7) sempre que existe um filamento carregado ou **Carregar** se estiver vazio.

Quando algum filamento está com pouco material é apresentada uma notificação sobre o filamento correspondente e uma mensagem a informar o utilizador desse facto, esta mensagem só será removida com a substituição do filamento.

Apesar do código do filamento ser mostrada na cor do próprio filamento, a cor é também mostrada de forma textual para não impedir pessoas com daltonismo de a identificar.

FIGURA 6.9: Menu Manutenção - Tab Extrusor, extrusor 1 selecionado.

No separador Extrusor (figura 6.9) são mostrados os vários extrusores carregados nos respetivos *slots* da impressora 3D BEE2B, para cada extrusor é mostrado o tipo de material, o diâmetro do *nozzle* e um botão **detalhes** para aceder a informação mais detalhada do extrusor selecionado (figura 6.10). Para selecionar um novo extrusor o utilizador apenas tem de clicar sobre ele. No separador Extrusor é ainda mostrado dois botões: **Trocar** e **Calibrar XY**. Caso não exista nenhum extrusor carregado o botão Trocar muda para **Carregar** e o Calibrar XY fica inativo.

FIGURA 6.10: Menu Manutenção - Tab Extrusor, detalhe de extrusor 4.

Quando o utilizador clica em **detalhes** tem acesso a mais alguma informação do extrusor como: materiais compatíveis e último material como na figura 6.10. Nesta vista o utilizador pode alternar entre os extrusores com as setas das margens ou sair dos detalhes com o botão **anterior** da barra de menus.

O botão Trocar vai iniciar o caso de uso UC08, secção 6.8, e o botão Calibrar XY vai iniciar o caso de uso UC18, secção 6.17.

No separador Mesa, como podemos ver na figura 6.11, a GUI mostra a informação detalhada da mesa instalada na impressora . Aqui, o utilizador pode consultar a temperatura atual da mesa, o volume de impressão ou o seu tempo de utilização. O botão **calibrar Z** permite iniciar o caso de uso UC17: Calibrar Z, que podemos ver na secção 6.16

FIGURA 6.11: Menu Manutenção - Separador Mesa.

6.3 UC01: Iniciar impressão a partir do PC

Esta secção vai mostrar a navegação do BEECura e GUI BEE2B durante o caso de uso UC01: Iniciar impressão a partir do PC (requisitos descritos na secção A.3.1).

Este caso de uso é iniciado no BEECura, onde depois de preparada a cena 3D, configurada a rede da impressora e de ter o PC emparelhado, o utilizador pode usar o botão imprimir como na figura 6.12 para iniciar uma impressão 3D. O utilizador pode ainda ver o estado da ligação à impressora (informação visual no canto superior esquerdo) e fazer *Slice & Save* que lhe permite fazer o *slice* da cena 3D e guardar o zip resultante localmente ou numa drive USB, para posteriormente inserir na impressora 3D e iniciar uma impressão a partir de uma drive USB.

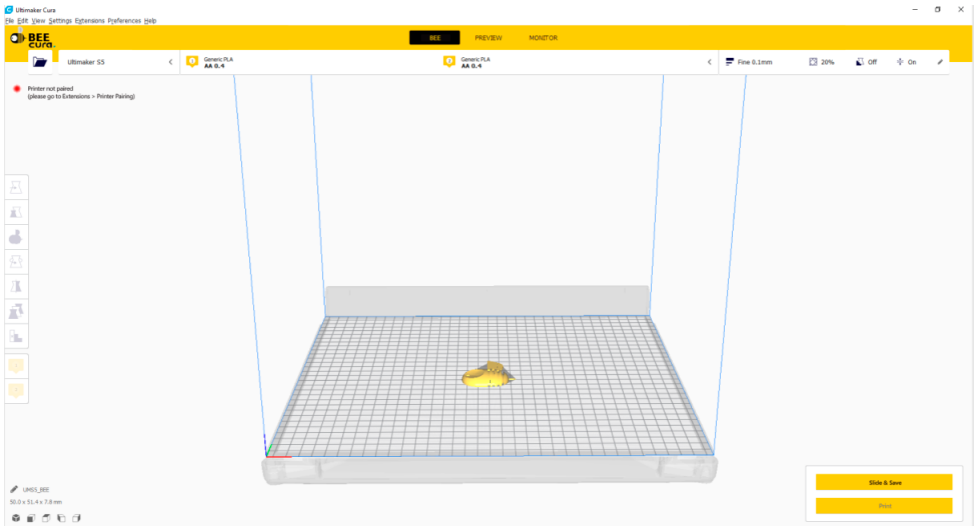


FIGURA 6.12: UC01: Imprimir a partir do PC - BEECura Vista Home.

Tal como descrito na secção 6.2.1 menu *Home*, Durante a impressão, o utilizador pode acompanhar a evolução da impressão na GUI e remover algum objecto da mesma, esta parte da navegação será descrita com mais detalhe na secção 6.6.

6.4 UC02: Iniciar impressão a partir de drive USB

Esta secção vai acompanhar a navegação do caso de uso UC02: Iniciar impressão a partir de drive USB (requisitos descritos na secção A.3.2).

Se o utilizador optar por fazer ***Slice & Save*** no caso de uso UC01 (descrito na secção 6.3), deve guardar o ficheiro zip resultante do *slice* numa drive USB e colocá-la na impressora 3D. A GUI deteta uma nova drive USB e respetivos ficheiros para impressão. Para iniciar este caso de uso o utilizador deve clicar no botão **imprimir** da vista *home* como na figura 6.3 seguindo para a vista de seleção de ficheiro a imprimir (figura 6.13).

FIGURA 6.13: UC02: Imprimir a partir de drive USB - Vista Seleção de Ficheiro, separador USB.

Na vista de seleção de ficheiro existem duas separadores, USB e Recentes. O separador que é mostrada por defeito caso seja detetada alguma drive USB na impressora é o separador USB, onde são mostrados todos os ficheiros disponíveis nessa drive para impressão, no caso da quantidade de ficheiros ser elevada é mostrado setas de navegação entre as várias páginas. Para ter acesso aos detalhes de um ficheiro o utilizador deve clicar na sua imagem passando assim para a vista de detalhe do ficheiro (figura 6.14).

FIGURA 6.14: UC02: Imprimir a partir de drive USB - Vista Detalhe de ficheiro.

Na vista de detalhe do ficheiro, o utilizador vê uma representação da cena 3D que será impressa (tal como durante a impressão, é possível remover objetos da impressão, neste caso antes de mandar imprimir, esta funcionalidade será descrita na secção 6.6), e informação detalhada do ficheiro selecionado com informação como: nome, data de *slice*, tempo estimado de impressão e utilizador (nome do PC emparelhado)(figura 6.14).

Para iniciar a impressão o utilizador deve clicar em **imprimir**, a impressora depois de validar os periféricos (UC09 - secção 6.9), inicia então o processo de aquecimento das ferramentas para poder iniciar a impressão, durante este processo é apresentada a vista de Aquecimento da Impressora (figura 6.15), nesta vista o utilizador tem acesso a uma barra de progresso indicando o estado atual do aquecimento e ainda um botão ***stop*** para cancelar o processo mediante confirmação (figura 6.16).

FIGURA 6.15: UC02: Imprimir a partir de drive USB - Vista Em Aquecimento.

FIGURA 6.16: UC02: Imprimir a partir de drive USB - Vista Em Aquecimento, Cancelar - confirmação.

Durante a impressão, o utilizador pode acompanhar a evolução da impressão na GUI e remover algum objeto da mesma, esta parte da navegação será descrita com mais detalhe na secção 6.6.

No final da impressão é iniciado o processo de arrefecimento da impressora 3D e a GUI mostra a vista correspondente (figura 6.15), que tal como a vista de Aquecimento também têm uma barra de progresso indicando o estado atual do processo de arrefecimento.

FIGURA 6.17: UC02: Imprimir a partir de drive USB - Em Arrefecimento.

No final deste processo a GUI mostra uma mensagem de impressão concluída ao utilizador (figura 6.18), este pode avançar com o botão **OK** ou esperar alguns segundos. Se existir alguma impressão na fila de impressão a impressora verifica novamente os periféricos (UC09 - secção 6.9) e inicia o processo de aquecimento da impressora como na figura 6.15 continuando a partir deste ponto.

FIGURA 6.18: UC02: Imprimir a partir de drive USB - Vista Impressão concluída.

6.5 UC03: Iniciar impressão de ficheiro recente.

Nesta secção será descrito o caso de uso UC03: Iniciar impressão de ficheiro recente (requisitos descritos na secção A.3.3).

Depois da impressora 3D imprimir o primeiro ficheiro é possível voltar a imprimir a mesma cena, as últimas impressões realizadas serão guardadas na impressora.

Este caso de uso é semelhante ao UC02: Imprimir a partir de uma drive USB (secção 6.4), apenas difere no local de armazenamento dos ficheiros mostrados. Para imprimir um ficheiro recente o utilizador deve clicar no botão **imprimir** da vista *home* como na figura 6.3 seguindo para a vista de seleção de ficheiro a imprimir mas desta vez deve selecionar o separador Recentes (figura 6.19). A partir deste ponto este caso de uso é igual ao UC02.

FIGURA 6.19: UC03: Iniciar impressão de ficheiro recente - Vista Seleção de ficheiro - separador Recentes.

6.6 UC04: Impressão de Cena 3D e UC06: Remover Objeto da Impressão.

Nesta secção vamos acompanhar a navegação dos casos de uso UC04: Impressão de Cena 3D e UC06: Remover Objeto da Impressão (requisitos descritos nas secções A.3.4 e A.3.6 respetivamente).

Quando o utilizador inicia uma impressão, a GUI mostra a vista A imprimir como na figura 6.20, esta vista mostra uma barra de progresso, uma representação 2D da cena 3D vista de topo (inicialmente era previsto a cena em 3D mas devido a limitações explicadas na secção 5.4.2 optou-se por esta representação em 2D), e alguma informação relativa à cena 3D em impressão como nome, data prevista de conclusão e periféricos em uso.

FIGURA 6.20: UC04: Impressão de Cena 3D - Vista A Imprimir.

Para remover um objeto da impressão atual, por exemplo, no caso do utilizador notar visualmente uma falha de impressão num dos objetos, o utilizador deve clicar no objeto que pretende remover da impressão, é então apresentada uma imagem do objeto em destaque e um botão **remover objeto** como na figura 6.21.

Quando o utilizador clica no botão **remover objeto** terá uma confirmação dessa ação como na figura 6.22, depois de confirmada a ação, a impressora continua normalmente a impressão ignorando todas as instruções do objeto removido. A representação 3D da cena 3D também refletirá esta alteração, removendo o objeto da cena.

Durante o processo de impressão o utilizador pode cancelar a impressão, para isso, deve clicar no botão **stop** e confirmar a intenção na vista seguinte (figura 6.23). Se o utilizador confirmar

FIGURA 6.21: UC06: Remover Objeto da Impressão - Vista A Imprimir, objeto selecionado.

FIGURA 6.22: UC06: Remover Objeto da Impressão - Vista A Imprimir, Remover objeto - confirmação.

o cancelamento da impressão, a impressora irá iniciar o processo de arrefecimento como o da figura 6.17, depois segue, a partir desta vista, o fluxo do use case UC02: Imprimir a partir de uma drive USB.

FIGURA 6.23: UC04: Impressão de Cena 3D - Vista A Imprimir, Cancelar impressão - confirmação.

Para o utilizador meter a impressão em pause, deve clicar em **pausar** e confirmar na vista seguinte (figura 6.24) para reafirmar a sua vontade de colocar a impressão em *pause*.

A impressora pode demorar alguns segundos a colocar a impressão em pause, quando concluir esse processo mostra a vista de Impressão em pause como a figura 6.25.

Esta vista é igual à vista a imprimir mas o botão que era **pausar** passa agora a ser **retomar**. De notar que na figura 6.25 podemos ver que já foi removido um objeto da cena que tinha

FIGURA 6.24: UC04: Impressão de Cena 3D - Vista A Imprimir, Pausar impressão - confirmação.

inicialmente 3 objetos, a indicação **MODIFICADO** mostra também que a cena 3D já sofreu alterações.

O utilizador pode retomar ou cancelar a impressão. Se o utilizador continuar a impressão a impressora irá iniciar o processo de aquecimento como o da figura 6.15 para poder retomar a impressão. Se, por outro lado, o utilizador desejar cancelar a impressão deve clicar em ***stop*** e confirmar essa ação na vista seguinte como a da figura 6.23 e seguirá tal como descrito anteriormente para a vista arrefecimento do use case UC02 (figura 6.17).

FIGURA 6.25: UC04: Impressão de Cena 3D - Vista Em pause.

6.7 UC07: Trocar Filamento

Nesta secção vamos acompanhar a navegação da GUI BEE2B durante o processo de troca de filamento.

Para o utilizador trocar de filamento, na vista *Home* clica no botão **manutenção** da barra lateral como na figura 6.26 e segue para a vista Manutenção (figura 6.8).

Na Vista Manutenção o utilizador pode ver todos os filamentos atualmente instalados na impressora, para cada filamento tem acesso a alguma informação básica como o código do material, cor e quantidade restante de material. Para ter acesso aos seus detalhes, o utilizador seleciona o filamento desejado, seguido de clique no botão detalhes como na Figura 6.27.

Na vista detalhe de filamento o utilizador pode ver mais alguma informação como diâmetro do filamento e tempo de utilização, pode ainda usar as setas de navegação para alternar entre

FIGURA 6.26: UC07: Mudar Filamento - Vista *Home*.

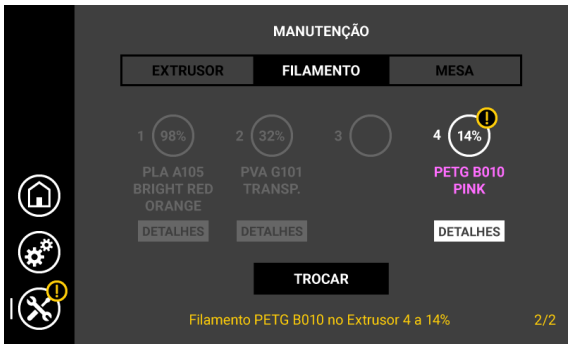


FIGURA 6.27: UC07: Mudar Filamento - Vista Manutenção, filamento 4 selecionado.

os vários filamentos. Para substituir o filamento o utilizador clica em **trocar** como na Figura 6.28.

FIGURA 6.28: UC07: Mudar Filamento - Vista Detalhes Filamento.

A Impressora vai preparar a ferramenta correspondente aquecendo o extrusor e removendo o filamento automaticamente, para cancelar este processo o utilizador pode clicar em ***stop*** como na figura 6.29. Para garantir que o utilizador não cancela a troca de filamento por engano, tem de confirmar a operação como na Figura 6.30. Devido ao elevado tempo que este processo demora é apresentado uma barra de progresso na barra lateral.

Depois da Impressora terminar o processo de remover o filamento o utilizador tem de manualmente remover a bobine de filamento da impressora indicando que concluiu a mesma remoção clicando em **seguinte** como na Figura 6.31.

FIGURA 6.29: UC07: Mudar Filamento - Vista A remover filamento 4.

FIGURA 6.30: UC07: Mudar Filamento - Vista A remover filamento 4, confirmação de botão **stop**.

FIGURA 6.31: UC07: Mudar Filamento - Vista Remover Bobine.

Para realizar a identificação automática o utilizador deve encostar a bobine de filamento ao sensor da impressora 3D. Como podemos ver na figura 6.32, nesta vista o utilizador tem mais duas opções disponíveis: **deixar sem filamento** e **inserir código manualmente**.

Caso o utilizador deseje deixar o extrusor sem filamento será apresentado uma alerta como na figura 6.33 voltando para a vista Manutenção com um clique em **ok**.

Se o utilizador necessitar de inserir o código manualmente segue para a vista de identificação manual de filamento como na figura 6.34, aqui o utilizador deve selecionar um filamento da lista disponivel e clicar em **selecionar** para seguir para a vista de novo de filamento (figura 6.35).

Seja quando o filamento é automaticamente identificado ou quando inserido manualmente a vista de novo filamento pede que o utilizador confirme a operação mostrando os detalhes do

FIGURA 6.32: UC07: Mudar Filamento - Vista A Identificar Automaticamente.

FIGURA 6.33: UC07: Mudar Filamento - Vista Filamento Vazio.

FIGURA 6.34: UC07: Mudar Filamento - Vista Identificação Manual.

filamento que está prestes a carregar. Para confirmar a operação o utilizar deve clicar em **confirmar** como na figura 6.35 seguindo assim para a vista Adicionar bobine (figura 6.37).

Se o filamento que o utilizador vai carregar for diferente, em cor ou no tipo de material, do último filamento utilizado pelo extrusor, o utilizador recebe uma alerta avisando que o material escolhido não corresponde ao último material utilizado. O utilizador pode confirmar a operação clicando em **continuar** como na figura 6.36 ou clicar em **corrigir** e voltar para a vista de deteção automática de filamento (figura 6.32).

Para adicionar a bobine à impressora 3D o utilizador deve colocar a bobine no suporte e inserir a ponta do filamento no furo correspondente, figura 6.37, depois a impressora vai detetar o novo filamento e seguir para a vista de Carregamento de filamento (figura 6.38). Caso o utilizador não complete este processo dentro de 30 segundos, o processo é abortado,

FIGURA 6.35: UC07: Mudar Filamento - Vista Novo Filamento.

FIGURA 6.36: UC07: Mudar Filamento - Vista Novo Filamento, Alerta de Material.

mudando a vista para A identificar automaticamente filamento.

FIGURA 6.37: UC07: Mudar Filamento - Vista Adicionar Bobine.

Tal como no descarregamento do filamento, durante o carregamento do filamento é também apresentado ao utilizador uma barra de progresso a indicar o estado do processo (figura 6.38), e a qualquer momento o utilizador pode cancelar este processo clicando no botão **stop**, onde mediante confirmação segue para a vista manutenção tal como nas figuras 6.29 e 6.30.

No final do carregamento de filamento é apresentado uma mensagem de sucesso da operação como na figura 6.39 seguindo para a vista manutenção mediante clique em **ok**, ou alerta de erro, neste caso o utilizador pode **repetir** o processo ou **cancelar** (figura 6.40). Se o utilizador cancelar volta para a vista manutenção (figura 6.27) se repetir segue para a vista de descarregamento de filamento (figura 6.29).

FIGURA 6.38: UC07: Mudar Filamento - Vista A Carregar Filamento.

FIGURA 6.39: UC07: Mudar Filamento - Vista A Carregar Filamento, Alerta Sucesso.

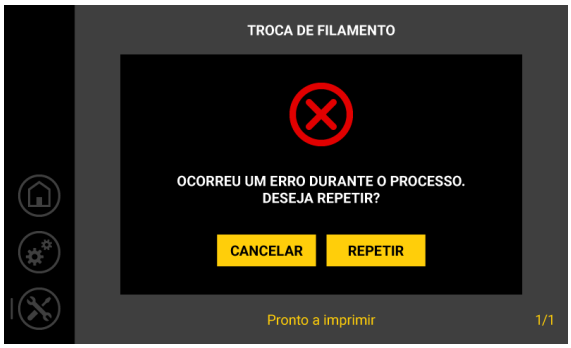


FIGURA 6.40: UC07: Mudar Filamento - Vista A Carregar Filamento, Alerta Erro.

6.8 UC08: Trocar Extrusor

Esta secção vai mostrar a navegação durante o caso de uso UC08: Trocar Extrusor (requisitos descritos na secção A.3.8).

Para trocar um extrusor o utilizador deve ir para a vista Manutenção (figura 6.9), clicar na tab Extrusor, selecionar o extrusor que deseja trocar ou carregar, caso a slot esteja vazia, e clicar no botão **trocar**.

Se o extrusor que o utilizador deseja trocar têm algum filamento carregado, o utilizador precisa de o remover primeiro, assim, a Impressora vai preparar a ferramenta correspondente aquecendo o extrusor e removendo o filamento automaticamente(figura 6.29) tal como no

FIGURA 6.41: UC08: Trocar Extrusor - Vista Manutenção : Separador Extrusor, extrusor 4 selecionado.

caso de uso UC07: Trocar Filamento.

Depois de remover o filamento, a impressora prepara o extrusor para ser removido, figura 6.42, este processo pode ser cancelado com o botão ***stop*** mediante confirmação conforme a figura 6.43. Quando o processo de preparação do extrusor estiver concluído, o utilizador é avisado como na figura 6.44 que pode remover o extrusor.

FIGURA 6.42: UC08: Trocar Extrusor - Vista A preparar extrusor.

FIGURA 6.43: UC08: Trocar Extrusor - Vista A preparar extrusor, confirmação de botão ***stop***.

O utilizador tem de manualmente remover o extrusor da impressora e depois clicar em **seguinte**. Na vista seguinte (figura 6.45), o utilizador apenas tem duas opções: ou **deixa sem extrusor**, sendo mostrado um aviso de extrusor vazio como na figura 6.46 ou **carrega** um novo **extrusor** como na figura 6.47.

FIGURA 6.44: UC08: Trocar Extrusor - Vista Remover extrusor.

FIGURA 6.45: UC08: Trocar Extrusor - Vista Sem ou Novo Extrusor.

Se o utilizador **deixar sem extrusor**, deve clicar **ok** como na figura 6.46 e segue para o menu Manutenção, separador extrusor, figura 6.9.

FIGURA 6.46: UC08: Trocar Extrusor - Vista Extrusor vazio.

A vista novo extrusor (figura 6.47), informa o utilizador que pode inserir o novo extrusor e posteriormente clicar em **seguinte**, obrigando assim o utilizador a voltar a GUI depois de colocar o novo extrusor por razões de segurança. A impressora vai detetá-lo automaticamente e mostrar ao utilizador os detalhes do extrusor que está prestes a adicionar como na figura 6.48.

O utilizador deve confirmar a informação do extrusor que está prestes a carregar e **confirmar** a operação como na figura 6.48 seguindo para a vista A carregar Extrusor (figura 6.49). Se o utilizador **cancelar**, volta para a vista A preparar extrusor como na figura 6.42 continuando a partir daí o fluxo normal.

FIGURA 6.47: UC08: Trocar Extrusor - Vista Novo extrusor.

FIGURA 6.48: UC08: Trocar Extrusor - Vista Novo extrusor : confirmação.

O utilizador pode cancelar o carregamento do extrusor com o botão ***stop*** (figura 6.49) mediante confirmação como o da figura 6.50

FIGURA 6.49: UC08: Trocar Extrusor - Vista A carregar extrusor.

Depois de carregar um novo extrusor, é altamente aconselhado efetuar o processo de calibração XY (secção 6.17), para garantir a melhor qualidade de impressão possível. O utilizador é questionado se deseja continuar com esta operação na vista Extrusor adicionado (figura 6.51), onde pode **cancelar** e voltar para o menu manutenção , separador Extrusor (figura 6.9), não efetuando a calibração XY do extrusor, ou **calibrar**, passando para a vista A Identificar automaticamente do UC07: Trocar Filamento (figura 6.32) seguindo este caso de uso até carregar um filamento. A vista A Identificar automaticamente filamento será ligeiramente diferente neste caso, tal como no caso de uso UC10: Configuração inicial, o botão **deixar sem filamento** vais estar inativo como na figura 6.63.

Depois de carregar um filamento, a GUI vai iniciar o processo de calibração XY para este

FIGURA 6.50: UC08: Trocar Extrusor - Vista A carregar extrusor - confirmação de ***stop***.

extrusor como na secção 6.17 mas iniciando logo na vista A imprimir regua de calibração, figura 6.102

FIGURA 6.51: UC08: Trocar Extrusor - Vista Extrusor adicionado.

6.9 UC09: Validar Periféricos

Esta secção vai mostrar a navegação durante o caso de uso UC09: Validar Periféricos (requisitos descritos na secção A.3.9).

FIGURA 6.52: UC09: Validar Periféricos - Vista Validação de Periféricos, aviso.

Sempre que a impressora for preparar as ferramentas para uma nova impressão, são validados os periféricos da impressora, se alguma coisa não estiver correta, por exemplo, se o diâmetro do *nozzle* ou filamentos instalados forem diferentes dos utilizados na , o processo de preparação

é interrompido e a GUI mostra um aviso como o da figura 6.52. Este aviso apenas informa que alguma ferramenta não está correta, o utilizado precisa de clicar no botão **detalhes** para ter acesso à informação detalhada do problema.

FIGURA 6.53: UC09: Validar Periféricos - Vista Validação de Periféricos, separador Extrusor.

Na vista validação de periféricos (figura 6.53) é mostrada informação relativa às várias ferramentas, sendo identificadas todas as ferramentas necessárias para a impressão prestes a iniciar, tanto as que estão corretas como aquelas onde forma detetadas problemas. O separador mostrada por defeito é a Extrusor, pois no caso de trocar o extrusor é necessário remover sempre primeiro o filamento.

Nesta vista o utilizador têm três opções: **cancelar**, **trocar** e **imprimir**.

Para cancelar a impressão o utilizador deve clicar em **cancelar** e depois confirmar esta ação no alerta de confirmação seguinte (figura 6.54).

FIGURA 6.54: UC09: Validar Periféricos - Vista Validação de Periféricos, cancelar impressão.

Se o utilizador quiser corrigir um extrusor deve selecioná-lo e clicar em **trocar**, de seguida deve confirmar a ação no alerta de troca de extrusor como o da figura 6.55.

Para ver os detalhes do filamento, o utilizador deve clicar no separador Filamento, tendo acesso à vista Validação de Periféricos, detalhes Filamento (figura 6.56), e confirmar tal como no extrusor a respetiva troca (figura 6.57)

Quando o utilizador estiver satisfeito com as alterações e desejar imprimir deve clicar em **imprimir**. Se todos os periféricos foram corrigidos a impressão inicia imediatamente, caso contrário, é mostrado um alerta como o da figura 6.58, onde o utilizador é avisado para o

FIGURA 6.55: UC09: Validar Periféricos - Vista Validação de Periféricos, troca extrusor.

FIGURA 6.56: UC09: Validar Periféricos - Vista Validação de Periféricos, detalhes filamento.

FIGURA 6.57: UC09: Validar Periféricos - Vista Validação de Periféricos, troca filamento.

facto de ainda existirem periféricos não corrigidos, mas permitindo-lhe ainda assim continuar a impressão.

FIGURA 6.58: UC09: Validar Periféricos - Vista Validação de Periféricos, continuar impressão

6.10 UC10: Configuração Inicial da Impressora

Esta secção mostra a navegação durante o caso de uso UC10: Configuração Inicial da Impressora (requisitos descritos na secção A.3.10).

Quando a impressora BEE2B é iniciada pela primeira vez necessita de uma configuração inicial. Este processo vai configurar a GUI BEE2B e as respetivas ferramentas. Este caso de uso também vai ser iniciado no final do restauro do software, quando o utilizador clica em **restaurar software** no menu Definições.

A primeira configuração que o utilizador vai fazer é escolher o idioma como na figura 6.59. Esta decisão vai influenciar o restante processo, pois a GUI já vai ser no idioma selecionado.

FIGURA 6.59: UC10: Configuração Inicial - Vista Seleção de Idioma.

Depois, a GUI mostra uma mensagem de boas vindas (figura 6.60).

Nesta vista, o utilizador clica em **começar** para avançar para a vista A preparar extrusor como a da figura6.42 do UC08: Trocar extrusor. Quando a impressora estiver preparada para o novo extrusor, a GUI mostra a vista Novo extrusor como o da figura 6.61.A impressora precisa de carregar pelo menos o extrusor 1 para conseguir imprimir.

Agora, o utilizador pode colocar o extrusor no slot 1, depois, por razões de segurança, tem de clicar em **seguinte**. A impressora vai detetar o extrusor colocado e apresentar essa informação ao utilizador como podemos ver na figura 6.62.

Quando o utilizador clicar **seguinte**, vai iniciar o processo de calibração Z como na figura **??** (caso de uso UC17) e segue o fluxo desse caso de uso até ao final da calibração. Quando a

FIGURA 6.60: UC10: Configuração Inicial - Vista bem-vindo.

FIGURA 6.61: UC10: Configuração Inicial - Vista Novo extrusor.

FIGURA 6.62: UC10: Configuração Inicial - Vista Novo extrusor : informação.

impressora terminar este processo, a GUI vai mostrar a vista A identificar filamento do caso de uso UC07: Trocar filamento como o da figura 6.63. Podemos observar que, o utilizador, no caso da configuração inicial, não têm a opção de **deixar sem filamento**, isto acontece porque a calibração XY inicial é obrigatória e esta calibração necessita de imprimir as réguas de calibração.

Depois de identificar o filamento, seja automaticamente, seja manualmente (figura fig:sb-uc07-7b1 do caso de uso UC7: Trocar de filamento), a GUI mostra a vista Novo Filamento ao utilizador, este pode confirmar o filamento que vai **carregar** ou **cancelar** e volta para a vista A identificar automaticamente filamento.

Se o utilizador clicar em **carregar**, a GUI vai mostrar a vista Adicionar bobine como na figura 6.37 do caso de uso UC07, depois do utilizador colocar a bobine no suporte e introduzir o filamento no furo correspondente, tal como nesse caso de uso, a GUI mostra a vista A carregar

FIGURA 6.63: UC10: Configuração Inicial - Vista A Identificar automaticamente filamento.

FIGURA 6.64: UC10: Configuração Inicial - Vista Novo Filamento.

filamento como na figura 6.38.

No final do carregamento do filamento a GUI questiona o utilizador se deseja carregar o extrusor 2 como na figura 6.61. Se o utilizador clicar **sim**, segue para a vista Novo extrusor como na figura 6.62 seguindo o fluxo deste caso de uso até o utilizador terminar o carregamento do filamento. A única exceção a este fluxo é a calibração Z (secção 6.16), esta calibração, apesar de ser feita por todos os novos extrusor, só é manual quando é o extrusor 1, para os restantes extrusor o processo é automático e sem intervenção do utilizador. Se o utilizador clicar **não** no final do carregamento do filamento a GUI questiona novamente o utilizador mas desta vez para o extrusor 3. Este processo é ainda repetido para o extrusor 4.

No final de todos os extrusores e respetivos filamentos estarem carregados, a impressora vai iniciar a impressão das réguas de calibração para todos os extrusores carregados e a GUI segue para a vista correspondente como na figura 6.102 do caso de uso UC18: Calibrar XY. Quando a impressora termina a impressão das réguas, são mostradas as instruções como na figura 6.104 seguindo para a vista Calibrar X (figura 6.65) e depois Calibrar Y (figura 6.66). Esta calibração é feita para todos os extrusores instalados ao mesmo tempo, os extrusores que estiverem vazios vão ter o *slider* correspondente inativo em ambas as calibrações.

No fim do processo de calibração, a GUI questiona o utilizador, tal como no UC18, se deseja imprimir um teste de impressão como na figura 6.107 seguindo o fluxo desse caso de uso até ao final.

Com a impressora pronta a imprimir (ferramentas e filamentos carregados, e calibrações efetuadas), a GUI informa o utilizador que para utilizar a impressora remotamente é necessário ligar a impressora a uma rede local como podemos ver na figura 6.67.

FIGURA 6.65: UC10: Configuração Inicial - Vista Calibrar X.

FIGURA 6.66: UC10: Configuração Inicial - Vista Calibrar Y.

FIGURA 6.67: UC10: Configuração Inicial - Vista Configuração de Rede.

Se o utilizador clicar em **configurar** vai realizar o caso de uso UC11: Modificar configurações de rede, a partir da figura 6.71. Ao clicar **cancelar** o processo de configuração inicial termina com a vista Configuração inicial concluída como podemos ver na figura 6.69.

Se o utilizador configurar uma rede, no final desse processo será questionado se deseja emparelhar um PC como podemos ver na figura 6.68.

Nesta vista, o utilizador pode clicar em **emparelhar** e seguir o caso de uso UC14: Emparelhar PC com impressora a partir da vista A aguardar emparelhamento (figura 6.86), ou pode clicar em **cancelar**, que, tal como no final do emparelhamento, termina a configuração inicial mostrando a figura 6.69.

Quando o utilizador clica **ok** nesta vista segue para a vista home, onde inicialmente vai estar a iniciar o sistema como na figura 6.5, e só no final deste processo o utilizar vai ter todas as

FIGURA 6.68: UC10: Configuração Inicial - Vista Emparelhar com PC.

FIGURA 6.69: UC10: Configuração Inicial - Vista Configuração inicial concluída.

funcionalidades disponiveis como na figura 6.3.

6.11 UC11: Modificar Configurações de Rede

Esta secção vai mostrar a navegação durante o caso de uso UC11: Modificar Configurações de Rede (requisitos descritos na secção A.3.11).

A opção de **configurar rede** é apresentada ao utilizador com um ícone que indica se existe ou não alguma rede atualmente ligada à impressora, na figura 6.70 vemos que a impressora não está ligada a nenhuma rede, podemos ainda verificar que enquanto a impressora não estiver ligada a nenhuma rede o botão emparelhar com PC está inativo.

FIGURA 6.70: UC11: Modificar Configurações de Rede - Vista Definições, Impressora sem rede.

Quando o utilizador inicia a configuração de rede e não tem nenhuma rede ligada à impressora, são apresentadas as instruções para ligar a impressora a uma rede. A vista A aguardar ligação (figura 6.71) fica ativa até o utilizador efetuar a ligação ou cancelar a ação com o botão **cancelar**. Para indicar que a impressora está a aguardar a ligação é mostrada uma animação exemplificada na figura 6.72, depois da ligação efetuada com sucesso o ícone de ligação à rede neste ecrã apresenta ainda a força do sinal da rede caso esta seja Wi-Fi.

FIGURA 6.71: UC11: Modificar Configurações de Rede - Vista A aguardar ligação.

FIGURA 6.72: UC11: Modificar Configurações de Rede - Animação de ícone de Rede para aguardar ligação e força de sinal.

Se o utilizador desejar cancelar a operação têm de confirmar esta ação, é apresentado um alerta de confirmação como a figura 6.73, aqui o utilizador pode clicar em **não** para continuar a aguardar ligação ou **sim** para confirmar o seu desejo de abandonar o processo de configuração de rede.

FIGURA 6.73: UC11: Modificar Configurações de Rede - Vista A aguardar ligação, alerta Confirmação de cancelamento.

Para efetuar a ligação da impressora a uma rede sem fios, a GUI informa o utilizador da rede e endereço que deve usar para, através de um *browser*, poder verificar as redes disponíveis para a impressora 3D e efetuar a ligação. Desta forma, visto a GUI não ter teclado físico ou

de ecrã, permitimos ao utilizador usar o seu PC para inserir a password da rede escolhida. A figura 6.74 mostra a sequência de ecrãs do *browser* para efetuar a ligação a um rede Wi-Fi.

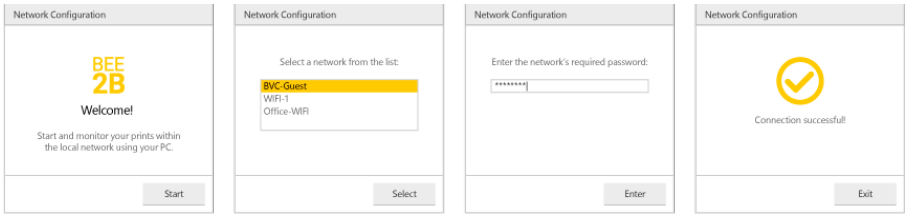


FIGURA 6.74: UC11: Modificar Configurações de Rede - Vista do *Browser* no endereço apresentado na GUI: Ligar a uma rede Wi-Fi.

Na figura 6.75 podemos ver o alerta que é mostrado quando existe algum erro durante a tentativa de ligação à rede, neste alerta podemos **repetir** e ficar novamente a aguardar ligação ou **cancelar** e voltamos para a vista Definições, figura 6.70.

FIGURA 6.75: UC11: Modificar Configurações de Rede - Vista Configuração de Rede, alerta falha de ligação.

Se o utilizador efetuar a ligação a uma rede com sucesso, a GUI mostra um aviso de ligação com sucesso indicando a rede à qual a impressora se ligou como na figura 6.76. Esta mensagem desaparece em **ok** ou passado alguns segundos ficando o utilizador na vista de configuração de rede com a informação da rede atualmente ligada 6.77.

FIGURA 6.76: UC11: Modificar Configurações de Rede - Vista Configuração de Rede, aviso concluído com sucesso.

Quando a impressora 3D está ligada a alguma rede e o utilizador clica em **configurar rede** da vista definições, a vista configuração de rede (figura 6.77) mostra informação detalhada da rede atualmente ligada como: nome da rede, força do sinal, ip e nome da impressora 3D.

Esta vista disponibiliza também o botão **ligar outra** para o utilizador efetuar uma ligação a uma rede diferente, seguindo novamente para o ecrã A aguardar ligação (figura 6.71).

FIGURA 6.77: UC11: Modificar Configurações de Rede - Vista Configuração de Rede, ligado à rede.

6.12 UC12: Restaurar Software de fábrica

Esta secção vai acompanhar o caso de uso UC12: Restaurar Software de fábrica (requisitos descritos na secção A.3.12).

Sempre que o utilizador desejar restaurar o software para a versão de fábrica deve ir ao menu Definições e selecionar a opção **restaurar software** (figura 6.6). A GUI apresenta a vista de Restauro do Software com uma mensagem que indica a versão que está atualmente a utilizar como na figura 6.78. Se o utilizador **cancelar** volta para o menu de definições se **restaurar** segue para a vista de confirmação de restauro como na figura 6.79.

FIGURA 6.78: UC12: Restaurar Software de fábrica - Vista Restauro de software.

Devido a ser um processo altamente destrutivo, o utilizador deve confirmar o processo de restauro conscientemente. O alerta de confirmação da figura 6.79 avisa o utilizador para os danos deste processo obrigando-o assim a reafirmar a sua intenção de realizar o restauro. Se o utilizador **cancelar** volta para o menu de definições e se **continuar** inicia o processo de restauro. Durante este processo o sistema vai reiniciar e vai arrancar com a configuração inicial da impressora descrito na secção 6.10.

FIGURA 6.79: UC12: Restaurar Software de fábrica - Vista Restauro de software, confirmação de restauro.

6.13 UC13: Atualizar Software

Nesta secção vamos ver a navegação do caso de uso UC13: Atualizar Software (requisitos descritos na secção A.3.13).

Quando existe uma atualização disponível para instalação, tanto o botão **atualizar software** como o botão **definições** vão apresentar uma notificação como na figura 6.80. O utilizador inicia o processo de atualização de software no botão **atualizar software** do menu Definições avançando para a vista Atualização de Software (figura 6.81).

FIGURA 6.80: UC13: Atualizar Software - Vista Definições, notificação de atualização pendente.

Na Vista Atualização de software, é apresentada informação sobre a versão que está atualmente a utilizar e informação sobre a nova versão (figura 6.81). O utilizador pode **cancelar** este processo voltando para o menu Definições ou **atualizar** para confirmar a intenção de atualizar o sistema seguindo para uma mensagem de alerta como a da figura 6.82 que avisa o utilizador que o sistema irá reiniciar.

Em caso de sucesso da operação, será apresentada a vista de Atualização de software com uma mensagem de sucesso como na figura 6.83 identificando a nova versão instalada.

Se, por outro lado, a operação não tiver sucesso, é mostrada uma mensagem de erro com as opções cancelar e repetir, onde o utilizador pode **cancelar** e voltar para o menu definições ou **repetir** o processo de atualização, neste caso segue para a vista de Atualização de software como na figura 6.82.

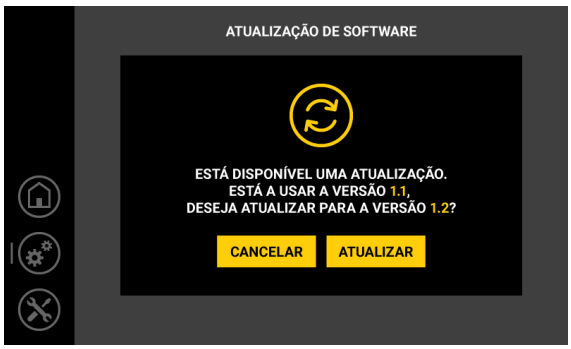


FIGURA 6.81: UC13: Atualizar Software - Vista Atualização de Software, atualização disponível.

FIGURA 6.82: UC13: Atualizar Software - Vista Atualização de Software, aviso que sistema irá reiniciar.

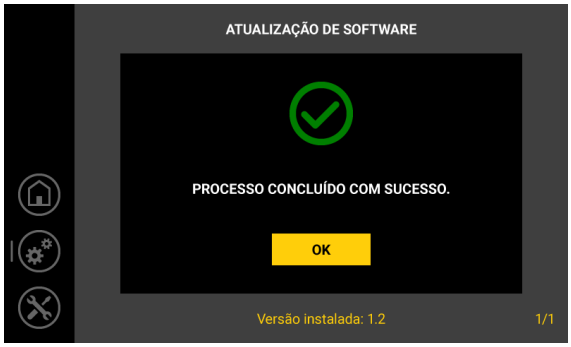


FIGURA 6.83: UC13: Atualizar Software - Vista Atualização de Software, aviso de sucesso da operação.

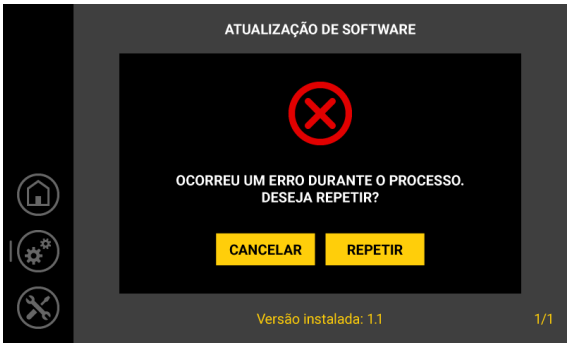


FIGURA 6.84: UC13: Atualizar Software - Vista Atualização de Software, alerta de erro na atualização.

6.14 UC14: Emparelhar PC com a Impressora

Esta secção vai acompanhar o caso de uso UC14: Emparelhar PC com a Impressora (requisitos descritos na secção A.3.14).

A opção **emparelhar com PC** do menu definições só está disponível se a impressora 3D estiver ligada a alguma rede. Este botão, à semelhança do seu vizinho **configurar rede**, também mostra uma notificação indicando se existe algum pc emparelhado no momento com a impressora (figura 6.85). O processo de emparelhamento é iniciado na GUI BEE2B e concluído no BEECura, onde o utilizador terá de inserir o código de emparelhamento.

FIGURA 6.85: UC14: Emparelhar PC com a Impressora - Vista Definições, Impressora com rede, nenhum PC emparelhado.

Quando o utilizador inicia o processo de emparelhamento, a GUI BEE2B mostra as instruções para concluir o processo, bem como outras informações relevantes como Nome da Impressora, rede atual e código de emparelhamento (figura 6.86). Para indicar que a impressora está a aguardar um emparelhamento é mostrada uma animação exemplificada na figura 6.87.

O utilizador deve agora deslocar-se ao seu PC e usar o BEECura opção extensions -> BEECura -> Printer Paring. O BEECura vai mostrar uma janela onde decorrerá o processo de emparelhamento.

Na da figura 6.88 vemos a sequência de vistas para concluir o processo. Na primeira vista o utilizador vê o nome do PC que está a usar e a que rede está ligado, juntamente com uma lista de impressoras 3D BEE2B na rede local, em modo de emparelhamento. O utilizador deve escolher a impressora que deseja emparelhar e clicar em ***Select***.

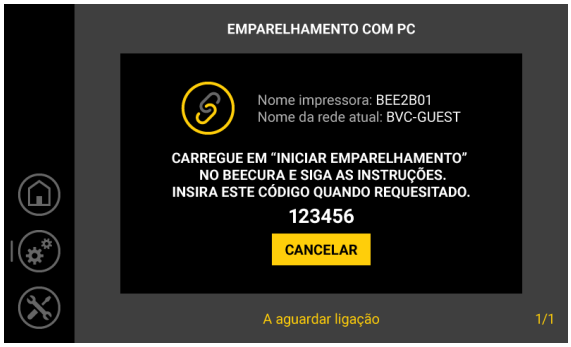


FIGURA 6.86: UC14: Emparelhar PC com a Impressora - Vista A aguardar emparelhamento.

FIGURA 6.87: UC14: Emparelhar PC com a Impressora - Animação de ícone de Emparelhamento para vista a aguardar emparelhamento e indicação de emparelhamento estabelecido.

Na segunda vista é pedido ao utilizador o código da impressora que deseja emparelhar, depois deve clicar em ***Pair*** para concluir o processo. Na vista final, para além de indicar o sucesso da operação, o utilizador pode adicionar um e-mail, para receber as notificações de processos relacionados com os ficheiros preparados pelo PC emparelhado.

FIGURA 6.88: UC14: Emparelhar PC com a Impressora - BEECura janela de emparelhamento com impressora.

Para parar este processo o utilizador deve clicar em **cancelar** da vista A aguardar emparelhamento (figura 6.86) e confirmar a operação no alerta de confirmação (figura 6.89), ao clicar **não** neste alerta a impressora continua em modo de emparelhamento.

Se ocorrer algum erro no emparelhamento é mostrado um alerta de erro como na figura 6.90 onde o utilizador pode, **cancelar** o processo e voltar para o menu definições ou **repetir** e continuar em modo de emparelhamento.(figura 6.86)

Quando o emparelhamento de um PC é efetuado com sucesso é apresentada uma mensagem de sucesso com o nome do PC que efetuou a ligação (figura 6.91). O utilizador pode clicar em **ok** ou esperar alguns segundos para esta mensagem desaparecer e ficar na vista de

FIGURA 6.89: UC14: Emparelhar PC com a Impressora - Vista A aguardar emparelhamento, alerta confirmação de cancelamento.

FIGURA 6.90: UC14: Emparelhar PC com a Impressora - Vista Emparelhamento com PC, alerta falha de emparelhamento.

Emparelhamento com PC (figura 6.92).

FIGURA 6.91: UC14: Emparelhar PC com a Impressora - Vista Emparelhamento com PC, aviso sucesso da operação.

Quando já existe algum PC emparelhado com a impressora, quando o utilizador seleciona a opção **emparelhar com PC** é mostrada a vista Emparelhamento com PC como na figura 6.92, onde é mostrado o nome da impressora e a rede atual. Nesta vista o utilizador tem dois botões disponíveis: **emparelhados** e **adicionar**.

O primeiro botão mostra uma lista de PCs emparelhados como na figura 6.93 e o segundo botão mete novamente a impressora em modo de emparelhamento como na figura 6.86.

Na vista de PCs Emparelhados (6.93)o utilizador vê os PCs atualmente emparelhados com a impressora e respetivos emails, têm também disponível o botão **desemparelhar** para poder

FIGURA 6.92: UC14: Emparelhar PC com a Impressora - Vista Emparelhamento com PC.

remover um PC da lista mediante de uma confirmação como a da figura 6.94.

FIGURA 6.93: UC14: Emparelhar PC com a Impressora - Vista PCs emparelhados.

FIGURA 6.94: UC14: Emparelhar PC com a Impressora - Vista PCs emparelhados, alerta de confirmação de desemparelhamento.

6.15 UC16: Mudar idioma

Esta secção vai acompanhar o caso de uso UC16: Mudar idioma (requisitos descritos na secção A.3.15).

Neste momento apenas existem dois idiomas disponíveis (português e inglês). Para o utilizador alternar entre elas deve, no menu definições (figura 6.6), escolher a opção **selecionar idioma**.

A vista Seleção de idioma mostra uma lista com todos os idiomas disponíveis e informação sobre o idioma atualmente em utilização (figura 6.95).

FIGURA 6.95: UC16: Mudar idioma - Vista Seleção de Idioma.

Para ativar outro idioma o utilizador deve selecionar o idioma pretendido e clicar em **selecionar**. O utilizador terá de confirmar esta ação como na figura 6.96, esta confirmação será já feita no novo idioma selecionado. Se o utilizador clicar **sim** muda o idioma de toda a GUI para o idioma selecionado, se clicar em **não**, a GUI mantém a mesmo idioma.

FIGURA 6.96: UC16: Mudar idioma - Vista Seleção de Idioma, confirmação.

6.16 UC17: Calibrar Offset Z

Esta secção vai acompanhar o caso de uso UC17: Calibrar Offset Z (requisitos descritos na secção A.3.16).

Sempre que o utilizador carregar um novo extrusor a impressora tem de calibrar o offset Z do extrusor, nos casos dos extrusores 2, 3 e 4 esta calibração é feita automaticamente sem intervenção do utilizador, mas se for o extrusor 1 o utilizador terá de efetuar este processo manualmente. Para além disso, o utilizador pode fazer a calibração Z em qualquer altura através do menu Manutenção separador Mesa botão **calibrar Z** (figura 6.11). Com esta opção, a impressora vai calibrar o extrusor 1 com ajuda do utilizador e depois configurar os extrusores 2, 3 e 4 automaticamente em função do primeiro.

Neste caso de uso, vamos acompanhar o utilizador quando este, clica no botão calibrar Z do menu Manutenção separador Mesa.

Quando o utilizador inicia a calibração Z a GUI mostra ao utilizador as instruções para o processo e fica a aguardar o **ok** do utilizador, como podemos ver na figura 6.97, para avançar para a vista Calibração Z (figura 6.98).

FIGURA 6.97: UC17: Calibração Z - Instruções de calibração Z.

FIGURA 6.98: UC17: Calibração Z - Vista Calibração Z.

Nesta vista, o utilizador pode subir ou descer a mesa em relação ao extrusor 1 nos botões com as setas. A seta normal movimenta a mesa uma distância curta, enquanto a seta dupla movimenta-a uma distância longa. Para concluir o processo, o utilizador deve clicar em **concluir**. O utilizador pode ainda rever as instruções de calibração no botão **instruções**.

No final da calibração a GUI mostra ao utilizador uma mensagem de sucesso, como a da figura 6.99, depois, o utilizador volta para o menu Manutenção separador Mesa clicando no botão **ok**.

FIGURA 6.99: UC17: Calibração Z - Vista Calibração Z concluida.

6.17 UC18: Calibrar *offset* XY

Esta secção vai acompanhar o caso de uso UC18: Calibrar Offset XY (requisitos descritos na secção A.3.17).

Sempre que o utilizador carregar um novo extrusor a impressora tem de calibrar o offset XY do extrusor. Para além disso, o utilizador pode fazer a calibração XY em qualquer altura através do menu Manutenção separador Extrusor, selecionar o extrusor e clicar no botão **calibrar XY** (figura 6.9). Durante o caso de uso UC10: Configuração inicial, descrito na secção 6.10, este processo é feito para todos os extrusores instalados ao mesmo tempo.

Como este processo vai demorar algum tempo e consumir material (vai ser necessário imprimir uma ou várias réguas de calibração), o utilizador necessita de confirmar a intenção de continuar com a calibração, assim, quando clica em **calibrar XY** no menu Manutenção separador Extrusor segue para o aviso de confirmação da figura 6.100.

FIGURA 6.100: UC18: Calibrar *offset* XY - Vista Confirmação de troca de filamento.

Caso o utilizador deseje confirmar a calibração do XY deve clicar em **continuar**, na vista seguinte deve indicar se apenas deseja fazer a calibração para o extrusor selecionado clicando em **atual**, ou se prefere fazer a **todos** os extrusores simultaneamente, como na figura 6.101.

FIGURA 6.101: UC18: Calibrar *offset* XY - Vista Escolha de extrusores a calibrar.

Depois, a impressora começa a imprimir as réguas de calibração para os extrusores selecionados e segue para a vista A imprimir réguas de calibração como na figura 6.102.

Na vista A imprimir réguas de calibração (figura 6.102), enquanto a impressora imprime as réguas de calibração que serão utilizadas no passo seguinte, é mostrada uma barra de progresso da operação. Se o utilizador desejar pode cancelar o processo com o botão ***stop***, mas terá de confirmar essa intenção como mostra a figura 6.103

Se o utilizador confirmar em **sim**, volta para o menu Manutenção separador Extrusor se, por outro lado, clicar em não, continua com o processo de impressão. No final desta impressão são mostradas as instruções de calibração ao utilizador como se vê na figura 6.104.

FIGURA 6.102: UC18: Calibrar *offset* XY - Vista A imprimir réguas de calibração.

FIGURA 6.103: UC18: Calibrar *offset* XY - Vista A imprimir réguas de calibração, confirmação de botão ***stop***.

FIGURA 6.104: UC18: Calibrar *offset* XY - Instruções de calibração XY.

O utilizador deve confirmar que leu as instruções clicando no botão ok, para seguir para a vista de Calibração X que podemos ver na figura 6.105.

A vista Calibração X mostra *sliders* para ajustar os *offsets* no eixo x (no caso de se estar a calibrar apenas um extrusor os outros *sliders* ficam inativos) e um botão **instruções**, para obter as instruções. Depois de ajustar o *slider* com o valor correto, o utilizador deve clicar **seguinte** para ajustar o eixo y (figura 6.106).

Tal como na vista anterior, o utilizador deve ajustar os *sliders* correspondentes aos extrusores que está a calibrar e clicar em **seguinte**. No final da calibração é apresentada uma mensagem de calibração concluída como na figura 6.107.

Para verificar a fiabilidade da calibração efetuada, é aconselhado que a impressora realize uma

FIGURA 6.105: UC18: Calibrar *offset* XY - Calibração X.

FIGURA 6.106: UC18: Calibrar *offset* XY - Calibração Y.

FIGURA 6.107: UC18: Calibrar *offset* XY - Calibração concluída.

impressão de teste, por isso, essa impressão é sugerida ao utilizador. O utilizador pode ignorar este teste e voltar para o menu manutenção separador Extrusor clicando em **cancelar**, ou realizar a impressão ao clicar em **imprimir** e seguir para a vista A imprimir teste como na figura 6.108.

Ao contrário da vista de impressão normal, esta não mostra a cena 3D, mas continua a disponibilizar o botão ***stop***, para que o utilizador possa cancelar o teste. Para cancelar, o utilizador necessita de o confirmar no alerta de confirmação mostrado na figura 6.109.

No final da impressão de teste, a GUI questiona o utilizador se a impressão foi bem sucedida (figura 6.110), o utilizador deve verificar a mesma e clicar **concluir** se ficou satisfeito com a calibração ou se, por outro lado, ficou insatisfeito e deseja efetuar nova calibração, clicar em **repetir**. Se o utilizador repetir o processo a impressora irá novamente aquecer e imprimir as réguas de calibração como na figura 6.103, o fluxo do processo seguirá a partir desse ponto

FIGURA 6.108: UC18: Calibrar *offset* XY - A imprimir teste.

FIGURA 6.109: UC18: Calibrar *offset* XY - A imprimir teste, confirmação de botão ***stop***.

até ao final. Se o utilizador concluir o processo segue para a vista Manutenção separador Extrusor.

FIGURA 6.110: UC18: Calibrar *offset* XY - Impressão de teste concluída.

6.18 UCC4: Gerir Fila de Impressão.

Nesta secção vamos acompanhar a navegação do BEECura durante o caso de uso UC05: Gerir Fila de Impressão (requisitos descritos na secção A.3.18.4).

Para gerir a fila de impressão o utilizador deve iniciar, no BEECura, a opção extensions -> BEECura -> PrintingQueue. Será mostrada a janela da fila de impressão, como na figura 6.111, onde o utilizador consegue visualizar toda a fila de impressão.

Em destaque no topo da janela aparece a impressão atual que o utilizador, mediante confirmação, pode cancelar. Por baixo da impressão atual aparece a restante fila, no topo dessa fila está a impressão seguinte que irá iniciar caso a atual seja cancelada.

Para alterar a ordem de impressão da fila, o utilizador apenas tem de mover as impressões para baixo ou para cima, ordenando-as como desejar. o utilizador pode ainda remover uma impressão da fila de impressão.

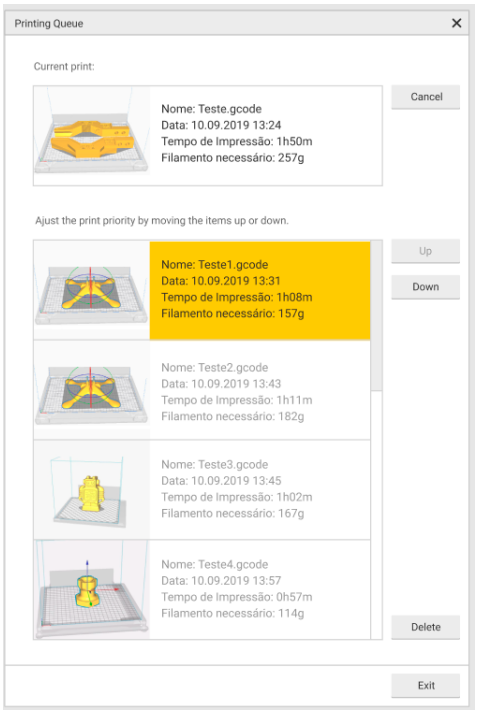


FIGURA 6.111: BEECura - Janela Gestão de fila de impressão.

Capítulo 7

Construção da solução

Neste capítulo é apresentado o desenvolvimento da solução. Na secção 7.1 é mostrada a organização do software dos módulos GUI e PrintServer. Na secção 7.2 é descrita a implementação da pipeline no servidor de entrega contínua. Depois é apresentado os apetos mais importantes do desenvolvimento dos vários módulos, na secção 7.3 a GUI BEE2B, o PrintServer na secção 7.4 e por fim, na secção 7.5, o BEECura.

7.1 Organização do Software

Todos os módulos do projeto usaram o gestor de repositórios de software GitLab. Foram criados dois repositórios, um para o plugin do Ultimaker Cura (BEECura) e outro para o servidor e interface gráfica. Esta secção descreve a organização dos módulos PrintServer BEE2B e GUI BEE2B no repositório git.

Na figura 7.1 podemos ver a visão geral da estrutura de ficheiros do repositório destes dois módulos.

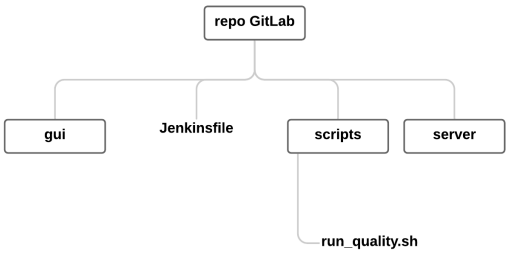


FIGURA 7.1: Repositório GitLab de projeto BEE2B.

As pastas **gui** e **server** correspondem aos módulos GUI BEE2B e PrintServer BEE2B respetivamente, e serão detalhadas nas secções 7.1.1 e 7.1.2 respetivamente.

Na pasta ***scripts*** são guardados *scripts* gerais que dependem dos dois módulos, como o run_quality.sh (*script* 7.6).

O ficheiro **Jenkinsfile** contém a configuração dos vários Estágios da pipeline BEE2B para o servidor de entrega contínua, este servidor será detalhado na secção 7.2.

7.1.1 GUI BEE2B

A pasta **gui** contém todos os scripts e código fonte da GUI BEE2B e é composto por:

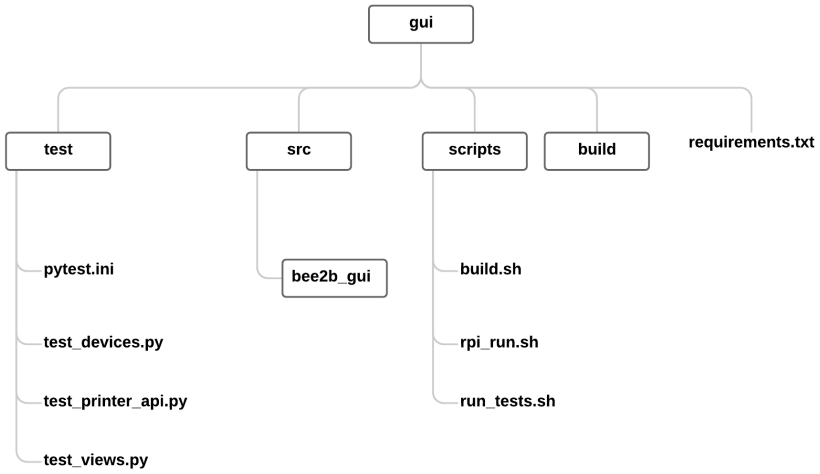


FIGURA 7.2: Repositório GitLab de projeto BEE2B, pasta gui.

- pasta **test** - contém todos os scripts python de teste para usar com a ferramenta pytest. Estes testes são detalhados na secção 7.3.5;
- pasta **src/bee2b_gui** - código fonte da aplicação, é apresentado na secção 7.3;
- pasta **scripts** - aqui são guardados todos os scripts utilizados pelo Jenkinsfile do servidor de entrega contínua;
- pasta **build** - pasta onde serão guardados os ambiente virtuais e outros ficheiros temporários;
- **requirements.txt** - este ficheiro contém a descrição dos requisitos da aplicação GUI BEE2B.

7.1.2 PrintServer BEE2B

Esta secção descreve a estrutura de ficheiros do módulo **server**, este módulo contém todos os *scripts* e código fonte do PrintServer BEE2B, como podemos ver na figura 7.3, é composto por:

- pasta **scripts** - Nesta pasta são guardados scripts utilziados pelo servidor de entrega contínua para criar o ambiente virtual da aplicação, gerar a documentação, correr o servidor e executar os testes unitários do PrintServer.
- pasta **printserver** - pasta raiz da aplicação Django, esta aplicação é detalhada na secção 7.4;
- pasta **build** - ficheiros temporários criados durante a pipeline BEE2B, por várias ferramentas como coverage ou unittest, são guardados nesta pasta;
- pasta **templates** - *templates* utilizados pela ferramenta apidoc. Esta ferramenta é utilizada para gerar automaticamente a documentação da API do servidor de impressão;
- **requirements.txt** - ficheiro com requisitos da aplicação PrintServer BEE2B;
- **apidoc.json** - ficheiro com as configurações da ferramenta apidoc.

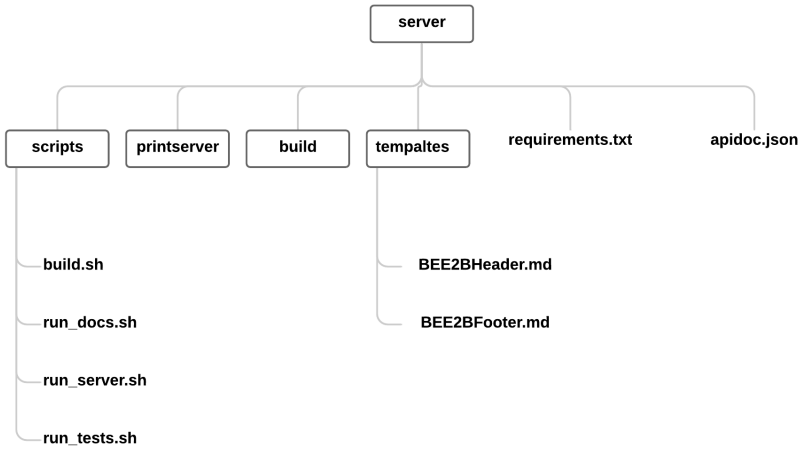


FIGURA 7.3: Repositório GitLab de projeto BEE2B, pasta server.

7.2 Servidor Entrega Contínua

Integração contínua e Entrega contínua são práticas que hoje em dia todas as empresas que desenvolvem software tentam implementar, pois estas práticas facilitam a integração, teste e publicação do trabalho diário.

"Integração contínua é uma prática de desenvolvimento de software em que os membros de uma equipa integram o seu trabalho com frequência, geralmente diariamente, levando a várias integrações por dia. Cada integração é verificada por uma compilação automática (incluindo testes) para detetar erros de integração o mais rápido possível. Muitas equipas acham que esta abordagem leva a problemas de integração significativamente reduzidos e permite que a equipa desenvolva software coeso mais rapidamente."(Fowler 2006)

Sempre que um programador faz alterações na aplicação, ou seja, envia o novo código para o repositório (geralmente GIT) da aplicação, é acionada uma compilação automática que inclui testes. Este processo permite detetar problemas de integração verificando se o novo código inserido quebra alguma parte da aplicação. Também permite deteção de erros quando os testes unitários falham.

"Entrega contínua é uma disciplina de desenvolvimento de software em que se constrói o software de forma a que o software possa ser enviado para produção a qualquer momento."(Fowler 2013a)

Para implementar a Entrega Contínua é necessário integrar o software continuamente feito pela equipa de desenvolvimento, construindo os executáveis e executando os testes automatizados com esse executável para detetar problemas. Para além disso, este executáveis são colocados em ambientes progressivamente mais parecidos com a produção para garantir que o software funciona na produção. Para fazer isto, geralmente usa-se uma *deployment pipeline*.

"Um dos desafios da compilação e testes automatizados, é a necessidade de uma compilação rápida, para obter um *feedback* rápido, mas testes abrangentes demoram muito tempo a serem executados. Uma *deployment pipeline* é uma forma de lidar com este desafio, dividindo a sua contrução em estágios. Cada estágio fornece confiança crescente, geralmente à custa de tempo extra. Os estágios iniciais podem encontrar a maioria dos problemas, gerando

feedback mais rápido, enquanto os estágios posteriores fornecem um *feedback* mais lento. As *deployment pipelines* são uma parte fulcral da Entrega Contínua."(Fowler 2013b)

O principal benefício desta abordagem é que esta cria um processo de *release* recetível, confiável e previsível, que por sua vez, gera grandes reduções no tempo de ciclo de desenvolvimento do software e, assim, obter recursos e correções para os utilizadores rapidamente (Humble 2010).

Para o desenvolvimento da GUI BEE2B e servidor de impressão foi utilizado esta abordagem, como o python não necessita de compilação (é uma linguagem interpretada) a sua compilação no âmbito da Entrega continua foca-se na construção do ambiente de execução e na execução dos testes.

Esta abordagem neste projeto teve várias vantagens:

- ambiente de execução - automatizando o ambiente de execução possibilita verificar se todos os requisitos da aplicação estão disponíveis para instalação e são instalados nos vários sistemas operativos.
- testes automáticos - Todos os testes são executados de forma automática e apresentam a respetiva cobertura. Testes unitários da Interface Gráfica e servidor de impressão. Testes de integração com API. Teste de *Deployment*. Testes de aceitação funcionais.
- documentação automática - A cada compilação é gerada automaticamente toda a documentação: Interface gráfica, Servidor de Impressão e correspondente API. Isto permite ter a cada momento todo o software com documentação atualizada.
- monitorização da qualidade do código - Monitorização constante da qualidade do código relativamente ao padrão python *Python Enhancement Proposal* (PEP)8.

Para a implementação do servidor de entrega contínua foi utilizado o Jenkins. Aqui configurou-se a *Pipeline* BEE2B com uma ligação ao repositório GitLab, para que, a cada nova entrada de código iniciasse uma nova compilação. Foi utilizado um *script* Jenkinsfile com toda a configuração dos vários Estágios que serão detalhados nas próximas secções. Podemos ver o projeto do Jenkins na figura 7.4 e o estado de desenvolvimento da pipeline na figura 7.5

7.2.1 Ambiente Virtual

```
1    stage('BuildEnvironment'){
2            steps{
3                dir("server") {
4                    sh 'chmod +x -R scripts/build.sh'
5                    sh 'scripts/build.sh'
6                }
7                dir("gui") {
8                    sh 'chmod +x -R scripts/build.sh'
9                    sh 'scripts/build.sh'
10               }
11           }
12   }
```

Listing 7.1: Jenkinsfile - Estágio BuildEnvironment

O primeiro estágio da *Pipeline* BEE2B é a criação dos ambientes virtuais (BuildEnvironment) para a execução das aplicações PrintServer BEE2B e GUI BEE2B como podemos ver em 7.1.

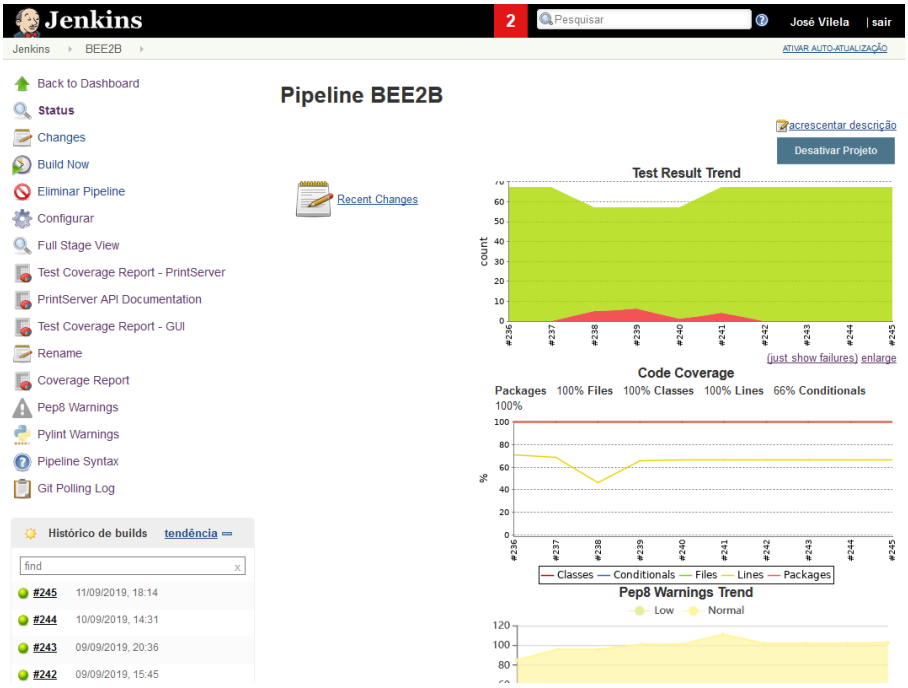


FIGURA 7.4: Projeto Jenkins BEE2B.

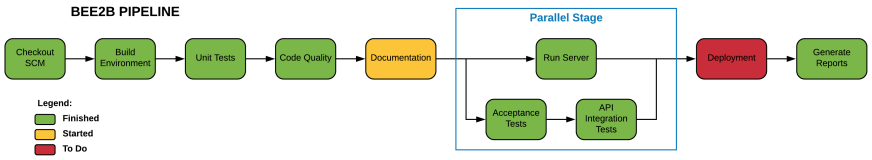


FIGURA 7.5: Estado de desenvolvimento da Pipeline BEE2B.

O primeiro ambiente virtual a ser criado é o do servidor de impressão, primeiro muda-se o diretório atual para o diretório do servidor (linha 3), depois torna o *script* scripts/build.sh executável e posteriormente executa-o.

```
1  #!/bin/bash
2
3  ## PrintServer
4
5  # build directory
6  if [[ -d build ]]; then
7    rm -rf build
8  fi
9  mkdir build
10
11 if [[ -d htmlcov ]]; then
12   rm -rf htmlcov
13   rm -f .coverage
14 fi
15
16 # Create/Activate PrintServer virtualenv
17 cd build
18 virtualenv -p python3 printserver_venv
19 source printserver_venv/bin/activate
20 cd ..
21
22 # Install Requirements
23 pip install -r requirements.txt
24
25 deactivate
```

Listing 7.2: server/scripts/build - Criação de ambiente virtual do servidor de impressão

Este *script* (7.2) vai, inicialmente limpar as pastas build e htmlcov (linhas 6 a 14), depois cria o ambiente virtual na pasta build e ativa-o (linhas 17 a 20), por fim são instalados todos os requisitos nessa ambiente virtual e desativa esse ambiente.

Depois deste script é iniciado o script para criar o ambiente virtual da Interface Gráfica que é semelhante mas agora vai instalar os requisitos da GUI no ambiente virtual criado em gui/build.

7.2.2 Testes Unitários

O segundo estágio da *Pipeline* é o dos testes unitários (figura 7.3).

```
1      stage('Unit Tests'){
2              steps{
3                    dir("server") {
4                        sh 'chmod +x -R scripts/run_tests.sh'
5                        sh 'scripts/run_tests.sh'
6                    }
7              }
```

Listing 7.3: Jenkinsfile - Estágio Unit Tests

Neste estágio apenas são executados testes unitários ao servidor de impressão.

```
#!/bin/bash
## PrintServer BEE2B - Run Unit Tests

# Activate PrintServer BEE2B virtualenv
source build/printserver_venv/bin/activate

# Clean old data
cd printserver
if [[ -d htmlcov ]]; then
  rm -rf htmlcov
  rm .coverage
  rm coverage.xml
fi

# Run PrintServer BEE2B coverage test
coverage run manage.py test -v 2

# Generate html and xml reports
coverage html --omit=*/migrations/*,*/build/*
coverage xml --omit=*/migrations/*,*/build/*

# Deactivate PrintServer BEE2B virtualenv
deactivate
```

LISTING 7.4: server/scripts/run_tests.sh - Execução de testes unitários do PrintServer e respetiva cobertura

No *script* 7.4 são executados os testes com a ferramenta coverage, que vai criar um ficheiro .coverage, que por sua vez será utilizado para gerar os relatórios de cobertura html e xml.

7.2.3 Qualidade de Código

No estágio seguinte é analisada a qualidade do código segundo o padrão PEP8, podemos ver este este estágio na figura 7.5.

```
    stage('Code Quality'){
            steps{
                sh 'chmod +x -R Scripts/run_quality.sh'
                sh 'Scripts/run_quality.sh'
            }
    }
```

LISTING 7.5: Jenkinsfile - Estágio Code Quality

Neste estágio são utilizadas duas ferramentas para obter as métricas de qualidade, o flake8 e o pylint.

```
#!/bin/bash

## GUI BEE2B - Code Quality

# Activate GUI BEE2B virtualenv
source gui/build/gui_venv/bin/activate

# Analyse GUI BEE2B Code with flake8 and pylint
flake8 . --max-line-length=120 --exclude=build,resources,test,server,bee_plugins > gui/build/flake8.txt
sed -i 's/^[./]*//' gui/build/flake8.txt

pylint --max-line-length=120 --disable=E0611,R0903 -f parseable gui/src/bee2b_gui/model gui/src/bee2b_gui/controller gui/src/bee2b_gui/view gui/src/bee2b_gui/services | tee gui/build/pylint.txt

# Deactivate GUI BEE2B virtualenv
deactivate

## PrintServer BEE2B - Code Quality

# Activate PrintServer BEE2B virtualenv
source server/build/printserver_venv/bin/activate

# Analyse PrintServer BEE2B Code with flake8 and pylint
flake8 . --max-line-length=120 --exclude=build,test,gui,bee_plugins > server/build/flake8.txt
sed -i 's/^[./]*//' server/build/flake8.txt
pylint --max-line-length=120 --disable=E0611 -f parseable --ignore=migrations server/printserver/printserver server/printserver/connectivity server/printserver/file_manager server/printserver/maintenance | tee server/build/pylint.txt

# Deactivate PrintServer BEE2B virtualenv
deactivate
```

LISTING 7.6: scripts/run_quality.sh - Execução da análise de qualidade de código

O *script* 7.6 executa análises de qualidade a ambas as aplicações começando pela GUI. Primeiro ativa o ambiente virtual da respetiva aplicação executando depois os comandos flake8 e pylint, no final desativa o respetivo ambiente virtual.

7.2.4 Documentação

Neste estágio será gerado toda a documentação da solução,podemos vê-lo no *script* 7.7.

```
    stage('Documentation'){
            steps{
                dir("server") {
                    sh 'chmod +x -R scripts/run_docs.sh'
                    sh 'scripts/run_docs.sh'
                }
            }
    }
```

LISTING 7.7: Jenkinsfile - Estágio Code Quality

O *script* 7.8 executa a geração da documentação da REST API do servidor de impressão (módulo connectivity.api). Este relatório está disponível no apêndice B. Primeiro ativa o ambiente virtual do servidor executando depois o comando apidoc, no final desativa o ambiente virtual.

Neste momento, apenas está implementado a geração automática da documentação da REST API do servidor de impressão utilizando a ferramenta apidoc. Para a GUI BEE2B e restantes módulos do servidor de impressão será usada a ferramenta Sphinx.

```
#!/bin/bash

## PrintServer API - Documentation

# Activate PrintServer virtualenv
source build/printserver_venv/bin/activate

# Generate documentation
apidoc -i printserver/connectivity/api/ -o ./apidoc/

# Deactivate PrintServer virtualenv
deactivate
```

LISTING 7.8: server/scripts/run_docs.sh - Geração automática da documentação da REST API

7.2.5 Testes de Integração, Aceitação e *Deployment*

Este estágio é diferente dos anteriores, pois vai executar dois estágios em paralelo (*script* 7.9).

```
1     stage('Deployment, Integration and Acceptance Tests'){
2             failFast true
3             parallel{
4         stage("Run Server") {
5           steps {
6             dir("server") {
7                             sh 'chmod +x -R scripts/run_server
      .sh'
8                             sh 'scripts/run_server.sh'
9                           }
10          }
11        }
12        stage('Tests') {
13          steps{
14              sleep 15
15              sh 'curl --silent --head "http
      ://127.0.0.1:8000/admin" | head -1 | cut -f 2 -d" "'
16                          dir("gui") {
17                              sh 'chmod +x -R scripts/run_tests
      .sh'
18                              sh 'scripts/run_tests.sh'
19                          }
20                          sh 'pkill -f runserver'
21          }
22        }
23
24      }
25    }
```

LISTING 7.9: Jenkinsfile - Estágio *Deployment, Integration & Acceptance Tests*

No primeiro desses estágios, o Run Server, é iniciado o servidor de impressão (linha 8). No segundo estágio, o Tests, são efetuados os testes de Integração, Aceitação e *Deployment*, ou smoke test (linha 15). No final deste segundo estágio termina-se o servidor (linha 20).

```
1  #!/bin/bash
2
3  ## Run PrintServer BEE2B
4
5  # Activate PrintServer BEE2B virtualenv
6  source build/printserver_venv/bin/activate
7
8  cd printserver
9
10 # Migrate PrintServer BEE2B database
11 python manage.py migrate
12
13 # Start PrintServer BEE2B
14 python manage.py runserver
15
16 # Deactivate PrintServer BEE2B virtualenv
17 deactivate
```

LISTING 7.10: server/scripts/run_server.sh - Script de inicialização de servidor de impressão BEE2B

O *script* 7.10 começa por ativar o ambiente virtual do PrintServer BEE2B, depois muda para

a pasta printserver e migra a base de dados, assim que terminar esta migração é iniciado o servidor, no final o ambiente virtual é desativado. O servidor estará a correr durante a execução dos testes do outro estágio.

```
#!/bin/bash

# Activate PrintServer BEE2B virtualenv
source build/gui_venv/bin/activate

# Export workspace to PYTHONPATH
export PYTHONPATH=${WORKSPACE}:${PYTHONPATH}

# Run PyTest tests with Xvfb
xvfb-run -e /dev/stdout pytest --junit-xml=build/
    gui_tests_junit.xml --cov-report=html --cov-report=xml --
    cov-report=term --fulltrace --cov=src/bee2b_gui/

# Deactivate PrintServer BEE2B virtualenv
deactivate
```

LISTING 7.11: gui/scripts/run_tests.sh - Execução de testes de integração e aceitação da GUI BEE2B e REST API

Com o servidor ligado, o *script* 7.11 executa os teste de aceitação e integração. Inicialmente ativa o ambiente virtual da GUI BEE2B depois exporta o caminho do WORKSPACE para o PYTHONPATH, de seguida são executados os testes de integração (com pedidos à API do PrintServer) e de aceitação. Como o servidor não têm ecrã disponível, usou-se a ferramenta Xvfb como *Display Server*. No final desativa o ambiente virtual.

7.2.6 Geração de relatórios

No estágio de geração de relatórios, são construídos os relatórios de testes unitários xml com o *plugin* e relatórios de cobertura xml com a ferramenta coverage (7.12).

```
    stage('Generate Reports') {
            steps {
                junit 'server/build/xunittest.xml, gui/build/
    gui_tests_junit.xml'
                cobertura autoUpdateHealth: false,
    autoUpdateStability: false,
                coberturaReportFile: '**/server/printserver/
    coverage.xml, **/gui/coverage.xml, **/api_test/coverage.
    xml',
                conditionalCoverageTargets: '70, 0, 0',
    failUnhealthy: false, failUnstable: false,
                lineCoverageTargets: '80, 0, 0',
    maxNumberOfBuilds: 0, methodCoverageTargets: '80, 0, 0',
                onlyStable: false, sourceEncoding: 'ASCII',
    zoomCoverageChart: false
```

LISTING 7.12: Jenkinsfile - Estágio *Generate Reports* : junit e coverage

Depois são gerados os relatórios html para o PrintServer BEE2B (7.13), PrintServer BEE2B API (7.14) e GUI BEE2B (7.15).

```
1                publishHTML([allowMissing: false,
    alwaysLinkToLastBuild: false, keepAll: false,
2                reportDir: 'server/printserver/htmlcov/',
    reportFiles: 'index.html',
3                reportName: 'Test Coverage Report -
    PrintServer',
4                reportTitles: 'Test Coverage Report -
    PrintServer'])
```

LISTING 7.13: Jenkinsfile - Estágio *Generate Reports*: relatório HTML PrintServer BEE2B

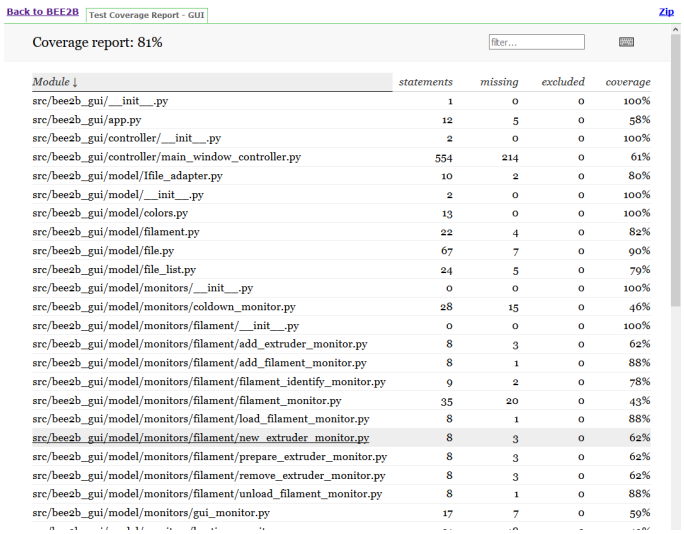
```
1                publishHTML([allowMissing: false,
    alwaysLinkToLastBuild: false, keepAll: false,
2                reportDir: 'server/apidoc/', reportFiles: '
    index.html', reportName: 'PrintServer API Documentation',
3                reportTitles: 'PrintServer API Documentation'
    ])
```

LISTING 7.14: Jenkinsfile - Estágio *Generate Reports*: relatório HTML PrintServer BEE2B API

```
1                publishHTML([allowMissing: false,
    alwaysLinkToLastBuild: false, keepAll: false,
2                reportDir: 'gui/htmlcov/', reportFiles: '
    index.html', reportName: 'Test Coverage Report - GUI',
3                reportTitles: 'Test Coverage Report - GUI'])
```

LISTING 7.15: Jenkinsfile - Estágio *Generate Reports*: relatório HTML GUI BEE2B

Na figura 7.6 vemos um exemplo do relatório de cobertura de testes da GUI gerado pelo *script* 7.15.



Module ↓	statements	missing	excluded	coverage
src/bee2b_gui/__init__.py	1	0	0	100%
src/bee2b_gui/app.py	12	5	0	58%
src/bee2b_gui/controller/__init__.py	2	0	0	100%
src/bee2b_gui/controller/main_window_controller.py	554	214	0	61%
src/bee2b_gui/model/Ifile_adapter.py	10	2	0	80%
src/bee2b_gui/model/__init__.py	2	0	0	100%
src/bee2b_gui/model/colors.py	13	0	0	100%
src/bee2b_gui/model/filament.py	22	4	0	82%
src/bee2b_gui/model/file.py	67	7	0	90%
src/bee2b_gui/model/file_list.py	24	5	0	79%
src/bee2b_gui/model/monitors/__init__.py	0	0	0	100%
src/bee2b_gui/model/monitors/coldown_monitor.py	28	15	0	46%
src/bee2b_gui/model/monitors/filament/__init__.py	0	0	0	100%
src/bee2b_gui/model/monitors/filament/add_extruder_monitor.py	8	3	0	62%
src/bee2b_gui/model/monitors/filament/add_filament_monitor.py	8	1	0	88%
src/bee2b_gui/model/monitors/filament/filament_identify_monitor.py	9	2	0	78%
src/bee2b_gui/model/monitors/filament/filament_monitor.py	35	20	0	43%
src/bee2b_gui/model/monitors/filament/load_filament_monitor.py	8	1	0	88%
src/bee2b_gui/model/monitors/filament/new_extruder_monitor.py	8	3	0	62%
src/bee2b_gui/model/monitors/filament/prepare_extruder_monitor.py	8	3	0	62%
src/bee2b_gui/model/monitors/filament/remove_extruder_monitor.py	8	3	0	62%
src/bee2b_gui/model/monitors/filament/unload_filament_monitor.py	8	1	0	88%
src/bee2b_gui/model/monitors/gui_monitor.py	17	7	0	59%

FIGURA 7.6: Projeto Jenkins BEE2B: Relatório de cobertura de testes da GUI.

Por fim, regista os resultados da análise estática de código das ferramentas flake8 e pylint (7.16).

```
1                 recordIssues enabledForFailure: true, tool:
    pep8(pattern: 'server/build/flake8.txt,gui/build/flake8.
    txt'),
2                 sourceCodeEncoding: 'UTF-8', referenceJobName
    : 'Plugins/warnings-ng-plugin/master'
3                 recordIssues enabledForFailure: true, tool:
    pyLint(pattern: 'server/build/pylint.txt, gui/build/
    pylint.txt'),
4                 sourceCodeEncoding: 'UTF-8', referenceJobName
    : 'Plugins/warnings-ng-plugin/master'
```

LISTING 7.16: Jenkinsfile - Estágio *Generate Reports*: flake8 e pylint

Na figura 7.7 podemos ver um exemplo da apresentação dos resultados da análise feita pela ferramenta flake8.

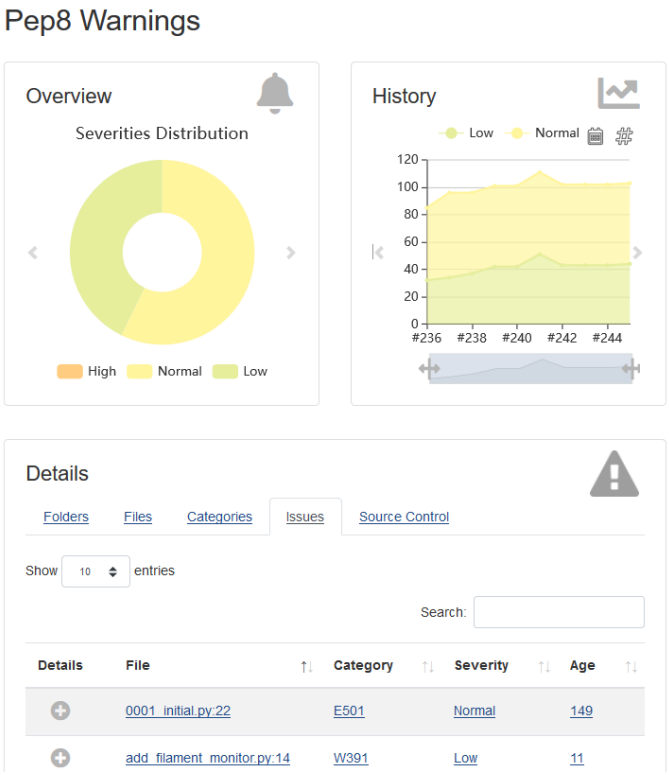


FIGURA 7.7: Projeto Jenkins BEE2B: PEP8 *Warnings*.

7.3 GUI BEE2B

7.3.1 Estrutura de ficheiros

Na figura 7.8 podemos observar como está organizado o código fonte da GUI na pasta **bee2b_gui**. Esta pasta é composta por um ponto de entrada da aplicação (ficheiro **app.py**), responsável pelo arranque da GUI, a pasta **resources**, onde são armazenados ficheiros necessários à aplicação como os ficheiros json de filamentos ou idiomas, e as três pasta **model**, **view** e **controller** utilizadas para organizar logicamente o código fonte da aplicação segundo o padrão MVC.

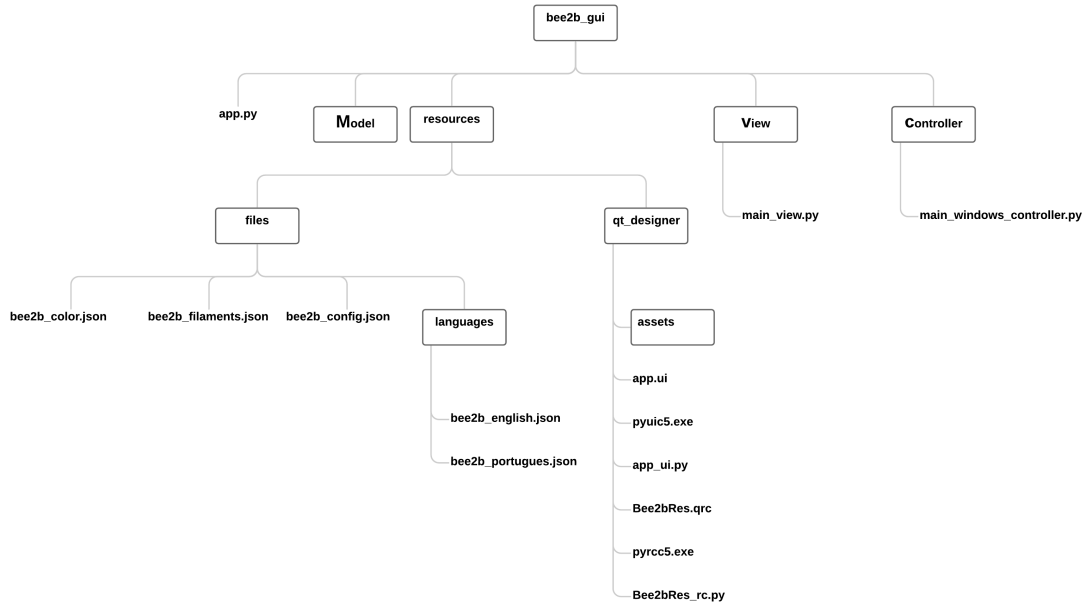


FIGURA 7.8: Estrutura de ficheiros da aplicação GUI BEE2B - pasta **bee2b_gui**.

7.3.1.1 Resources

Esta pasta é composta pelas pastas **files** e **qt_designer**.

Na pasta **files** serão arquivados os ficheiros estáticos necessários à aplicação:

- **bee2b_color.json** - Ficheiro estático com toda a informação das cores dos filamentos disponíveis, podemos ver na figura 7.17 um excerto desse ficheiro com a especificação em rgba de algumas cores para utilizar no texto do código completo do filamento como podemos ver na figura 6.8;

```
"silver": "rgba(192,192,192,255)",
"turquoise": "rgba(64,224,208,255)",
"neon green": "rgba(57,255,20,255)",
"orange": "rgba(255,165,0,255)",
"blanc gris": "rgba(252,252,252,255)",
"zinc yellow": "rgba(241,207,68,255)",
"signal yellow": "rgba(243,173,30,255)",
"bright red orange": "rgba(233,112,56,255)",
```

LISTING 7.17: GUI - excerto do ficheiro bee2b_colors.json

- **bee2b_filaments.json** - Este ficheiro contém todos os filamentos disponíveis para a identificação manual de filamento, como podemos ver na figura 6.34. Podemos ver alguns destes filamentos no excerto do ficheiro em 7.18;

```
"10": "A031 – Neon Green",
"11": "A032 – Orange",
"12": "A101 – Transparent",
"13": "A102 – Blanc Gris",
"14": "A103 – Zinc Yellow",
"15": "A104 – Signal Yellow",
"16": "A105 – Bright Red Orange",
"17": "A106 – Traffic Red",
"18": "A107 – Tomato Red",
```

LISTING 7.18: GUI - excerto do ficheiro bee2b_filaments.json

- **bee2b_config.json** - Ficheiro com todas as configurações da GUI. Como podemos ver em 7.19 neste momento apenas guarda a versão da aplicação e o idioma escolhido;

```
{"version": "1.0", "language": "PT"}
```

LISTING 7.19: GUI - ficheiro bee2b_config.json

- pasta **languages** - Nesta pasta serão guardados todos os ficheiros referentes aos idiomas suportados pela aplicação. Neste momento apenas suporta português (ficheiro **bee2b_portugues.json**) e inglês (ficheiro **bee2b_english.json**). Para adicionar um novo idioma teríamos que adicionar nesta pasta o ficheiro correspondente ao idioma respeitando a estrutura dos outros dois e com todo o texto traduzido para o novo idioma.Na figura 7.20 podemos ver um excerto do **bee2b_english.json**.

```
  "home": {
    "title": "",
    "info": {
      "connected": "Ready to print",
      "disconnected": "Booting system"},
    "bottons": {
      "print": "Print",
      "state": "Print State",
      "add_print": "Add Print"}
  },
  "files": {
    "title": "File Selection",
    "info": "",
    "tabs": {
      "usb": "USB",
      "recents": "Recents"}
  },
  "file_detail": {
    "title": "File Selection",
    "info": "",
    "bottons": {
      "print": "Print"},
    "name": "Name",
    "date": "Date",
    "print_time": "Printing time",
    "user": "User"
```

LISTING 7.20: GUI - ficheiro bee2b_english.json

A pasta **qt_designer** contém todos os ficheiros relativos ao desenvolvimento da GUI com a ferramenta Qt Designer:

- **app.ui** - ficheiro da aplicação Qt Designer, depois de desenhado todo o layout guarda-se este ficheiro para posteriormente converter para python.
- **pyuic5.exe** - utilitário fornecido pelo Qt Designer. Este utilitário é uma interface de linha de comandos que permite converter os ficheiros *.ui do Qt Designer para um ficheiro python.
- **app_ui.py** - Ficheiro python com todo o código correspondente ao interface gráfico. Este ficheiro é gerado automaticamente com o utilitário **pyuic5.exe** e utilizado pelo main_window_controller.
- **Bee2bRes.qrc** - ficheiro xml com todos os recursos utilizados pela **app.ui**, este ficheiro será posteriormente convertido para python.
- **pyrcc5.exe** - utilitário fornecido pelo Qt Designer, utilizado para converter ficheiros *.qrc em ficheiros python. Este utilitário também uma interface de linha de comandos.
- **Bee2bRes_rc.py** - ficheiro python convertido pelo utilitário **pyrcc5.exe** com todos os recursos da aplicação. Este ficheiro é importado pelo ficheiro **app_ui.py**
- pasta **assets** - pasta onde são guardados todos as imagens e ícones utilizados pela aplicação.

7.3.2 Mapeamento do design para o código

Com a conclusão dos diagramas de classes e diagramas de interação para cada caso de uso (secção 5.3), temos detalhes suficientes para iniciar a codificação das classes da camada de domínio (pasta **Model**).

Os artefactos *Unified Modeling Language* (UML) criados durante o *design* da aplicação, diagramas de sequência e de classes, serão usados como *input* do processo de geração de código.

Uma das grandes vantagens da utilização de casos de uso com análise e design orientada a objetos, e com programação orientada a objetos, é que os artefactos criados durante o processo oferecem um roteiro para todo o desenvolvimento da aplicação, desde os requisitos até ao código.

para implementar o design numa linguagem de programação, como o python, é necessário escrever código para:

- definições de classes e interfaces;
- definições de métodos.

Nas próximas secções vamos ver a sua geração em python.

7.3.2.1 Criar definições de classes a partir de diagramas de classes

Com os diagramas de classes temos informação sobre o nome das classes ou interfaces, superclasses, métodos e atributos da classe.

Podemos ver na figura 7.9, como fazemos o mapeamento dos atributos e métodos a partir do diagrama de classes.

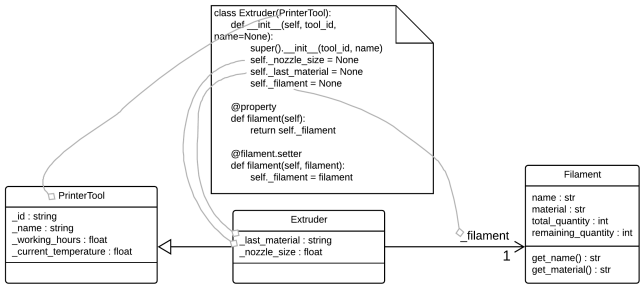


FIGURA 7.9: Classe Extruder em python.

De notar a adição do código para o construtor em python def ___init___(...) que deriva da mensagem create('id00') enviado ao Extruder no diagrama de sequência 5.9. Isto indica que o construtor necessita do parametro tool_id.

7.3.2.2 Criar método a partir de diagramas de sequência

A sequência de mensagens num diagrama de sequência é convertido em métodos das diversas classes. O diagrama parcial *trocar filamento* da figura 7.9 mostra a definição do método *goto_filament*.

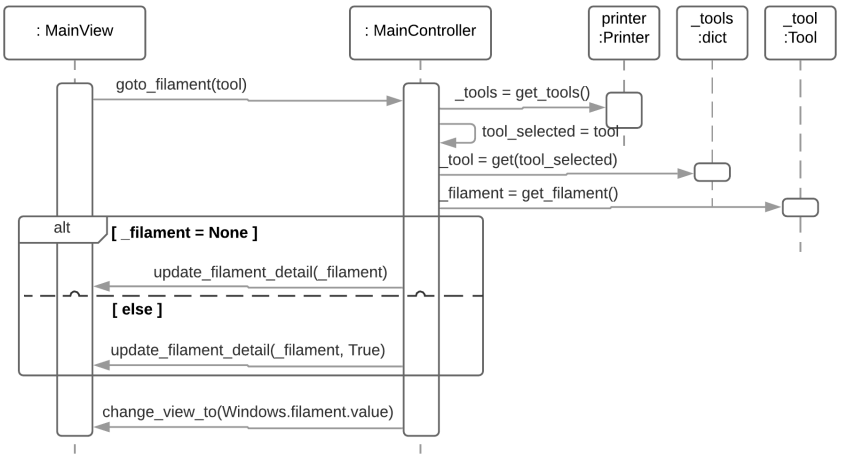


FIGURA 7.10: Diagrama de sequência parcial trocar filamento.

A mensagem *goto_filament* é enviada à instância de *MainController*, assim o método *goto_filament* é definido na classe *MainController*.

A primeira mensagem enviada dentro do método *goto_filament* é a *get_tools*, enviada à instância de *Printer*, para obter um dicionário de ferramentas, depois é registado no argumento *tool*_selected a *tool* passada como parâmetro do método *goto_filament*.

De seguida é enviada a mensagem *get* ao dicionário das ferramentas para obter a ferramenta selecionada e outro *get* à instância de *tool* para obter o filamento instalado nessa ferramenta que é guardado no atributo *_filament*.

Depois de obter o filamento carregado é enviada a mensagem *update_filament_detail* à instância de *main_window* para atualizar a vista de detalhes de filamentos. Por fim é enviada a mensagem change_view_to para alterar a vista a da GUI.

Em resumo, cada mensagem dentro de um método, como mostrado no diagrama 7.9, é mapeado para uma instrução do método em python. Podemos ver o método completo *goto_filament* convertido do diagrama anterior.

```
1     def goto_filament(self, tool):
2         """Change the view to Filament View."""
3         _tools = self.printer.get_tools()
4         self.tool_selected = tool
5         _tool = _tools.get(self.tool_selected)
6         _filament = _tool.filament
7         if _filament is not None:
8             self.main_view.update_filament_detail(_filament)
9         else:
10            self.main_view.update_filament_detail(_filament,
    empty=True)
11        self.main_view.change_view_to(Window.filament.value)
```

LISTING 7.21: Classe *MainController*: método *goto_filament*

7.3.3 Design da Interface gráfica com Qt Designer

- gui - qt designer - resources - workflow

Com a aplicação Qt Designer podemos acelerar o processo de desenho da GUI configurando as várias vistas da aplicação, configuração de componentes e seus posicionamentos.

Para a GUI BEE2B foi utilizado um *QWidget* que corresponde a toda a aplicação. Neste *QWidget* vão estar os seguintes componentes:

- uma *QStackedWidget* para a informação principal das diversas vistas. Cada vista será um *QWidget* dentro deste *QStackedWidget*.
- uma *QFrame* com a barra de menus. Aqui estarão os *QPushButtons Home*, *Settings*, *Maintenance*, *Stop* e *Back*. Também terá uma barra de progresso com o componente *QProgressBar*
- dois QPushButtons para as setas de navegação
- dois QLabel para as zonas informativas título e informação complementar.

7.3.3.1 Estilos dos componentes

Apesar dos estilos dos diversos componentes terem sido feitos e testados no Qt Designer o necessidade destes serem alterados dinamicamente obrigou à criação de uma estratégia alternativa. Assim, foi criado a pasta **style** na camada *model* que armazena todos os estilos dos diversos componentes utilizados.

No código 7.22 podemos ver o método *get_extruder_detail_style* da classe ExtruderStyle, utilizado para obter o estilo do extrusor caso esteja carregado com algum nozzle ou não.

```
@staticmethod
  def get_extruder_detail_style(empty=False):

      if empty:
          return "QPushButton{color: rgb(255,255,255);
  background-color: rgba(0,0,0,0); border-color: rgb(0,0,0)
  ; " \
                 "border-style: solid; border-width: 2px;
  border-radius: 30px; margin: 30px; margin-top: 40px; " \
                 "margin-bottom: 20px;}"
      else:
          return "QPushButton{color: rgb(255,255,255);
  background-color: rgba(0,0,0,255); border-color: black; "
   \
                 "border-style: solid; border-width: 2px;
  border-radius: 30px; margin: 30px; margin-top: 40px; " \
                 "margin-bottom: 20px;}"
```

LISTING 7.22: Classe *ExtruderStyle*: método *get_extruder_detail_style*

7.3.3.2 Fluxo de trabalho

Na figura 7.11 podemos ver o fluxo de trabalho com o Qt Designer. Depois de fazer todas as alterações no interface do Qt Designer guardamos os ficheiros *app.ui* e *Bee2bRes.qrc*, que serão convertidos com as ferramentas pyuic5.exe e pyrcc5.exe respetivamente, nos ficheiros python *app_ui.py* e *Bee2bRes_rc.py*.

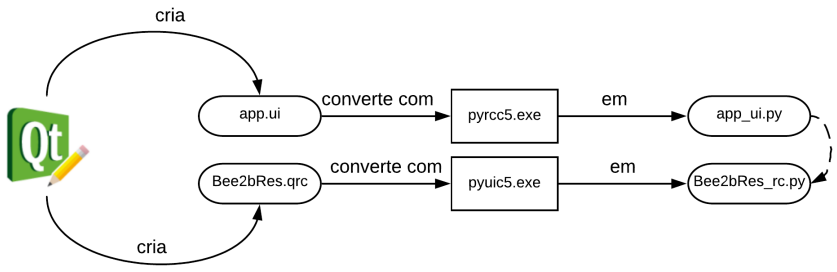


FIGURA 7.11: Fluxo de trabalho do desenvolvimento da GUI com Qt Designer.

Para converter estes ficheiros usamos os comandos apresentados no *script* 7.23.

```
:: Gerar ficheiro python com layout:
pyuic5 --import-from resources.qt_designer app.ui > app_ui.py

:: Gerar ficheiro python com recursos:
pyrcc5 -o Bee2bRes_rc.py Bee2bRes.qrc
```

LISTING 7.23: comandos dos utilitários pyuic5.exe e pyrcc5.exe

O ficheiro *app_ui.py* é utilizado pela Classe MainController, esta classe é a responsável pela gestão das vistas.

7.3.4 Inicio da aplicação

O ponto de entrada da aplicação é o ficheiro bee2b_gui/app.py 7.24.

```
1  import sys
2  from PyQt5.QtWidgets import QApplication
3  from bee2b_gui.model.printer import Printer
4  from bee2b_gui.controller.main_window_controller import
       MainController
5
6
7  class App(QApplication):
8      def __init__(self, sys_argv):
9          super(App, self).__init__(sys_argv)
10         self.printer = Printer()
11         self.main_controller = MainController(self.printer)
12
13
14 if __name__ == "__main__":
15     app = App(sys.argv)
16     sys.exit(app.exec_())
```

LISTING 7.24: ponto de entrada da GUI BEE2B: *app.py*

A classe App têm como superclasse QApplication que é importado do módulo QtWidgets do PyQt5. Esta classe vai ter dois atributos, uma instância da classe *Priter* e outra da classe *MainController*. Esta instância da classe *Priter* é enviada no contrutor da classe *MainController*.

Podemos ver o mapeamento do design para o construtor da classe *Printer* no código 7.25

```
class Printer:
    """Class Printer keep the state of the printer,
    state_monitor and device_manager."""

    printer_state_changed = pyqtSignal(str)

    def __init__(self, token='
    e3d3bc1ff88438a4bb2be7ac88500ef1fee8d3f0'):
        """Initialize the Printer."""
        super().__init__()
        self._printer_server = PrinterAPI()
        self._printer_server.set_token(token)
        self._state_monitor = threading.Thread(target=self.
    _monitor_state, name="printer.monitor_state")
        self._state_monitor.daemon = True
        self._state_monitor.start()
        self._usb_files = UsbFilesAdapter()
        self._recent_files = RecentFilesAdapter()
        self.logger = logging.getLogger(__name__)
        self._tools = {}
        self.create_tools()
        self.colors = Colors()
        self.views = Text()
        self.filaments = {}
        self.load_filaments()
        self.load_config()
        self.views.load_text(self.language)
```

LISTING 7.25: Construtor da classe *Printer*

O construtor da classe *Printer* recebe um *token* como parâmetro para acesso à API. Este construtor vai inicialmente criar uma instância da classe *PrinterAPI*, depois usa o método *set_token* para registar o *token* passado à instância de *Printer* na instância da classe *PrinterAPI*.

Segue-se a criação e iniciação de uma *thread*, *_state_monitor*, responsável por monitorizar o estado do servidor de impressão. O atributo *deamon* da *thread* é colocada a *True* para esta *thread* ser terminada juntamente com a aplicação principal.

Depois são criadas instâncias das classes *UsbFilesAdapter* e *RecentsFilesAdapter*, iniciado o *logger* e o dicionário *_tools* vazio. Segue-se a chamada ao método create_tools que inicia as diversas ferramentas da impressora, nomeadamente, os quatro extrusores.

Por fim, são carregados os ficheiros json com as informações sobre as cores, filamentos disponíveis, configurações e textos a serem utilizados na apresentação da GUI.

Podemos ver o construtor da classe *MainController* no código 7.26.

```
class MainController(QObject):
    """Control the application."""

    progress_value_change = pyqtSignal(float)

    def __init__(self, printer):
        super().__init__()
        self.printer = printer
        self.logger = logging.getLogger(__name__)
        self.main_view = MainView(self.printer, self)
        self.main_view.setWindowFlags(Qt.CustomizeWindowHint
    | Qt.FramelessWindowHint)
        self.main_view.show()
        self.gui_monitor = GuiMonitor(self)
        self.gui_monitor.start()
        self.gui_monitor.new_state_sig.connect(self.
    on_state_change)
        self.heating_canceled = False
        self._file = None
        self._recent_file = False
        self.tool_selected = None
        self.new_filament = None
        self.is_correct_material = True
        self.change_extruder = False
        self.update_filaments()
```

LISTING 7.26: Construtor da classe *MainController*

Este construtor vai criar uma instância da classe MainView passando como parâmetros a instância de *Printer* e a própria instância de *MainController*. É iniciado uma instância de *GuiMonitor*, responsável por atualizar a janela conforme as alterações do estado do servidor de impressão e liga-se o sinal pyqt *new_state_sig* ao método do *MainController on_state_change*. Depois são iniciadas algumas variais auxiliares e atualizadas as informações das ferramentas e filamentos.

```
class MainView(QMainWindow):
    """Main View of the application"""

    # pylint: disable=too-many-instance-attributes
    # Eight is reasonable in this case.

    def __init__(self, printer, main_controller):
        super().__init__()

        self.title = "Main view"
        self._printer = printer
        self._main_controller = main_controller
        self._ui = Ui_mainWindow()
        self._ui.setupUi(self)
        self.hide_frames()
        if sys.platform == "linux":
            self.showFullScreen()
            self.setCursor(Qt.BlankCursor)
        self.setup_home_view()

    # listen for model event signals
        self._main_controller.progress_value_change[float].
    connect(self.set_progress_bar_value)

        self.refreshFiles()
        self.update_filament_list()
```

LISTING 7.27: Construtor da classe *MainView*

7.3.5 Testes

Esta secção apresenta o desenvolvimento dos testes unitários, de aceitação e integração da GUI BEE2B.

Para a realização destes testes foi utilizada a *framework* pytest, para os testes de aceitação foi utilizado o plugin pytest-qt, este plugin fornece fixtures para escrever testes para PyQt.

7.3.5.1 Testes de Integração

Os testes de integração vão testar o acesso à API BEE2B em diversos cenários, como, numa impressão completa, numa pausa e retoma de uma impressão ou no cancelamento de uma impressão.

No excerto do ficheiro **test_printer_api.py** 7.28 vemos que antes de definir o primeiro teste é definida a variável global a todos os testes *TOKEN* que invoca o método estático *request_auth* da classe *PrinterAPI* para obter um token de acesso à API. Depois é definido o teste *test_api_init*, este teste cria uma instância da classe *PrinterAPI* e chama o método set_token dessa instância para inserir o *TOKEN* no cabeçalho dos pedidos à API. Por fim, faz um pedido *get_state* à instância da classe *PrinterAPI* e valida que o resultado é "printer.connected".

```
TOKEN = PrinterAPI.request_auth()


def test_api_init():
    api = PrinterAPI()
    api.set_token(TOKEN)
    excepted_result = 'printer.connected'
    assert excepted_result == api.get_state()
```

Listing 7.28: Ficheiro *test_printer_api.py*: método *test_api_init*

Vamos agora acompanhar parte dos testes deste ficheiro, o primeiro inicia uma impressão e valida a mudança de estado, depois valida que a impressão inicia efetivamente e no último confirma que esta impressão termina e a impressora fica pronta para imprimir novamente.

No teste **test_api_start_print** 7.28, depois de registar o *TOKEN*, é criado uma instância de *File* com uma cena exemplo presente na pasta *resources/files*. Este ficheiro é então enviado no pedido start_print para iniciar a impressão. No fim valida se a impressora iniciou a impressão, ou seja, passou para o estado *Em Aquecimento*.

```
def test_api_start_print():
    api = PrinterAPI()
    api.set_token(TOKEN)
    _file = File(name="BeeX2", path="resources/files/
    BeeX2_BEE2B")
    excepted_result = 'printer.heating'
    api.start_print(_file)
    assert excepted_result == api.get_state()
```

Listing 7.29: Ficheiro *test_printer_api.py*: método *test_api_start_print*

No segundo teste 7.30 é esperado que o resultado do método get_current_job retorne um dicionário com uma chave *response* a *True*, o que quer dizer a impressora está a imprimir. Como, quando uma impressão é iniciada o primeiro estado é o aquecimento, o teste vai ficar num ciclo, onde faz chamadas sucessivas ao método get_state, enquanto a impressora estiver a aquecer.

```
def test_api_get_current_job():
    api = PrinterAPI()
    api.set_token(TOKEN)
    excepted_result = True
    while api.get_state() == 'printer.heating':
        sleep(1)
    data = api.get_current_job()
    assert excepted_result == data["response"]
```

Listing 7.30: Ficheiro *test_printer_api.py*: método *test_api_get_current_job*

Por fim, o último teste 7.31, fica, primeiro num ciclo à espera que a impressão termine, e depois, noutro ciclo à espera que termine o arrefecimento. No final do teste é verificado se o estado da impresso é Ligado, o que representa que a impressora está pronta a imprimir.

```
def test_api_print_done():
    api = PrinterAPI()
    api.set_token(TOKEN)
    while api.get_state() == 'printer.printing':
        sleep(1)
    while api.get_state() == 'printer.coldown':
        sleep(1)
    excepted_result = 'printer.connected'
    assert excepted_result == api.get_state()
```

LISTING 7.31: Ficheiro *test_printer_api.py*: método *test_api_print_done*

7.3.5.2 Testes de Aceitação

Os testes de aceitação vão testar a navegação da GUI durante os vários casos de uso. Com o auxílio do *plugin* pytest-qt, podemos simular a interação com a GUI e validar que esta se comporta como o esperado.

O ficheiro **test_views.py** também inicia com a declaração de um TOKEN para ser utilizado pela aplicação na autenticação com a API.

O teste test_change_filament 7.32 acompanha e valida a navegação do caso de uso UC07: Trocar filamento

De notar que todos os testes de aceitação usam a fixture qtbot, que fornece métodos para simular a interação com o utilizador, como cliques do rato ou o carregar de uma tecla.

Inicialmente é criada uma instância da classe *Printer* com o *TOKEN* para a autenticação da API que é passado como parâmetro na criação da instância da classe *MainController*. Depois, com o método do qtbot addWidget adiciona-se a vista main_view da instância do *MainController*. O método seguinte, *waitForWindowShown*, espera que a janela do main_view seja mostrada.

Para mudar de filamento precisamos de ir ao menu manutenção, para isso usa-se o método *mouseClick* para simular o clique do botão esquerdo do rato (que é igual ao toque no ecrã) no botão *maintenance_btn*, depois valida-se que a nova vista é a de Manutenção.

No menu de manutenção, usamos novamente o método *mouseClick* do *qtbot*, desta vez para simular o clique no botão de detalhes. Agora para iniciar a impressão simula-se o clique no botão de trocar filamento e depois botão sim na confirmação.

Caso algum filamento já esteja carregado, o teste valida que é apresentada a vista de descarregamento do filamento e usa o método *waitUntil* enquanto a vista não se alterar para a vista a identificar automaticamente filamento. Nesta vista e nas vistas A adicionar filamento e A carregar filamento seguintes, o teste vai novamente esperar até que seja concluído cada um dos processos. No final é validado que a vista termina no menu Manutenção.

```
def test_change_filament(qtbot):
    printer = Printer(TOKEN)
    main_controller = MainController(printer)
    qtbot.addWidget(main_controller.main_view)
    qtbot.waitForWindowShown(main_controller.main_view)

    qtbot.mouseClick(main_controller.main_view._ui.maintenance_btn, QtCore.Qt.LeftButton)
    assert main_controller.main_view.get_current_view() == Window.maintenance.value

    qtbot.mouseClick(main_controller.main_view._ui.filament1_btn, QtCore.Qt.LeftButton)
    assert main_controller.main_view.get_current_view() == Window.filament.value

    qtbot.mouseClick(main_controller.main_view._ui.filament_confirm_sim_btn, QtCore.Qt.LeftButton)

    if main_controller.printer.get_tools().get(main_controller.tool_selected).filament is not None:
        assert main_controller.main_view.get_current_view() == Window.unload_filament.value

        def unload_filament_updated():
            assert main_controller.main_view.get_current_view() == Window.new_filament.value
        qtbot.waitUntil(unload_filament_updated, timeout=50000)

    def filament_identify_updated():
        if main_controller.main_view._ui.material_error_frame_new_filament.isVisible():
            qtbot.mouseClick(main_controller.main_view._ui.material_error_yes_btn, QtCore.Qt.LeftButton)
            assert True
        else:
            assert main_controller.main_view.get_current_view() == Window.add_filament.value
    qtbot.waitUntil(filament_identify_updated, timeout=50000)

    def add_filament_updated():
        assert main_controller.main_view.get_current_view() == Window.load_filament.value
    qtbot.waitUntil(add_filament_updated, timeout=50000)

    def load_filament_updated():
        assert main_controller.main_view.get_current_view() == Window.maintenance.value
    qtbot.waitUntil(load_filament_updated, timeout=50000)
```

Listing 7.32: Ficheiro *test_views.py*: método *test_change_filament*

7.4 PrintServer BEE2B

Para o servidor de impressão, é usada uma estrutura básica de uma aplicação Django, com duas *apps* dentro do projeto: *Connectivity* e *FileManager* Como podemos verificar na figura 7.12, existem dois blocos relativos a cada *App* do servidor. Cada uma destas *Apps* fornece

um conjunto de métodos acessíveis através da REST API. A *App Connectivity* é, por sua vez, composta por dois módulos: *Printer* e *Connection*.

Tanto a GUI BEE2B como o BEECura acedem às funcionalidades da impressora através da REST API

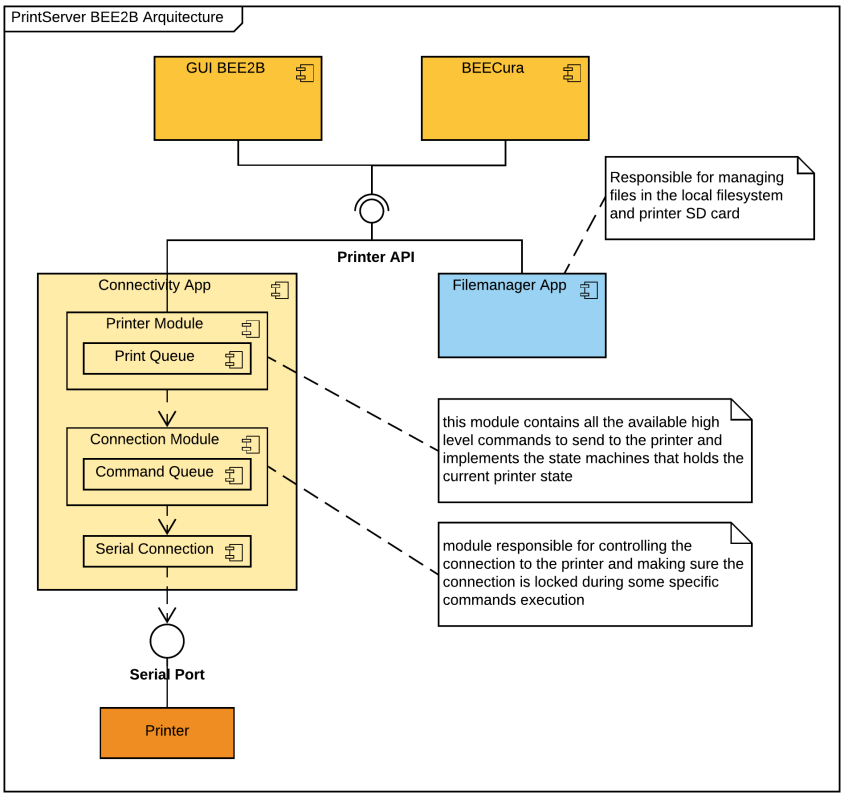


FIGURA 7.12: Arquitectura de PrintServer BEE2B - Vista Lógica.

7.4.1 *App Connectivity*

Esta *app* é composta por três módulos: *connection*, *printer* e *api* (figura fig:server-arquitecture). Apenas vamos detalhar esta *app*, visto ser a mais importante.

- O módulo ***connection*** é responsável por toda a comunicação com a impressora.
- O módulo ***printer*** contém o interface de alto nível com a impressora, com os comandos mais importantes e que por sua vez comunica com a impressora usando os métodos fornecidos pelo módulo *connection*.
- O módulo ***api*** é o interface para o exterior e que por sua vez faz uso dos métodos fornecidos pela interface do módulo *printer* para aceder às funções da impressora.

7.4.1.1 Módulo *connectivity.printer*

O módulo *printer* é composto por várias classes, sendo este o módulo mais importante, vamos agora ver as mais relevantes com mais detalhe.

- *Printer* - Representa a impressora 3D e contém os atributos e métodos necessários para funcionar como interface alto de nível. Esta classe vai usar o módulo *Connection* para comunicar com o impressora 3D e o módulo *FileManager* para gestão de persistência de dados.
- Interface *PrinterState* - Representa o estado da impressora, todos os estados da impressora implementam este interface, estes estados serão detalhados ainda nesta secção na figura 7.14.
- *PrintJob* - Esta classe representa um trabalho de impressão e têm os atributos necessários para guardar a informação do trabalho de impressão como: id, filamento, número de linhas e tempo estimado.
- *PrintQueue* - Usa o módulo python *queue* para implementar a fila de impressão, usada para guardar os trabalhos de impressão pendentes.
- *PrinterTool* - Esta classe abstrata representa uma ferramenta da impressora (extrusor ou mesa) e contém os atributos necessários para guardar informação da mesma. Superclasse da classe *Extruder*. A classe *Printer* contém um dicionário que representa o conjunto de ferramentas disponíveis da impressora e que é preenchido com a informação no momento da ligação com a impressora.
- *Extruder* - Subclasse de *PrinterTool*, representa o extrusor e adiciona aos atributos herdados o diametro do nozzle e o código do último material utilizado.
- *ExtruderState* - Representa o estado do extrusor, todos os estados do extrusor implementam este interface, estes estados serão detalhados ainda nesta secção na figura 7.16.
- *Filament* - Serve para guardar a informação do filamento carregado num extrusor da impressora
- *FilamentState* - Representa o estado do filamento, todos os estados do filamento implementam este interface, estes estados serão detalhados ainda nesta secção na figura 7.18.
- *VirtualPrinter* - Classe que simula a ligação a uma impressora 3D, usada para testes enquanto a impressora 3D não está disponível.

O diagrama de classes da figura 7.13 representa todo o módulo *printer*, os estados da impressora, do extrusor e do filamento não estão detalhados para não tornar o diagrama demasiado complexo.

O diagrama da figura 7.14, mostra os eventos e estados possíveis da impressora, mostra ainda o comportamento da impressora na reação a um evento. Este diagrama de estado mostra todo o ciclo de vida da impressora.

O primeiro estado da impressora é o **Desligado** (*Disconnected*) que só responde ao evento *ligar* (*connect*) passando a impressora para o estado **A Ligar** (*Connecting*), neste estado responde a dois eventos, *sucesso* (*success*) ou erro de ligação (*connection error*), e passa para os estados **Ligada** (*Connected*) e **Erro** (*Error*), respetivamente.

No estado **Erro** a impressora apenas responde ao evento *ligar* voltando a meter a impressora no estado **A Ligar**. A impressora pode, durante o tempo em que está **Ligada**, perder a ligação por algum *erro de ligação*, voltando assim para o estado **Erro**. Existem duas formas da impressora

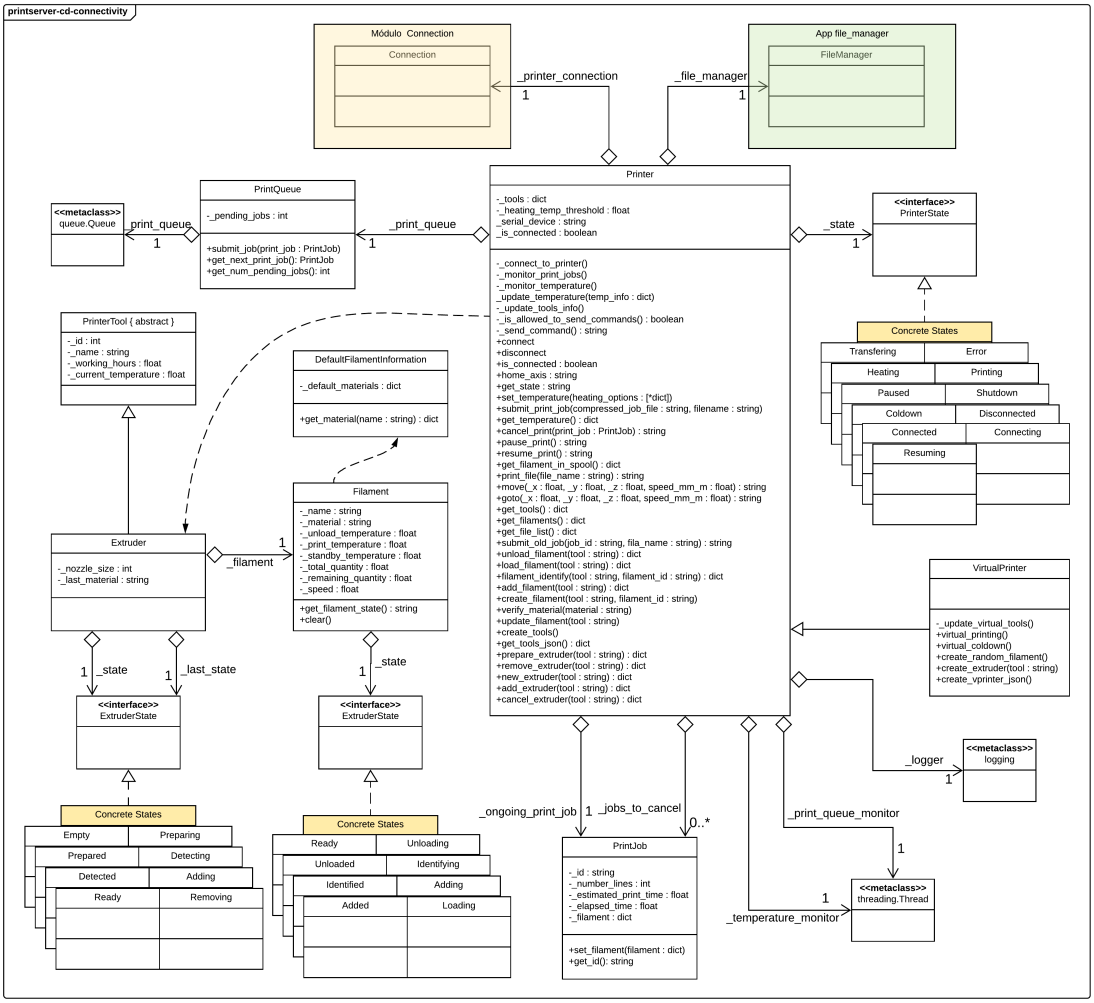


FIGURA 7.13: Diagrama de classes do módulo *printer*.

iniciar o processo de aquecimento, a primeira com o evento *imprime ficheiro* (*print file*) que acontece quando o utilizador manda imprimir através do BEECura ou da GUI BEE2B opção imprimir – aba USB, ficando assim a impressora no estado **A Aquecer e Transferir** (Heating & Transfering), e a segunda com o evento *inciar aquecimento* (start heating) quando o utilizador utiliza a opção imprimir – aba Recentes, passando a impressora para o estado **A Aquecer** (*Heating*). A diferença entre estes dois estados é que quando o utilizador imprime um ficheiro recente já não precisa de transferir o ficheiro pois este já se encontra no armazenamento na impressora. Quando o ficheiro *acaba de ser transferido* (*transfer finished*) a impressora passa para o estado **A Aquecer** pois o processo de aquecimento é invariavelmente mais lento do que o processo de transferência do ficheiro.

Quando o aquecimento estiver terminado *(heating finished)* a impressora começa então a impressão ficando assim no estado **A Imprimir** (*Printing*). A impressão pode ser colocada no estado **Parada** (*Paused*) pelo utilizador a qualquer momento. Quando a impressão está parada o utilizador pode *desligar a impressora* (*shutdown*) ficando no estado **Desligada** (*Shutdown*) ou *retomar a impressão* (*resume*) passando para o estado **A Retomar** (*Resuming*). A impressora pode, quando é desligada com uma impressão a decorrer, retomar a impressão quando é novamente ligada, senão existe nenhuma impressão quando é desligada então

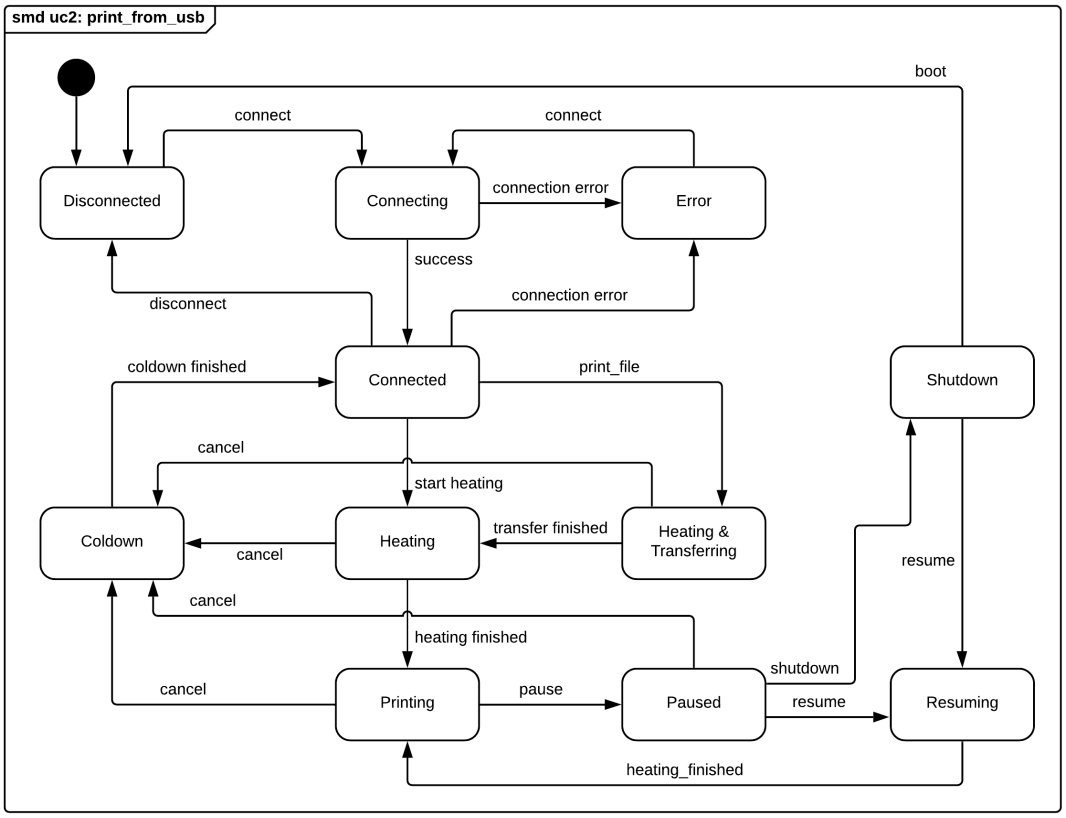


FIGURA 7.14: Diagrama de Estados da Impressora.

o estado da impressora passa para **Desligado** quando for ligada. Quando uma impressão é *retomada* a impressora necessita de aquecer os extrusores novamente antes de continuar a impressão, quando este processo termina a impressora volta ao estado **A Imprimir**.

Os estados **A Aquecer**, **A Aquecer e Transferir**, **A Imprimir** e **Parada** podem ser cancelados passando a impressora para o estado **A Arrefecer** (*Coldown*).

O estado **A Arrefecer** apenas responde ao evento arrefecimento concluído (*coldown finished*) passando a impressora para o estado **Ligada**.

Vamos agora detalhar os estados da impressora com o diagrama de classes da figura 7.15. A Interface *printerState* estabelece os eventos possíveis para os vários estados, cada estado concreto implementa a *PrinterState*. A impressora é composta por um estado, representado por uma instância de de um estado concreto e também implementa cada um dos eventos para posteriormente chamar o mesmo método do próprio estado.

O diagrama de estado seguinte, Figura 7.16, mostra os eventos e estados possíveis do extrusor, mostra ainda o seu comportamento na reação a um evento.

Inicialmente o extrusor começa no estado **Vazio** (*Empty*), neste estado o extrusor apenas responde ao evento **preparar extrusor** (*prepare_extruder*) passando o seu estado para **Em preparação** (*Preparing*).O extrusor durante a preparação responde a dois eventos, **extrusor preparado** (*extruder_prepared*) que mete a impressora no estado **Preparado** (*Prepared*) ou

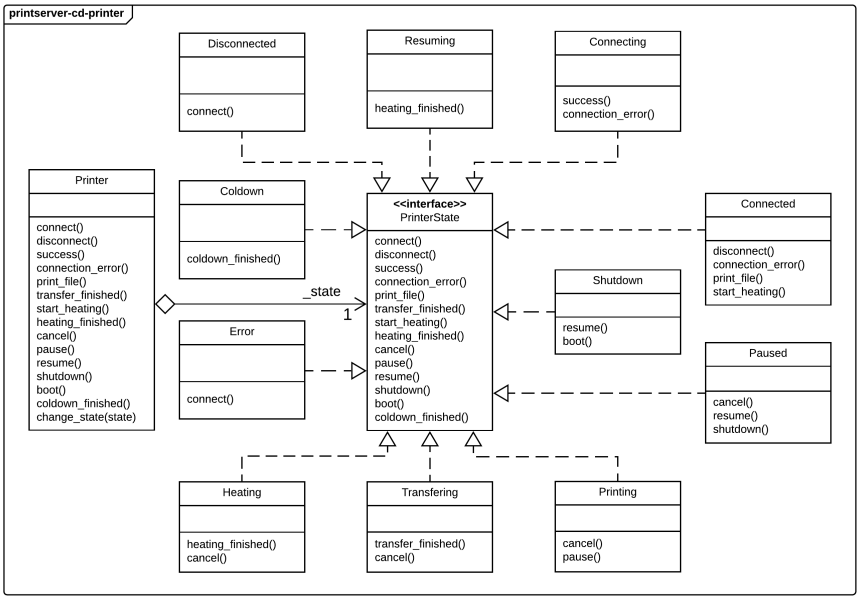


FIGURA 7.15: Diagrama parcial de classes da impressora.

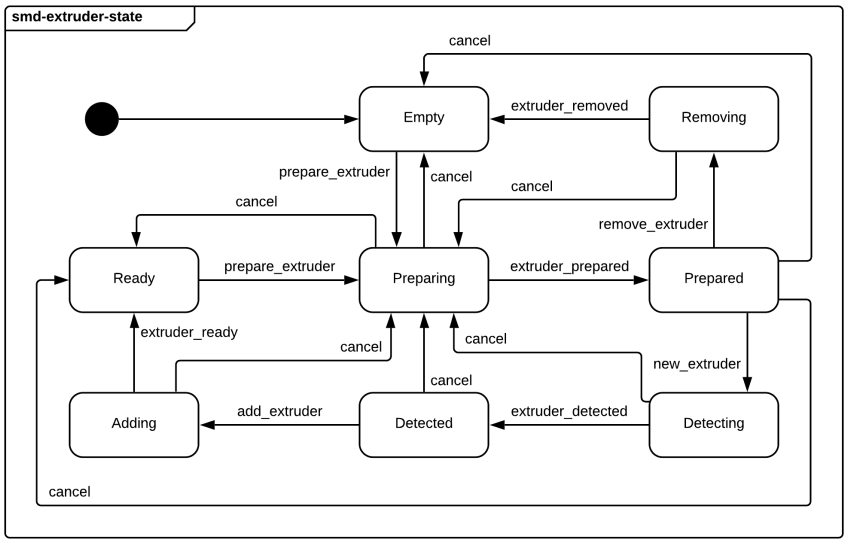


FIGURA 7.16: Diagrama de Estados do Extrusor.

cancelar (*cancel*) que muda o estado do extrusor conforme o seu o último estado, **Vazio** ou **Pronto** (*Ready*).

O extrusor no estado **Preparado** responde a três eventos : **cancelar**, **novo extrusor** (*new_extruder*) e **remover extrusor** (*remove_extruder*). **Cancelar**, tal como no estado **Preparado** faz o extrusor voltar para o estado anterior (**Vazio** e **Pronto**). O método **novo extrusor** muda o extrusor **A Detetar** (*Detecting*) enquanto o método **remover extrusor** passa o extrusor para o estado **A Remover** (*Removing*).

No estado **A Remover**, o extrusor pode ser **cancelado** passando-o para o estado **Em preparação** ou o responde ao evento **extrusor removido** (*extruder_removed*).

Tal como no estado **A Remover**, durante o estado **A Detetar**, o extrusor pode ser **cancelado** mas agora responde adicionalmente ao evento **extrusor detetado** (*extruder_detected*) passando o estado do extrusor para **Detetado** (*Detected*).

Durante o estado **Detetado**, com o evento **cancelar** volta mais uma vez para o **Em preparação** enquanto o evento **adicionar extrusor** (*add_extruder*) passa o extrusor para o estado **A Adicionar** (*Adding*).

No estado **A Adicionar** o extrusor responde mais uma vez a dois eventos, **cancelar** e **extrusor preparado** (*extruder_ready*), mudando o estado do extrusor para **Em preparação** e **Pronta** respetivamente.

No último estado, Pronta, o extrusor apenas responde a um evento, o **preparar extrusor**.

O diagrama da figura 7.17 mostram os detalhes dos estados do extrusor. A Interface *ExtruderState* estabelece os eventos possíveis para os vários estados, cada estado concreto implementa a *ExtruderState*. O extrusor é composta por um estado, representado por uma instância de um estado concreto e também implementa cada um dos eventos para posteriormente chamar o mesmo método do próprio estado.

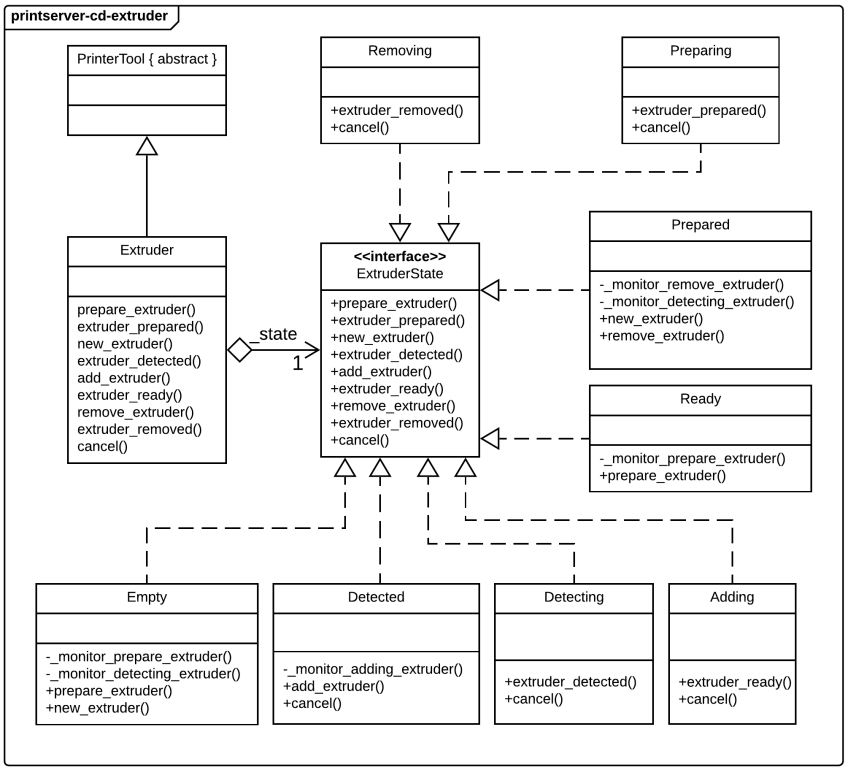


FIGURA 7.17: Diagrama parcial de classes do Extrusor.

O diagrama de estado da figura 7.18, mostra os eventos e estados possíveis do filamento, mostra ainda o seu comportamento na reação aos vários eventos.

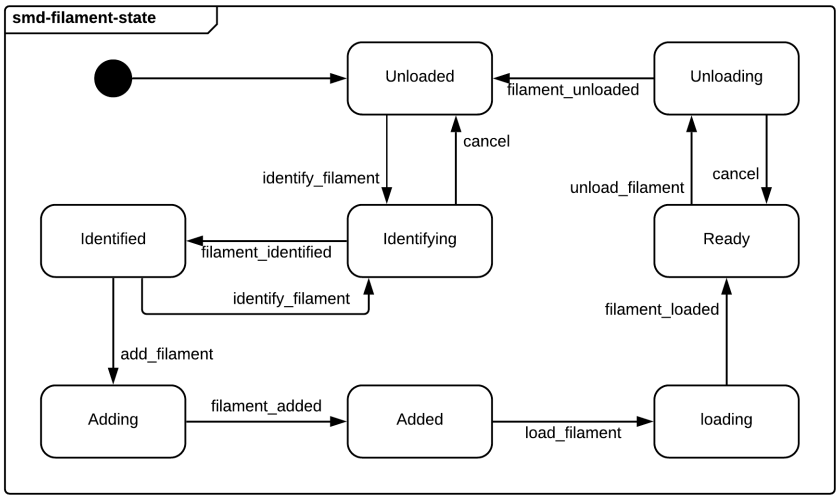


FIGURA 7.18: Diagrama de Estados do Filamento.

O primeiro estado do filamento é o **Descarregado** (*Unloaded*), neste estado o filamento apenas responde ao evento **identificar filamento** (*identify_filament*) passando o estado do filamento para **A Identificar** (*Identifying*).

Quando o filamento está no estado **A Identificar**, aceita os eventos **cancelar** (*cancel*) e **filamento identificado** (*filament_identified*) passando o estado do filamento para **Descarregado** e **Identificado** (*Identified*) respetivamente.

No estado **Identificado**, o evento **adicionar filamento** (*add_filament*) vai mudar o estado do filamento para **A Adicionar** (*Adding*) que por sua vez só responde ao evento **filamento adicionado** (*filament_added*) passando o estado do filamento para **Adicionado** (*Added*).

Para que o filamento mude para o estado **A Carregar** (*Loading*) é necessário o evento **carregar filamento** (*load_filament*) enquanto o filamento está no estado **Adicionado**. No estado **A Carregar**, o filamento responde ao evento **filamento carregado**(*filament_loaded*) e muda o seu estado para **Pronto** (*Ready*).

Quando o filamento está no estado **Pronto** apenas aceita o evento **descarregar filamento** (*unload_filament*) passando o filamento para o estado **A Descarregar** (*Unloading*). Neste estado, o filamento responde a dois eventos: **cancelar** e **filamento descarregado** (*filament_unloaded*) mudando o estado do filamento para **Pronto** e **Descarregado** respetivamente.

O diagrama da figura 7.19 mostram os detalhes dos estados do filamento. A Interface *FilamentState* estabelece os eventos possíveis para os vários estados, cada estado concreto implementa a *FilamentState*. O filamento é composto por um estado, representado por uma instância de um estado concreto e também implementa cada um dos eventos para posteriormente chamar o mesmo método do próprio estado.

7.4.1.2 Módulo *connectivity.connection*

Este módulo é responsável por toda a comunicação de baixo nível com a impressora. O diagrama de classes da figura 7.20 representa todo o módulo *connection*.

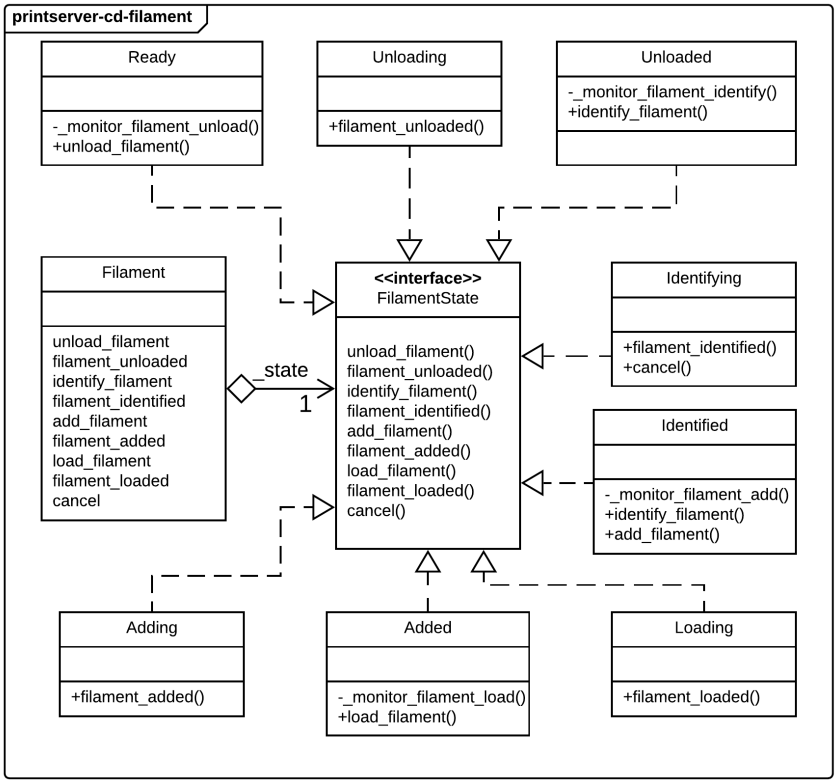


FIGURA 7.19: Diagrama parcial de classes do Filamento.

A classe *Connection* contém o interface de alto nível enquanto a classe *SerialConnection* implementa a comunicação série ao nível da ligação.

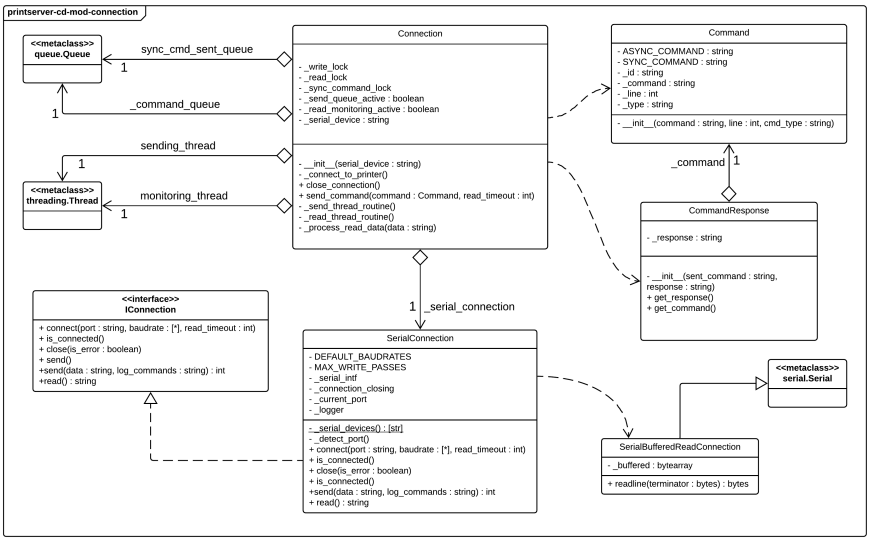


FIGURA 7.20: Diagrama de classes do módulo *printer*.

7.4.1.3 Módulo *connectivity.api*

Este módulo é constituído pelos vários métodos da REST API e será implementado com a *DjangoRESTFramework*, a especificação desta API está detalhada no B.

7.5 BEECura

O Código fonte do BEECura está dividido como mostra a figura 7.21

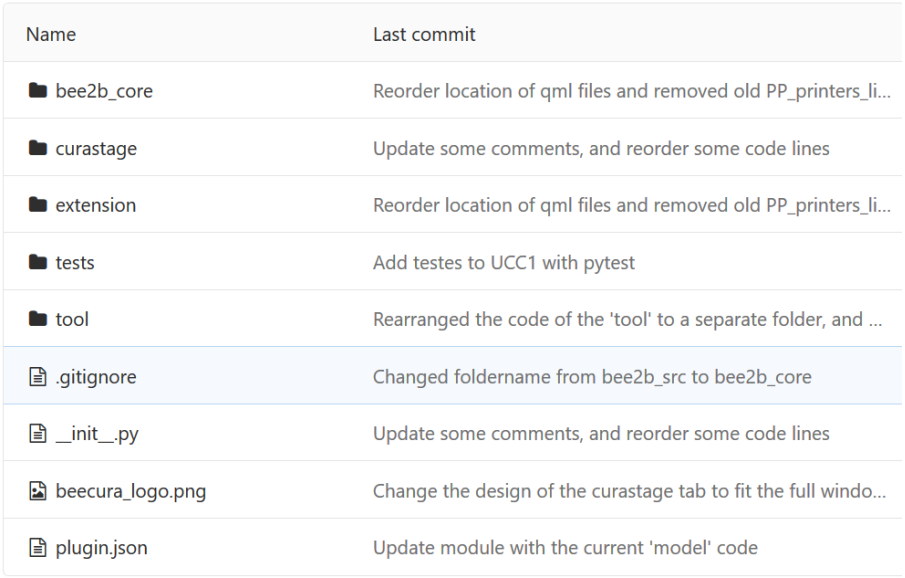


FIGURA 7.21: Organização de ficheiros do código fonte do BEECura.

Raiz do projeto

A pasta bee2b_core contem o código fonte principal, excluindo os comandos da GUI e de interação entre núcleo e GUI. Os comandos adicionais encontram-se na pasta *curastage* (a pasta extension, e a pasta tool são para opções alternativas). Na raiz do *plugin* podemos encontrar os seguintes ficheiros:

- **plugin.json** : Contem a informação com o título do *plugin*, autor, versão e descrição (figura 7.33);

```
{
    "name": "BEE2B-plugin",
    "author": "BEEVERYCREATIVE",
    "version": "1.0",
    "description": "... ",
    "api": "6.0",
    "i18n-catalog": "... "
}
```

LISTING 7.33: BEECura - plugin.json

- **___init___.py** : indica ao cura que o BEECura é composto por três componentes. O principal componente em termos de funcionalidades é o BEE_curastage, os outros dois componentes do plugin são para módulos extra que poderão ser implementadas opcionalmente em futuro(figura 7.34);

```
def register(app):
    return {
        "extension": BEE_extension.BEE_extension(),
        "tool": BEE_tool.BEE_tool(),
        "stage": BEE_curastage.BEE_curastage(app)
    }
```

LISTING 7.34: BEECura - __init__.py

Pasta bee2b_core

Na figura 7.22 podemos ver como está organizada a pasta bee2b_core, que contém o código fonte principal.

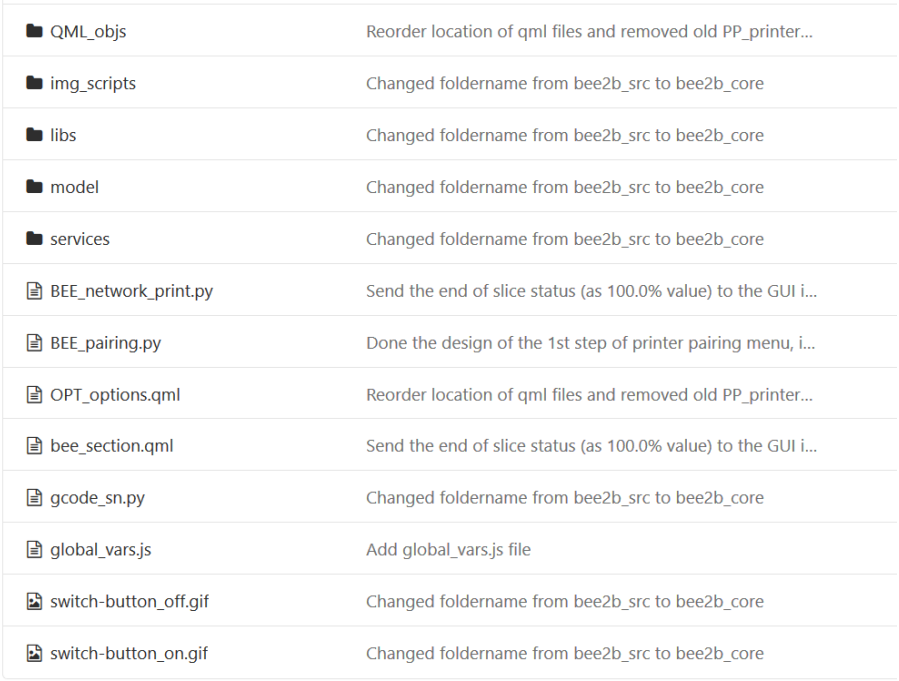


FIGURA 7.22: Organização de ficheiros do código fonte da pasta bee2b_core.

Vamos agora descrever os ficheiros mais importantes:

- **img_scripts/autocrop_img.py** : serve para fazer o crop das imagens que se irão obter com o openscad, que é o programa para ler ficheiros *.STL e fazer uma vista 3D;
- **img_scripts/stl_dimensions.py** : permite encontrar os valores da dimensão em xx, yy, e zz das peças em centímetros;
- **img_scripts/alpha_channel.py** : serve para converter o fundo de uma imagem *.PNG para a cor transparente (assumiu-se que a cor que se deseja remover é a que está na borda no pixel (0, 0), o que tem que ser verdade para as figuras que se gerou pois têm margens).
- **img_scripts/colorize.py** : serve para fazer as imagens com a cor alterada que serão apresentadas na GUI BEE2B quando o objeto é selecionado.
- **gcode_sn.py** : serve para converter o gcode gerado pelo cura no gcode da BEE2B, fazendo as seguintes tarefas:

1. adicionar o id do objeto;
2. converter os comandos de x, y, e z em MESH para: x, e y em MESH e z em NONMESH;
3. adicionar a linha do comando seguinte do próximo MESH.

- **BEE_network_print.py** : contem os comandos principais para obter a informação da ligação da impressora e gerar o *.zip com os ficheiros necessários para iniciar impressão por rede ou por USB. Este ficheiro contem a classe BEE_network_print com as funções:
  - create_screenshot
  - create_cura_gcode
  - create_json_and_pngs
  - create_bee2b_gcode
  - slice_and_save
  - create_zip
  - f_openscad
- **pasta model** : que contém o modelo desenvolvido para a GUI BEE2B, reutilizando assim o código necessário.

7.5.1 UCC1 - Obter a informação da ligação da impressora

Neste caso de uso, descrito em A.3.18.1, precisamos de obter o estado da impressora através do servidor de impressão e mostrá-la na interface gráfica. Vamos também começar a personalizar o BEECura. Para atingir este objetivo foram utilizadas várias abordagens até se chegar à melhor abordagem.

Primeira abordagem

Verificou-se que o botão de slicing aparece nos *stages* prepare e preview, e que cada *stage* é um objeto semi-independente entre si que instância esse botão, contudo apenas queremos manipular o código do stage BEE, e queremos que a informação da impressora esteja sempre visível. O problema é que deste modo seria preciso alterar código de *plugins* externos, o que não é desejável.

Segunda abordagem

Como não queremos alterar o código dos outros stages , tentou-se um botão alternativo, como por exemplo um botão da barra azul do *header* que está lá sempre: o marketplace. Detetou-se então que o botão marketplace é definido no mesmo ficheiro que a imagem do logo : resources/qml/MainWindow/MainWindowHeader.qml.

Então, nesse ficheiro, adicionou-se o código para incluir mais uma imagem de teste, neste caso o logo do cura ampliado, obtendo a figura 7.23, onde se nota que a palavra cura vem do item imagem que se adicionou. Mais uma vez, o problema desta abordagem é que esta mudança é feita no código do cura original e é pretendido apenas trabalhar no nosso *plugin*.

Terceira abordagem

FIGURA 7.23: UCC1 - Obter a informação da ligação da impressora - segunda abordagem.

Nesta abordagem criou-se um novo curastage, e através dele consegui-se manipular tudo, em particular o cabeçalho.

Para que a informação da ligação da impressora apareça sempre escondemos as *tabs* que não mostram a informação de ligação da impressora, e depois criamos todos os elementos e *tabs* necessários dentro do nosso curastage, para que a informação fique sempre visível.

Em primeiro lugar, foi criado um curastage que aparece sob a forma de um tab. Depois, foi necessário indicar ao cura que queremos que o BEE *tab* seja e o que aparece visível no arranque, para tal criou-se as seguintes funções (7.35) no ficheiro curastage/BEE_curastage.py:

```
1     def f(self):
2         self.init_t = datetime.today()
3         self._application = Application.getInstance()
4         self.f_bee2b_curastage()
5
6     def f_bee2b_curastage(self):
7         self._application.getController().setActiveStage("BEE2B-plugin")
8         delta_t = datetime.today()-self.init_t
9         if delta_t.total_seconds()<2:
10            t = Timer(0.001, self.f_bee2b_curastage)
11            t.start()
```

LISTING 7.35: BEECura - Funções f e f_bee2b_curastage

De notar que se inseriu o número de segundos para indicar durante quanto tempo é que esta instrução vai ser executada em ciclo, com o valor pontual de 2, este valor foi obtido empiricamente através de testes em linux e windows, mas há a possibilidade de ter que se mudar para outros processadores, quanto maior for esse valor maior é a fiabilidade do *plugin* aparecer como desejado, por outro lado, quanto menor for, menos pesado fica o processo. Depois foi necessário adicionar os botões todos do cura para meter no nosso tab, para tal, foram feitos dois novos ficheiros qml, o OPT_options.qml e QML_objs/QML_views_selector.qml inspirados no código presente nos *stages* prepare e preview.

7.5.2 UCC2 - Iniciar uma impressão por rede

Para iniciar um impressão por rede , depois de preparar-mos a cena que desejamos imprimir, clicamos em slice & print, isto irá gerar um zip com os conteúdos pretendidos e efetuado o pedido start_print à API do servidor de impressão, enviando este zip criado no pedido.

Os ficheiros python desenvolvidos para este caso de uso foram:

- bee2b_core/BEE_network_print.py
- bee2b_core/img_scripts/autocrop_img.py
- bee2b_core/img_scripts/stl_dimensions.py
- bee2b_core/img_scripts/alpha_channel.py
- bee2b_core/img_scripts/colorize.py (opcional).

O ficheiro principal do processamento da impressão por rede é o BEE_network_print.py, a partir do qual se chamam os outros quando apropriado. Um objeto da classe BEE_network_print.py será inicializado durante o funcionamento do *plugin*, a partir de outros ficheiros:

- **BEE_curastage** : tem várias funções úteis nomeadamente a função get_printer_state (figura 7.36)

```
1     def get_printer_state(self):
2         try:
3             if (self.model is None):
4                 try:
5                     from ..bee2b_core.model import printer_api as
    model_bee2b
6                     Logger.log("w", "imported model.printer_api
    successfully!")
7                 except BaseException as e:
8                     Logger.log("w", "could not import model.printer_api
    !")
9                     return str(e)
10                self.model = model_bee2b.PrinterAPI()
11                self.client_id = self.my_mac_address()
12                try:
13                    self.client_id += "_-_"+platform.node()
14                except:
15                    print("ERROR: could not get platform.node().")
16                print(self.client_id)
17            if str(self.model.response).lower()=="none":
18                self.token = self.model.request_auth(self.client_id)
19                self.model.set_token(self.token)
20
21            result = self.model.get_state()
22            print("PRINTER STATUS:", result)
23            return "PRINTER STATUS: "+str(result).lower()
24        except BaseException as e:
25            return str(e)
```

LISTING 7.36: BEE_curastage - Função get_printer_state

- **BEE_service.py** : faz a ligação entre os ficheiros qml e o ficheiro BEE_curastage.py, onde por exemplo se definiu a função call (figura 7.37), cujo resultado é chamado

no ficheiro bee_section.qml, através de um temporizador. São também definidos os handlers para o evento clique dos botões slice & save e print (figura 7.38).

```
1     @pyqtSlot()
2     def call(self):
3         self._printer_status = BEE_service._bee_stage.
      get_printer_state()
4         return self._printer_status
```

Listing 7.37: BEE_service - Função call

```
1     @pyqtSlot()
2     def on_slice_and_save(self):
3         self._bee_stage.on_slice_and_save(self)
4
5     @pyqtSlot()
6     def on_print(self):
7         BEE_service._bee_stage.on_print()
```

Listing 7.38: BEE_service - Funções on_slice_and_save e on_print

O objeto _bee_stage (instância da classe BEE_curastage) contém os métodos necessários ao processamento dos dados em python, nomeadamente, no caso do UCC2, incluiu-se a chamada à função slice_and_save a uma instância da classe BEE_network_print

```
1             self._network_print.slice_and_save(self.job_name, service)
```

Listing 7.39: BEE_curastage - Função parcial on_slice_and_save

métodos da classe BEE_network_print:

- **create_screenshot** : este método permite tirar um foto do ecrã inteiro utilizando o módulo pyscreenshot (figura 7.40);

```
1     def create_screenshot(self, job_name):
2     img = pyscreenshot.grab()
3     img.save(BEE2B_folder+job_name+".png")
```

Listing 7.40: BEE_network_print - Função create_screenshot

- **create_cura_gcode**: este método gera o gcode correspondente ao último slice efetuado, e utiliza um objeto do tipo GcodeWriter que foi instanciado no arranque do cura na criação da classe BEE_network_print (figura 7.41). O *slicing* é chamado na função slice_and_save() com *self.engine_backend.slice()*.

  Durante o desenvolvimento deste método foi encontrado um problema que posteriormente foi corrigido. O slice através desta função chamada pelo nosso *plugin*, e definida no cura, era lançado em background em modo paralelo e assim o gcode não era preenchido no primeiro clique, mas apenas nos seguintes após o slice ter decorrido. Resolveu-se este problema usando uma *thread* adicional que espera enquanto o slice é efetuado, verificando o estado da variável self.engine_backend._slicing. O gcode só será gerado quando esta variável estiver em falso.

```
1     def create_cura_gcode(self, job_name):
2         fname_gcode_cura = BEE2B_folder+job_name+"_1st.gcode"
3         f = open(fname_gcode_cura, "w+")
4         self.gcode_writer.write(f, "dummy_arg")
5         f.close()
```

LISTING 7.41: BEE_network_print - Função create_cura_gcode

- **create_json_and_pngs** : este método é muito importante pois percorre todos os objetos da cena, obtendo as suas propriedades e registando-as num ficheiro json que será incluído no zip enviado ao servidor de impressão, como podemos ver na figura 7.24. Para cada objeto na cena, vai ainda encontrar os seus ficheiros stl e com recurso à ferramenta openscad gerar as imagens necessárias à GUI BEE2B para apresentação da cena 2D vista de topo.

  Para gerar o json e as imagens eram necessárias várias instruções comuns, nomeadamente na seleção das propriedades de cada objeto da cena e respetivo tratamento, assim, optou-se por juntá-las no mesmo método.

  Para a geração do ficheiro json (7.42) foi necessário obter:

  – o tempo estimado de impressão;

  – a quantidade de filamento;

  – a informação de cada objeto.

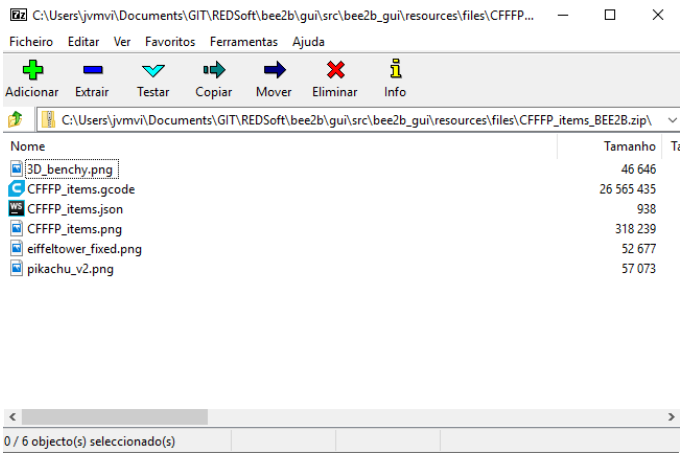


FIGURA 7.24: UCC2 - Iniciar uma impressão por rede : Zip gerado pelo botão slice & save.

```
{
    "name": "CFFFP_items",
    "path": "",
    "job_id": "",
    "job_date": "2019-07-23T12:46:46",
    "filament": "PLA",
    "filament_quantity": 3.97241,
    "printing_time": 29698,
    "objects": [
        {
            "name": "pikachu_v2 - 00",
            "id": "00",
            "model": "pikachu_v2",
            "x": 175.0,
            "y": 150.0,
            "w": 38.53448906414749,
            "h": 73.92774005318675
        },
        {
            "name": "eiffeltower_fixed - 01",
            "id": "01",
            "model": "eiffeltower_fixed",
            "x": 101.0,
            "y": 146.0,
            "w": 54.66200256347656,
            "h": 54.66200256347656
        },
        {
            "name": "3D_benchy - 02",
            "id": "02",
            "model": "3D_benchy",
            "x": 235.0,
            "y": 108.0,
            "w": 60.000999450683594,
            "h": 31.003999710083008
        }
    ]
}
```

LISTING 7.42: UCC2 - Iniciar uma impressão por rede : ficheiro json gerado pela função create_json_and_pngs

Para obter a informação dos elementos todos da cena foi preciso ir buscar a informação aos respetivos elementos como podemos ver no código 7.43, e depois processam-se os dados para cada elemento el obtemos a informação que podemos ver no json 7.42 entre as linhas dez e dezoito.

```
        ## vai percorrer a cena do cura a procura da informacao relativa a cada objeto:
        objects_lst = []

        print(">>>>>>> APPLICATION: "+str(dir(self._application)))
        print(self._application.recentFiles)
        for j in self._application.recentFiles:
            print(j.url())

        scene = self._application.getController().getScene()
        print(">>>>>>> SCENE: "+str(dir(scene)))

        obj = scene.getRoot()
        print(">>>>>>> OBJ: "+str(obj))
        print(dir(obj))

        ## cria a pasta onde vai guardar os *.scad e *.png dos objectos da cena:
        if not os.path.exists(BEE2B_folder+"pngs_&_scads"):
            os.mkdir(BEE2B_folder+"pngs_&_scads")
        os.chdir(BEE2B_folder+"pngs_&_scads")

        all_chidren = obj.getAllChildren()
        print(">>>>>>> CHILDRENS: ")

        models_lst = []          ##objecto que vai conter uma lista dos valores para o campo "model", do json.


        var_i=0
        if all_chidren!=None:
            for el in all_chidren:
                print("\t"+str(el))
                print("\t\t"+str(type(el)))
                if str(type(el))=="<class 'cura.Scene.CuraSceneNode.CuraSceneNode'>":
                    print(dir(el))
                    print(el.getPosition())
                    print(el.getOrientation())
                    print(el.getScale())
                    print(el.getName())
```

LISTING 7.43: BEE_network_print - Função parcial create_json_and_pngs

O método create_json_and_pngs vai ainda gerar as várias imagens para incluir no zip, durante o processo são gerados também alguns ficheiros intermédios auxiliares:

- pngs_&_scads/eiffeltower_fixed - 00.scad : ficheiro scad que chama um ficheiro STL e a partir do qual se faz a perspetiva de topo;
- pngs_&_scads/eiffeltower_fixed - 00.png : perspetiva de topo inicialmente obtida;
- pngs_&_scads/eiffeltower_fixed - 00_-_cropped.png : perspetiva de topo depois de cortar as margens;
- eiffeltower_fixed.png : ficheiro final com transparência, em modo normal

– eiffeltower_fixed_hover.png : ficheiro final com a cor de *hover* para indicar modo "elemento seleccionado".

O Método f_openscad (7.44) é crucial para a geração das imagens. Para além disso são utilizados as funções auxiliares autocrop_img, convert_to_transparent e colorize para gerar as imagens finais.

```
    def f_openscad(self, arg_STL, filepath_i_scad, filepath_i_png,
view_type="3d", colormap_hover=False):
        f_scad = open(filepath_i_scad, "w")

        max_length=max(self.result_json["x_length"], self.result_json["
y_length"])
        scad_xx = self.result_json["x_length"]/max_length
        scad_yy = self.result_json["y_length"]/max_length

        if view_type=="2d":
            ## pega no objeto e centra-o na cena do openscad com o
comando <<translate>>, usando os valores devolvidos pelo metodo de
stl_dimensions.py;
            ## faz uma escala ao objeto de modo que este fique incluido
 num cubo de tamanho 1x1x1, com uma escala proporcional a original:
 a dimensao maior e 1, e as restantes sao o valor que mantem a
proporcinalidade do objeto;
            ## este cubo e util porque assim podemos exportar o objeto
com uma dada camara (escolhida para integrar o cubo), sabendo que o
 objeto fica todo visivel.
            f_scad.write('''resize(['''+str(scad_xx)+''', '''+str(
scad_yy)+''', 1.00])
{translate(['''+str(-self.result_json["x_med_stl"])+''', '''+str(-
self.result_json["y_med_stl"])+''', 0])
{import("'''+arg_STL+'''");}}$vpt=[0,0,0];$vpr=[0,0,0];''')

        else:
            f_scad.write('''
vector_xyz = ['''+str(self.result_json["x_length"])+''', '''+str(
self.result_json["y_length"])+''', '''+str(self.result_json["
z_length"])+''''];

length_v = max(vector_xyz);

resize(vector_xyz/length_v)
{translate(['''+str(-self.result_json["x_med_stl"])+''', '''+str(-
self.result_json["y_med_stl"])+''', 0])
{import("'''+arg_STL+'''");}}
$vpt = [0.5, 2, 0];        //coordenadas do centro da janela da
camara
$vpr = [78.2, 0, 352.1];   //rotacao da camara
$vpd = 4;                  //distancia em profundidade
''')
```

Listing 7.44: BEE_network_print - Método f_openscad 1/2

```
1         f_scad.close()
2         ## correr o ficheiro *.scad para a imagem actual c/ um comando
      tal como:
3         ## openscad BEE.scad -o BEE.png --projection orthogonal --
      camera=0,0,0,0,0,0,2.7 --render
4         if ('linux' in sys.platform) or ('darwin' in sys.platform):
5             openscad_cmd = "openscad"
6         elif 'win' in sys.platform:
7             openscad_cmd = "C:\Program Files\OpenSCAD\openscad.com"
8
9         if view_type=="2d":
10            lst_cmd = [openscad_cmd, filepath_i_scad, "-o",
      filepath_i_png, "--projection", "orthogonal", "--camera
      =0,0,0,0,0,0,2.7", "--render"]
11        else:
12            lst_cmd = [openscad_cmd, filepath_i_scad, "-o",
      filepath_i_png, "--projection", "orthogonal", "--camera
      =0.5,2,0,78.2,0,352.1,4", "--render"]
13
14        lst_cmd_hover = lst_cmd+["--colorscheme", "Tomorrow"]
15
16        if ('linux' in sys.platform) or ('darwin' in sys.platform):
17            if not colormap_hover:
18                subprocess.call(lst_cmd)
19            else:
20                subprocess.call(lst_cmd_hover)
21        elif 'win' in sys.platform:
22            if not colormap_hover:
23                subprocess.call(lst_cmd, shell=True)    ##the option "
      shell=true" hides the cmd prompt of openscad - @WINDOWS.
24            else:
25                subprocess.call(lst_cmd_hover, shell=True)
```

LISTING 7.45: BEE_network_print - Método f_openscad 2/2

Inicialmente para gerar as imagens com cor *default* e cor de *hover*, foi utilizado o openscad com o valor de colorscheme *default* e com um valor diferente pelo comando –colorscheme. Esta abordagem fez com que o rendering fosse feito por duas vezes independentes tornando assim o componente default duas vezes mais lento. Para melhorar a performance, foi criado o script de colorize que converte a imagem com cor *default* para a cor *hover* sem repetir o *rendering*, para tal faz-se uma leitura da matriz da imagem com os dados *(Red, Green, Blue)* (RGB) e tratam-se os conteúdos em modo *(Hue, Saturation, Value)* (HSV).

- **create_bee2b_gcode** : este método cria três gcodes, um criado pelo cura (7.46), uma versão temporária com os ids dos objetos (7.47) e a versão final para a impressora BEE2B, com a adição da linha com *";MESH seguinte"* e separação de Z e XY (7.48).

  O código relativo à geração dos gcodes da BEE2B foi implementado no ficheiro gcode_sn.py, que é incluído pelo ficheiro BEE_Network_print.py

```
;;; original do cura:
;LAYER:153
;MESH:eiffeltower_fixed.stl
G0 X168.771 Y143.771 Z15.6
...
```

LISTING 7.46: UCC2 : gcode original gerado pelo cura

```
;;; obtido com o metodo gcode_sn():
;LAYER:153
;MESH:eiffeltower_fixed.stl (ii)
Mxxx Iii
G0 X168.771 Y143.771 Z15.6
...
```

LISTING 7.47: UCC2 : gcode com ids dos objetos

```
;;; obtido com o metodo mesh_nonmesh_fix():
;LAYER:153
;MESH:nonmesh
G0 Z15.6
;MESH:eiffeltower_fixed.stl (... )
Mxxx Iii L<linha c/ ;MESH seguinte>
G0 X168.771 Y143.771
...
```

LISTING 7.48: UCC2 : gcode final

- **slice_and_save** : esta função vai chamar o *slice* do cura, e o método create_zip que escreve o ficheiro zip. Estes dois processos são chamados sequencialmente para que o segundo método use o ficheiro gcode gerado no primeiro método como *input*.
- **create_zip** : esta função vai criar um ficheiro zip com todos os ficheiros gerados anteriormente. Este método recorre à biblioteca python zipfile.

7.5.3 UCC3 - Emparelhar o PC com uma impressora BEE2B

Neste caso de uso foi desenvolvido o ficheiro bee2b_core/QML_objs/PP.qml que corresponde à parte gráfica. Este ficheiro PP.qml (abreviatura de *print pairing*), é importado peloficheiro bee2b_core/OPT_options.qml, que contem os três tabs: prepare, preview e monitor. Por sua vez este último é importado pelo ficheiro bee2b_core/bee_section.qml, que contem informação geral e o campo de informação do estado da impressora que está sempre visível no canto superior esquerdo.

Para este use case, no momento de redação deste documento, ainda só foi desenvolvido o aspeto gráfico da GUI, faltando a comunicação com o servidor para poder efetuar o emparelhamento.

7.5.4 UCC4 - Gerir fila de impressão

Foi feito o ficheiro bee2b_core/QML_objs/MPQ.qml para o desenvolvimento da monitorização da fila de impressão, a introdução hierárquica deste ficheiro é análoga à do ficheiro do UCC3. – Para este use case, tal como no UCC3, no momento de redação deste documento,

ainda só foi desenvolvido o aspeto gráfico da GUI, faltando a comunicação com o servidor para poder mostrar e alterar os dados relativos à fila de impressão.

Capítulo 8

Avaliação da Solução

Neste capítulo será descrito o método de avaliação da solução proposta. É feita uma descrição das grandezas a utilizar na avaliação da solução, da hipótese a testar, da metodologia de avaliação que será utilizada e de como a hipótese a testar será testada.

8.1 Variáveis Para a Avaliação

Para a avaliação da solução, definiram-se x variáveis, sendo elas:

- **Usabilidade da GUI BEE2B:** pretende-se aferir se a aplicação é de fácil utilização, com uma interface apelativa e intuitiva.
- ***Uptime* da impressora 3D BEE2B:** pretende-se verificar se uma GUI com elevada usabilidade se traduz num aumento de *uptime* da impressora 3D BEE2B.
- ***Downtime* da impressora 3D BEE2B:** pretende-se verificar se uma GUI com elevada usabilidade se traduz numa diminuição de *downtime* da impressora 3D BEE2B.
- **Tempo de impressão:** pretende-se comprovar que remover um objeto da cena 3D que está a imprimir se traduz numa diminuição do tempo de impressão.
- **Material consumido:** pretende-se comprovar que remover um objeto da cena 3D que está a imprimir se traduz numa diminuição do material gasto durante a impressão.

8.2 Hipótese a Testar

A hipótese a testar irá permitir saber se a solução cumpriu ou não o objetivo.

A solução tinha como principal objetivo melhorar a usabilidade das impressoras 3D, proporcionando assim um aumento de *uptime* e diminuição de *downtime*. Tinha ainda o objetivo de implementar funcionalidades que permitissem poupar tempo e material no caso de falha parcial da impressão.

Assim para a solução temos as seguintes hipóteses a testar:

Hipótese nula (H0): O uso da GUI BEE2B com elevada usabilidade não conduziu a um aumento de *uptime* da impressora

H0: *uptime* de impressora 3D sem GUI = *uptime* de impressora 3D com GUI

Hipótese alternativa (H1): O uso da GUI BEE2B com elevada usabilidade conduziu a um aumento de *uptime* da impressora

H1: *uptime* de impressora 3D sem GUI $<$ *uptime* de impressora 3D com uptime

8.3 Metodologia de avaliação

Para avaliação da solução desenvolvida foram utilizados dois métodos de avaliação, o *Quantitative Evaluation Framework* (QEF) e questionários aos utilizadores. Os resultados dos questionários vão servir como input para o QEF, que fará a avaliação global da solução.

Serão feitos no mínimo duas sessões de testes, os testes alpha e beta, mais sessões poderão ser necessárias caso se verifique a necessidade de continuar a melhorar alguns aspetos da aplicação.

Os testes alpha poderão ser feitos ainda sem ligação à impressora, sem todos os casos de uso implementados e com um layout que ainda poderá sofrer alterações.

Por outro lado, nos testes beta, é expetável que os testes já sejam feitos na impressora 3D, com todos os casos de uso estejam funcionais e o layout da GUI final ou muito próximo.

O anexo B é uma cópia do questionário utilizado nos testes alpha.

Na secção 8.4 vamos ver com mais detalhe esta framework de avaliação de qualidade de software, o QEF

8.4 QEF

O QEF, é um método que tem como objetivo " medir quantitativamente a qualidade de um determinado conteúdo digital no contexto de um sistema educativo " (Escudeiro 2008). Este método consiste numa avaliação quantitativa da qualidade do software, que é constituída por um espaço a três dimensões: Funcionalidade, Adaptabilidade e Eficiência. Cada dimensão é constituída por um conjunto de Fatores e cada Fator por um conjunto de requisitos. Assim, com este método, podemos obter o grau de desempenho do sistema em qualquer fase do desenvolvimento.

Cada requisito tem um peso específico dentro do fator em que está inserido, podendo variar entre dois e dez.

Para efetuar a avaliação devemos atribuir uma percentagem de conclusão a cada requisito, para definir essa percentagem deve-se verificar os critérios de avaliação desse requisito. Depois de preenchidas as percentagens de conclusão de todos os requisitos, obtemos a percentagem de conclusão dos objetivos do nosso sistema.

Foi utilizado este método porque é uma poderosa ferramenta de controlo de qualidade sendo assim completamente adequado ao projeto.

Se seguida será apresentado o QEF da solução proposta, os seus fatores, requisitos e métrica de avaliação.

8.4.1 Fatores

Este capítulo descreve os fatores utilizados no QEF deste projeto agrupados por dimensões. Inicialmente é apresentado a dimensão Funcionalidade (8.4.1.1), seguida da dimensão Adaptabilidade (8.4.1.2) e por fim a dimensão Eficiência (8.4.1.3).

8.4.1.1 Funcionalidade

A dimensão funcionalidade é constituída por três fatores que serão agora descritos.

1. **Funcional (Casos de Uso):** Refere-se à conclusão dos vários casos de uso definidos nos requisitos do projeto.
2. **PrintServer API:** Refere-se à implementação dos métodos da REST API do PrintServer necessários à conclusão dos casos de uso descritos no fator anterior (Funcional).
3. **BEECura *Plugins*:** Refere-se à implementação dos *plugins* e personalizações necessárias para à conclusão dos Use Cases descritos no primeiro fator descrito neste subcapítulo (Funcional).

8.4.1.2 Adaptabilidade

A dimensão adaptabilidade é constituída também por três fatores.

1. **Versatilidade:** Facilidade do software em se adaptar a diferentes condições de utilização.
2. **Usabilidade:** Esforço necessário para a utilização do sistema, baseado num conjunto de implicações e de condições do utilizador.
3. **Manutenção:** Refere-se ao esforço necessário para a realização de alterações específicas, no produto de software.

8.4.1.3 Eficiência

Os seguintes três fatores formam a dimensão eficiência.

1. **Navegação:** Refere-se à facilidade e qualidade da navegação da GUI BEE2B.
2. **Qualidade do Conteúdo:** Qualidade do conteúdo apresentado na GUI BEE2B.
3. **Suporte:** Qualidade do serviço de suporte ao utilizador.

8.4.2 Requisitos e métricas de avaliação

Este subcapítulo descreve cada um dos requisitos e respetivas métricas de avaliação. Inicialmente são apresentados os requisitos dos fatores da dimensão Funcionalidade, seguido dos requisitos dos fatores da dimensão Adaptabilidade e por fim os da dimensão Eficiência.

8.4.2.1 Funcionalidade - Funcional

No fator Funcional da dimensão Funcionalidade são usados os casos de uso registados nos requisitos da GUI BEE2B descritos no subcapítulo A.3.

1. **Requisito:** FF01 - Iniciar impressão a partir do PC (BEECura) (UC01 - A.3.1);

   Métrica de avaliação: O utilizador pode iniciar a impressão a partir do PC (BEECura);
2. **Requisito:** FF02 - Iniciar impressão a partir de uma drive USB (UC02 - A.3.2);

   Métrica de avaliação: O utilizador pode iniciar a impressão a partir de uma drive USB ligada à impressora BEE2B (GUI BEE2B);

3. **Requisito:** FF03 - Iniciar impressão de um ficheiro recente (UC03 - A.3.3);

   Métrica de avaliação: O utilizador pode iniciar a impressão de um ficheiro recente presente no cartão SD da impressora BEE2B (GUI BEE2B);

4. **Requisito:** FF04 - Apresentar uma cena 3D (renderização de cena 3D/2D)

   Métrica de avaliação: O utilizador pode iniciar uma impressão e visualizar a renderização da impressão na GUI BEE2B (UC04 - A.3.4);

5. **Requisito:** FF06 - Remover objeto da impressão (UC06 - A.3.6);

   Métrica de avaliação: O utilizador pode remover um objeto da impressão atual, a impressão continua normalmente e a renderização da cena 3D/2D é atualizada;

6. **Requisito:** FF07 - Substituir Filamento (UC07 - A.3.7);

   Métrica de avaliação: O utilizador pode substituir o filamento de um dos extrusores na GUI BEE2B através de um assistente;

7. **Requisito:** FF08 - Substituir Extrusor (UC08 - A.3.8);

   Métrica de avaliação: O utilizador técnico pode substituir um dos extrusores/hotends através de um assistente;

8. **Requisito:** FF09 - Validar de periféricos (UC09 - A.3.9);

   Métrica de avaliação: O sistema consegue fazer a validação dos periféricos e apresenta um assistente por cada periférico com problemas;

9. **Requisito:** FF10 - Configuração Inicial da Impressora (UC10 - A.3.10);

   Métrica de avaliação: O utilizador pode fazer a configuração inicial da impressora BEE2B;

10. **Requisito:** FF11 - Modificar configurações de rede (UC11 - A.3.11);

    Métrica de avaliação: O utilizador pode alterar as configurações de rede;

11. **Requisito:** FF12 - Restaurar software de fábrica (UC12 - A.3.12);

    Métrica de avaliação: O utilizador pode restaurar o software de fábrica;

12. **Requisito:** FF13 - Atualizar Software (UC13 - A.3.13);

    Métrica de avaliação: O utilizador pode atualizar o software BEE2B;

13. **Requisito:** FF14 - Emparelhar PC com impressora (UC14 - A.3.14);

    Métrica de avaliação: O utilizador pode emparelhar um PC com a impressora BEE2B;

14. **Requisito:** FF16 - Mudar idioma (UC16 - A.3.15);

    Métrica de avaliação: O utilizador pode alternar na GUI BEE2B entre os idiomas Inglês e Português;

15. **Requisito:** FF17 - Calibrar *offset* Z (UC17 - A.3.16);

    Métrica de avaliação: O utilizador pode calibrar o *offset* Z da impressora BEE2B;

16. **Requisito:** FF18 - Calibrar *offset* XY (UC18 - A.3.17);

    Métrica de avaliação: O utilizador pode calibrar o *offset* XY da impressora BEE2B;

Opções para fator Funcional:

- (0%) - Sem acesso à funcionalidade;
- (50%) - acesso parcial à funcionalidade;
- (100%) - acesso total à funcionalidade;

8.4.2.2 Funcionalidade - PrintServer API

No fator PrintServer API da dimensão Funcionalidade são avaliados os métodos API do servidor de impressão BEE2B, as métricas de avaliação são baseadas no número de métodos concluídos descritos no subcapítulo B.

1. **Requisito:** FP01 - Gerir impressora;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão geral da impressora BEE2B disponíveis e completamente funcionais: connect, disconnect, get_state, request_authorization e get_printer_info.

   Total de métodos para gestão geral de impressora: 5

2. **Requisito:** FP02 - Gerir Movimento;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão de movimento da impressora BEE2B disponíveis e completamente funcionais: go_to, home e move.

   Total de métodos para gestão de movimento da impressora: 3

3. **Requisito:** FP03 - Gerir Filamento;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão de filamento da impressora BEE2B disponíveis e completamente funcionais: get_filament, load_filament e unload_filament, filament_identify, add_filament.

   Total de métodos para gestão de filamento da impressora: 5

4. **Requisito:** FP04 - Gerir Ferramentas;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão de ferramentas da impressora BEE2B disponíveis e completamente funcionais: set_tool, get_tool, periph_validation, get_stream, get_temperature e set_temperature.

   Total de métodos para gestão de ferramentas da impressora: 6

5. **Requisito:** FP05 - Gerir Ficheiros;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão dos ficheiros da impressora BEE2B disponíveis e completamente funcionais: remove_file, get_file_list, get_file_settings e set_file_settings.

   Total de métodos para gestão de ficheiros da impressora: 4

6. **Requisito:** FP06 - Gerir Extrusor;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gerir os extrusores da impressora BEE2B disponíveis e completamente funcionais: prepare_extruder, remove_extruder, new_extruder, add_extruder e cancel_extruder.

Total de métodos para gestão dos extrusores da impressora: 5

7. **Requisito:** FP07 - Gerir impressão;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão de uma impressão na impressora BEE2B disponíveis e completamente funcionais: cancel_print, start_print, pause, resume, print_recent_job, remove_object.

   Total de métodos para gestão de uma impressão na impressora BEE2B: 6

8. **Requisito:** FP08 - Gestão de Fila de Impressão;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão da fila de impressão da impressora BEE2B disponíveis e completamente funcionais: remove_from_queue, get_current_job, sort_queue, get_queue.

   Total de métodos para gestão da fila de impressão da impressora BEE2B: 4

9. **Requisito:** FP09 - Gerir Rede;

   Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão da rede da impressora BEE2B disponíveis e completamente funcionais: start_network_config, cancel_network_config, get_networks e connect_network.

   Total de métodos para gestão da rede da impressora BEE2B: 4

10. **Requisito:** FP10 - Gerir Emparelhamento;

    Métrica de avaliação: Métodos da REST API do servidor de impressão para gestão do emparelhamento da impressora BEE2B disponíveis e completamente funcionais: start_pairing, pair, cancel_paring, pairing_list, unpair.

    Total de métodos para gestão do emparelhamento da impressora BEE2B: 5

11. **Requisito:** FP11 - Gerir Calibrações;

    Métrica de avaliação: Métodos da REST API do servidor de impressão para gerir as calibrações da impressora BEE2B disponíveis e completamente funcionais: start_calibrate_z, add_calibrate_z, stop_calibrate_z, start_calibrate_xy, add_calibrate_xy e stop_calibrate_xy.

    Total de métodos para gestão da fila de impressão da impressora BEE2B: 6

12. **Requisito:** FP12 - Impressora Virtual;

    Métrica de avaliação: Implementação de impressora virtual BEE2B para efeitos de testes, é constituída pelos mesmos métodos que a impressora real.

    Total de métodos para a impressora virtual da impressora BEE2B: 53

Estes métodos podem ser testados com ferramentas de teste como Postman, repetier e Gabarit para testes.

Opções para fator PrintServer API:

- (0%) - 0 métodos concluídos;
- (25%) - 25% dos métodos concluídos;
- (50%) - 50% dos métodos concluídos;
- (75%) - 75% dos métodos concluídos;

- (100%) - todos os métodos concluídos;

8.4.2.3 Funcionalidade - BEECura *Plugins*

No fator BEECura *Plugins* da dimensão Funcionalidade são avaliadas as funcionalidades desenvolvidas para o *plugin* do Ultimaker Cura BEECura descritos na secção A.3.18 do apêndice A.

1. **Requisito:** FC01 - *Skin* Ultimaker Cura BEEVERYCREATIVE;

   Métrica de avaliação: O utilizador pode utilizar o *plugin* BEECura.

2. **Requisito:** FC02 - Impressão por rede (UCC2 - A.3.18.2);

   Métrica de avaliação: O utilizador iniciar uma impressão por rede com o *plugin* BEECura.

3. **Requisito:** FC03 - Fila de Impressão (UCC4 - A.3.18.4);

   Métrica de avaliação: O utilizador consegue gerir a fila de impressão com o *plugin* BEECura.

4. **Requisito:** FC04 - Informação da ligação da impressora (UCC1 - A.3.18.1);

   Métrica de avaliação: O utilizador pode utilizar o *plugin* BEECura para obter a informação do estado e ligação da impressora BEE2B.

5. **Requisito:** FC05 - Emparelhamento da impressora (UCC3 - A.3.18.3);

   Métrica de avaliação: O utilizador pode utilizar o *plugin* BEECura para emparelhar um PC com a impressora BEE2B.

6. **Requisito:** FC06 - Perfil de Impressão;

   Métrica de avaliação: O utilizador consegue carregar o perfil da impressora BEE2B para o Ultimaker Cura.

Opções para fator BEECura *Plugins*:

- (0%) - Sem acesso à funcionalidade;
- (50%) - acesso parcial à funcionalidade;
- (100%) - acesso total à funcionalidade;

8.4.2.4 Adaptabilidade - Versatilidade

No fator Versatilidade da dimensão Adaptabilidade é avaliada a facilidade com que o software se adapta a diferentes condições de utilização.

1. **Requisito:** AV01 - Linguagem da GUI de acordo com a língua selecionada (Inglês ou Português);

   Métrica de avaliação: Utilizador consegue alternar entre as idiomas Português e Inglês;

2. **Requisito:** AV02 - GUI BEE2B rede;

   Métrica de avaliação: A GUI deve funcionar normalmente mesmo sem ligação à Internet ou rede local;

3. **Requisito:** AV03- BEECura multiplataforma

   Métrica de avaliação: O software BEECura executa sem problemas nas três principais plataformas: Windows, Linux e macOS.

Opções para fator Versatilidade requisitos AV01 e AV02:

- (0%) - Não;
- (100%) - Sim;

Opções para fator Versatilidade requisito AV03:

- (0%) - 0 plataformas;
- (25%) - 1 plataformas;
- (50%) - 2 plataformas;
- (100%) - todas;

8.4.2.5 Adaptabilidade - Usabilidade

No fator Usabilidade da dimensão Adaptabilidade é avaliado o esforço necessário para a utilização do sistema, baseado num conjunto de implicações e de condições do utilizador.

1. **Requisito:** AU01 - iniciar impressão;

   Métrica de avaliação: A principal tarefa, imprimir, deve ser de fácil acesso. Qualquer utilizador deve conseguir imprimir facilmente.

   Avaliação de questionários a utilizadores que nunca leram o manual. A pergunta do questionário que influencia esta métrica é: 2.5

2. **Requisito:** AU02 - GUI intuitiva;

   Métrica de avaliação: A GUI deve ser simples, rápida e fácil de entender. O utilizador não deve precisar de ler o manual para ser capaz de navegar na GUI e ter acesso a todas as funcionalidades.

   Avaliação de questionários a utilizadores que nunca leram o manual. A pergunta do questionário que influencia esta métrica é: 2,6 e 2,9;

3. **Requisito:** AU03 - Design da GUI homogéneo;

   Métrica de avaliação: O design de todos os ecrãs da GUI é homogéneo. Botões, navegação, fontes, entre outros elementos, consistentes e sempre na mesma posição;

4. **Requisito:** AU04 - GUI com *feedback* imediato;

   Métrica de avaliação: Todas as ações na GUI fornecem algum tipo de *feedback* visual imediato;

5. **Requisito:** AU05 - GUI com boa usabilidade;

   Métrica de avaliação: Avaliação de questionários a utilizadores que nunca leram o manual. A pergunta do questionário que influencia esta métrica é: 2.7;

6. **Requisito:** AU06 - Assistentes de manutenção;

   Métrica de avaliação: Existem assistentes para as tarefas de manutenção na GUI;

7. **Requisito:** AU07 - Ajuda na manutenção;

   Métrica de avaliação: As tarefas de manutenção devem ter procedimentos simples e com apresentação de ajuda. Avaliação de questionários a utilizadores. A pergunta do questionário que influencia esta métrica é: 2,10;

Para a avaliação dos requisitos AU01, AU02, AU05 e AU07 serão feitos no mínimo 6 questionários.

Opções para fator Usabilidade requisitos AU03, AU04 e AU06:

- (0%) - Não;
- (100%) - Sim;

Opções para fator Usabilidade requisito AU01, AU02, AU05 e AU07:

- (0%) - Média entre 0 e 1 pontos;
- (25%) - Média entre 1 e 2 pontos;
- (50%) - Média entre 2 e 3 pontos;
- (75%) - Média entre 3 e 4 pontos;
- (100%) - Média entre 4 e 5 pontos;

8.4.2.6 Adaptabilidade - Manutenção

No fator Manutenção da dimensão Adaptabilidade é avaliado o esforço necessário para a realização de alterações específicas do software.

1. **Requisito:** AM01 - Padrões de design;

   Métrica de avaliação: Padrões de design são aplicados corretamente sempre que é apropriado;

2. **Requisito:** AM02 - Dados sensíveis;

   Métrica de avaliação: A aplicação não guarda qualquer tipo de dados sensíveis sobre o utilizador;

3. **Requisito:** AM03 - Registo de acções;

   Métrica de avaliação: Todas as ações do utilizador ficam registadas num *log* que será armazenado de forma persistente para futura análise;

4. **Requisito:** AM04 - Registo de Erros;

   Métrica de avaliação: Todos os erros são registados num *log* que será armazenado de forma persistente para futura análise;

5. **Requisito:** AM05 - Documentação de código;

   Métrica de avaliação: Existe documentação do código atualizada;

6. **Requisito:** AM06 - PEP8;

   Métrica de avaliação: todo o código deve estar organizado segundo as normas de codificação PEP8;

7. **Requisito:** AM07 - Testes unitários

   Métrica de avaliação: A cobertura dos testes unitários deve ser superior a 70%

8. **Requisito:** AM08 - MVC

   Métrica de avaliação: Clara separação entre GUI e classes de negócio segundo o padrão de design MVC

9. **Requisito:** AM09 - Glossário

   Métrica de avaliação: Existe um glossário com toda a terminologia do domínio.

Opções para fator Manutenção requisito AM05:

- (0%) - Sem documentação;
- (25%) - Entre 0% a 25% do código documentado;
- (50%) - Entre 25% a 50% do código documentado;
- (75%) - Entre 50% a 75% do código documentado;
- (100%) - Todo o código documentado;

Opções para restantes requisitos do fator Manutenção:

- (0%) - Não;
- (100%) - Sim;

8.4.2.7 Eficiência - Navegação

No fator Navegação da dimensão Eficiência é avaliada a facilidade e qualidade de navegação da GUI BEE2B.

1. **Requisito:** EN01 - Estrutura GUI;

   Métrica de avaliação: Todos os ecrãs devem permitir ir para o menu inicial e manutenção;

   Avaliação de questionários de utilizador. As perguntas do questionário que influenciam esta métrica são: 2.3;

2. **Requisito:** EN02 - GUI rápida;

   Métrica de avaliação: A GUI é rápida e simples, funcionalidades que demorem mais de 1 segundo apresentam uma barra de progresso;

3. **Requisito:** EN03 - GUI sem erros;

   Métrica de avaliação: A GUI não apresenta erros, em caso de erros inesperados, estes são tratados e registados no *log* de erros;

4. **Requisito:** EN04 - Utilizador controla GUI;

   Métrica de avaliação: Todas as ações devem ser efetuadas por interação humana. Não devem existir ações automáticas à exceção das notificações;

5. **Requisito:** EN05 - Notificações;

Métrica de avaliação: A GUI possui um sistema de notificações ao utilizador para o alertar de alterações importantes (por exemplo: atualização de software, impressão concluída ou aquecimento concluído);

Opções para fator Navegação:

- (0%) - Não;
- (100%) - Sim;

8.4.2.8 Eficiência - Qualidade do Conteúdo

No fator Qualidade do Conteúdo da dimensão Eficiência é avaliada a qualidade do conteúdo apresentado na GUI BEE2B.

1. **Requisito:** EQ01 - Texto correto;

   Métrica de avaliação: Todas as frases são curtas, bem escritas e bem construídas gramaticalmente;

2. **Requisito:** EQ02 - Mensagens claras;

   Métrica de avaliação: Todas as mensagens são fáceis de entender e não apresentam códigos não legíveis ou forma similar de codificação;

3. **Requisito:** EQ03 - GUI para daltónicos;

   Métrica de avaliação: A utilização de cores deve tem especial atenção para ser claro para daltónicos;

   Avaliação de questionários de utilizador. Não há um mínimo de questionários necessários devido à especificidade do utilizador. As perguntas do questionário que influenciam esta métrica são: 1.1;

4. **Requisito:** EQ04 - Racismo;

   Métrica de avaliação: Não é apresentada qualquer tipo de mensagem ofensiva, racista, xenófoba ou antissemita;

Opções para fator Navegação:

- (0%) - Não;
- (100%) - Sim;

8.4.2.9 Eficiência - Suporte

No fator Suporte da dimensão Eficiência é avaliada a qualidade do serviço de suporte prestado ao utilizador.

1. **Requisito:** ES01 - Configuração inicial fácil ;

   Métrica de avaliação: A configuração inicial é fácil para o público alvo;

   Avaliação de questionários de utilizador. As perguntas do questionário que influenciam esta métrica são: 2.1;

2. **Requisito:** ES02 - GUI estável;

   Métrica de avaliação: A GUI é estável e corre sistematicamente sem falhas;

Avaliação de questionários de utilizador. As perguntas do questionário que influenciam esta métrica são: 2.12;

3. **Requisito:** ES03 - Cópia de Segurança;

   Métrica de avaliação: Existe uma estratégia de cópias de segurança da base de dados do PrintServer BEE2B;

4. **Requisito:** ES04 - Recuperação de Falha;

   Métrica de avaliação: Existe um procedimento de recuperação de falha de base de dados do PrintServer BEE2B;

5. **Requisito:** ES05 - Facilidade de atualização;

   Métrica de avaliação: O utilizador deve facilmente atualizar o software da impressora BEE2B;

   Avaliação de questionários de utilizador. As perguntas do questionário que influenciam esta métrica são: 2.11;

6. **Requisito:** ES06 - Manual;

   Métrica de avaliação: O utilizador tem acesso ao manual em duas línguas, português e inglês;

Opções para requisito ES03 do fator Suporte:

- (0%) - Não existe uma estratégia de cópias de segurança da base de dados;
- (50%) - Cópias de segurança diárias;
- (75%) - Cópias de segurança diárias com redundância;
- (100%) - Cópias de segurança diárias com redundância e documentação;

Opções para requisito ES04 do fator Suporte:

- (0%) - Não existe um procedimento de recuperação de falha de base de dados;
- (50%) - Procedimento funcional de recuperação de falha de base de dados;
- (100%) - Procedimento funcional de recuperação de falha de base de dados e documentação;

Opções para requisito ES06 do fator Suporte:

- (0%) - Não existe nenhum manual;
- (50%) - Existe manual em português ou em inglês;
- (100%) - Existe manual disponível em português e inglês;

8.5 Avaliação da solução

A avaliação da solução é feita em dois momentos, nos testes alpha e nos testes beta. Nos primeiros testes, a solução ainda não está completamente desenvolvida e pode não ter uma impressora real ligada usando assim a impressora virtual. Os últimos testes devem ser efetuados já na impressora 3D.

No momento de redação deste documento, ainda são foram efetuados os testes alpha, assim, esta avaliação ainda não será a avaliação final.

Para os testes alpha foi utilizado uma raspberry pi com um ecrã de sete polegadas como o da figura 2.1, a GUI apresentava o layout da terceira iteração (podemos ver a evolução da GUI na figura 6.1) e com os seguintes casos de uso concluídos a funcionar com a impressora virtual:

- UC01: Imprimir a partir do PC
- UC02: Iniciar impressão a partir de drive USB
- UC03: Iniciar Impressão de ficheiro recente
- UC04: Impressão de uma cena 3D
- UC07: Trocar Filamento
- UC08: Trocar Extrusor
- UC16: Mudar Idioma

Quem testou a aplicação respondeu aos questionários de qualidade do apêndice C. Os resultados desse questionário podem ser vistos no apêndice D. Destes resultados foram obtidos algumas métricas para serem utilizadas na avaliação do QEF, podemos ver o resultado desta avaliação no apendice E.

O resultado de 73% indica que o projeto está a evoluir positivamente, embora o aspeto funcional ainda não estar completamente implementado. Apesar de um dos aspetos importantes, a remoção do objeto da impressão, ainda não estar completamente implementado, a usabilidade da GUI foi aprovada nestes primeiros testes e possibilitou bons melhoramentos implementados na quarta e última iteração do layout da GUI.

Como ainda não há impressoras BEE2B a funcionar, não foi possível, à data deste documento, testar as hipóteses apresentadas na secção 8.2. Isto só será possível no final do projeto, obtendo os dados de telemetria das impressoras BEE2B e comparando-as com outras impressoras da BEEVERYCREATIVE.

Capítulo 9

Conclusão e Perspetivas de Trabalho Futuro

Neste capítulo são apresentadas as conclusões desta dissertação. Na secção 9.1 é efetuada uma descrição resumida do trabalho desenvolvido, realçando os objetivos atingidos. O trabalho que falta desenvolver até ao final do projeto e as ideias para o trabalho futuro são desratas na secção 9.2.

9.1 Conclusões e Objetivos Alcançados

O presente projeto surgiu da intenção de melhorar a usabilidade das interfaces gráficas com o utilizador das impressoras 3D, depois de se verificar que os ecrãs da grande maioria das impressoras 3D têm um baixo nível de usabilidade. Embora existam algumas impressoras com ecrãs já com uma usabilidade melhorada, verificou-se uma limitação comum a todas, nenhuma oferece funcionalidades que promovam melhores rentabilidades e menos desperdício de matéria prima.

Numa primeira fase foram analisadas vários softwares de impressão 3D, para perceber se seria necessário construir uma solução de raiz para a manipulação da cena 3D ou se podíamos optar pela construção de *plugins* para alguma destas ferramentas. Foram ainda estudadas *frameworks* e módulos python, para perceber quais a utilizar para os módulos a implementar, a Interface Gráfica e o Servidor de Impressão. Durante esta fase foi implementado um servidor de entrega contínua para promover melhor eficiência do desenvolvimento do projeto e ao mesmo tempo ajudar a cumprir alguns requisitos não funcionais.

Durante a fase de design foram documentados os casos de uso funcionais e não funcionais da solução.

A fase da construção da solução foram programados os vários módulos da solução, BEECura, PrintServer BEE2B e GUI BEE2B. Durante esta fase foram também desenvolvidos todos os testes para o caso de uso em desenvolvimento. Durante esta fase, apenas foi desenvolvido o necessário de cada módulo para cumprir o caso de uso a desenvolver.

Estas últimas duas fases são feitas em simultâneo, ou seja, é feito o design para um determinado caso de uso e depois é programado e desenvolvidos os respetivos testes.

Relativamente ao design gráfico da GUI, foram efetuadas quatro iterações, a cada uma, foram feitas sessões de *design thinking* que tiveram como objetivo pensar num grupo de casos de uso e elaborar um esboço para o seu *layout* e avaliar o *layout* da iteração anterior procurando promover melhoramentos.

No final, as aplicações desenvolvidas foram devidamente avaliadas por utilizadores que respondendo a questionário de qualidade forneceram métrica para a avaliação global QEF.

Devido ao facto de, no momento de redação deste documento, o projeto ainda não ter terminado, neste momento ainda não foram atingidos todos os objetivos inicialmente propostos. No entanto, pela avaliação do QEF, verificamos que, no que diz respeito à usabilidade da GUI, o objetivo foi atingido, no entanto, apenas no final do projeto, com a impressora BEE2B já a funcionar, será possível confirmar a hipótese a testar, ou seja, se a GUI possibilitou aumentos de *uptime* e diminuições de *downtime*.

Como inicialmente proposto, foram desenvolvidas duas aplicações, uma GUI para ecrã tátil e um servidor de impressão, e um *pluggin* para o Ultimaker Cura (BEECura).

O BEECura permite manipular a cena 3D, ver o estado do servidor de impressão e imprimir a partir da rede ou exportar o ficheiro de impressão para uma drive USB.

A GUI BEE2B permite, imprimir um ficheiro a partir de uma drive USB ou ficheiro recente, ver a renderização 2D vista de topo e selecionar um objeto para remover, trocar de filamento, trocar de extrusor ou mudar de língua.

O Servidor de Impressão disponibiliza uma API para as duas aplicações anteriormente descritas poderem comunicar com a impressora 3D, este servidor tem implementado uma impressora Virtual que simula o comportamento da impressora 3D, pois durante o desenvolvimento esta impressora ainda não estará disponível.

9.2 Trabalho Futuro

Até ao final do projeto, Junho de 2020, ainda existem casos de uso e componentes para serem implementados (toda a parte gráfica dos diversos casos de uso já está implementada):

- Servidor de entrega contínua - Concluir o estágio Documentação, de momento apenas documenta a API;
- Servidor de entrega contínua - Estágio Deployment;
- UC09 - Validação de Periféricos;
- UC10 - Configuração Inicial da Impressora;
- UC11 - Modificar Configurações de Rede;
- UC12 - Restaurar Software de fábrica;
- UC13 - Atualizar Software;
- UC14 - Emparelhar PC com a Impressora;
- UC17 - Calibrar Offset Z;
- UC17 - Calibrar Offset XY;
- BEECura UCC3 - Emparelhar o PC com uma impressora BEE2B;
- BEECura UCC4 - Gerir fila de impressão;
- BEECura Deployment - Neste momento, apenas é distribuído isoladamente como plugin, ainda é necessário implementar a sua distribuição incorporado com o Ultimar Cura.

- PrintServer BEE2B - implementar métodos para os caso de uso a implementar;
- PrintServer BEE2B - testar camada de comunicação com impressora BEE2B e corrigir o necessário.

Mesmo não estando no final do projeto, observando o trabalho já realizado numa perspetiva autocrítica construtiva, verifica-se as seguintes limitações e potenciais necessidades de desenvolvimento futuro (depois do final do projeto):

- A GUI está preparada para a adição futura de novos idiomas, no entanto, para adicionar um novo idioma é necessário editar "à mão"um ficheiro json. Um melhoramento futuro seria o desenvolvimento de uma pequena aplicação, que possibilitasse ao utilizador importar esse json e editá-lo durante a apresentação das diversas vistas, facilitando assim este processo e minimizando a probabilidade de erros.
- Quando o utilizador inicia um impressão a partir do BEECura poderia ter uma opção de número de cópias, à semelhança do que acontece com uma impressora de papel normal. Esta funcionalidade iria facilitar nos casos em que o utilizador deseja imprimir a mesma cena várias vezes. Esta funcionalidade também se poderia estender à gestão de Fila de impressão (UCC4), permitindo ao utilizador diminuir ou aumentar o número de cópias das cenas presentes na fila.
- De momento o servidor de entrega contínua não usa nenhuma ferramenta de *build*, tudo é orquestrado com recursos a *scripts bash*, um melhoramento a este processo seria utilizar a ferramenta Gradle juntamente com o *plugin* pygradle para orquestrar todo o ciclo de vida da aplicação. Esta ferramenta foi testado durante a fase de análise mas descartada devido a problemas encontrados que estavam a atrasar o processo de desenvolvimento, como o servidor de entrega contínua não é prioritário para o desenvolvimento deste projeto, simplificou-se o processo optando-se por *scripts bash* para as diversas tarefas.

Bibliografia

Aeronautics, National e Space Administration (2014). *Space Tools On Demand:3D Printing in Zero G*. URL: https://www.nasa.gov/sites/default/files/files/3D_Printing-v3.pdf (acedido em 15/09/2019).

Allee, Verna (2006). *What is ValueNet Works Analysis?* Harold Jarche. URL: http://jarche.com/wp-content/uploads/2007/03/what_is_valuenet_works_analysis.pdf (acedido em 07/01/2019).

Associates, DRM (2016). *Value Analysis and Function Analysis System Technique*. NPD Solutions. URL: http://www.npd-solutions.com/va.html (acedido em 07/01/2019).

Atala, Anthony (1 de mar. de 2011). «Imprimir um rim humano». Em: *TED*: *TED2011*. URL: https://www.ted.com/talks/anthony_atala_printing_a_human_kidney?language=pt#t-995090 (acedido em 15/09/2019).

Awards, TCT (2018). *2018 Winners & Highly Commended*. TCT Awards. URL: https://tctawards.com/tctawards/en/page/2018-winners (acedido em 03/01/2019).

BEEVERYCREATIVE (2018). *Agência Espacial Europeia vai testar protótipo de impressora 3D desenvolvida pela BEEVERYCREATIVE*. BEEVERYCREATIVE. URL: https://beeverycreative.com/press-release-project-melt-update/ (acedido em 15/09/2019).

Buffalo, University at (2008). *CNC notes - Computer numerical control (CNC)*. University at Buffalo. URL: https://www.studocu.com/en/u/1030056 (acedido em 03/10/2019).

Burns, Marshall (1999). *The StL Format. Standard Data Format for Fabbers*. Fabbers. URL: https://www.fabbers.com/tech/STL_Format#Sct_specs (acedido em 16/09/2019).

Charles W. Hull Arcadia, Calif. (11 de mar. de 1986). «Apparatus for Production of Three-Dimensional Objects by Stereolithography». US4575330. 3D Systems Inc. URL: https://patentimages.storage.googleapis.com/5c/a0/27/e49642dab99cf6/US4575330.pdf (acedido em 15/09/2019).

Chatellier, Cara (25 de out. de 2017). «Mind-blowing 3D printing feats and what to expect in 2018». Em: *readwrite*: *Connected Devices*. URL: https://readwrite.com/2017/10/25/3d-printing-feats-expectations-2018/ (acedido em 15/09/2019).

Daly, Jimmy (13 de ago. de 2013). «The History of 3D Printing [Infographic]». Em: *State Tech Magazine*. URL: https://statetechmagazine.com/article/2013/08/history-3d-printing-infographic (acedido em 15/09/2019).

Danaylov, Nikola (4 de mar. de 2012). «Bespoke Innovations' 3D-printed Prosthetics: If Lizards Can Grow Tails, Humans Should Print Limbs». Em: *Singularity Weblog*. URL: https://www.singularityweblog.com/bespoke-innovations-3d-printed-prosthetics-if-lizards-can-grow-tails-humans-should-print-limbs/ (acedido em 15/09/2019).

Django (2019). *Django packages - Django REST framework*. Django. URL: https://djangopackages.org/grids/g/django-rest-framework/ (acedido em 07/10/2019).

ESA (2019). *Building a lunar base with 3D printing*. ESA. URL: http://www.esa.int/Our_Activities/Space_Engineering_Technology/Building_a_lunar_base_with_3D_printing (acedido em 15/09/2019).

Escudeiro P. & Bidarra, J. (2008). «Quantitative Evaluation Framework (QEF)». Em: *RISTI – Revista Ibérica de STI* 1, pp. 16–27. ISSN: 1646-9895. URL: https://www.researchgate.net/publication/257579051_Quantitative_Evaluation_Framework_QEF.

Fabb, General (4 de set. de 2014). *the-3d-printshow-global-awards-2014*. Fabbaloo. URL: https://www.fabbaloo.com/blog/2014/9/4/the-3d-printshow-global-awards-2014 (acedido em 03/01/2019).

Fowler, Martin (19 de out. de 1996). *Analysis Patterns: Reusable Object Models*. Ed. por Addison-Wesley Professional. ISBN: 978-0201895421.

– (1 de mai. de 2006). *Continuous Integration*. URL: https://martinfowler.com/articles/continuousIntegration.html (acedido em 08/10/2019).

– (30 de mai. de 2013a). *Continuous Delivery*. URL: https://martinfowler.com/bliki/ContinuousDelivery.html (acedido em 08/10/2019).

– (30 de mai. de 2013b). *DeploymentPipeline*. URL: https://martinfowler.com/bliki/DeploymentPipeline.html (acedido em 08/10/2019).

Gordon, Leslie (8 de set. de 2011). «The Urbee hybrid: the world's first 3D printed car». Em: *Machine Design*: *News*. URL: https://www.machinedesign.com/news/world-s-first-3d-printed-uav (acedido em 15/09/2019).

Humble, Jez (6 de ago. de 2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Ed. por Addison-Wesley Professional. 1st Edition, pp. 17–21. ISBN: 978-0321601919.

Intel (2018). *Intel NUC Board NUC7i7DNBE*. URL: https://www.intel.com/content/www/us/en/products/boards-kits/nuc/boards/nuc7i7dnbe.html (acedido em 03/01/2019).

ISO (2016). *Specification for additive manufacturing file format (AMF)*. ISO/ASTM 52915:2016. Versão 1.2. ISO. URL: https://www.iso.org/obp/ui/#iso:std:iso-astm:52915:ed-2:v1:en (acedido em 16/09/2019).

Jarche, Harold (21 de mar. de 2011). *Mapping quality with VNA*. Harold Jarche. URL: http://jarche.com/2011/03/mapping-quality-with-vna/ (acedido em 08/01/2019).

Jetbrains (2019). *Python 2019 - The state of Developer Ecosystem in 2019 Infographic*. Jetbrains. URL: https://www.jetbrains.com/lp/devecosystem-2019/python/ (acedido em 07/10/2019).

Kenton, Will (3 de mar. de 2018). *Perceived Value*. Investopedia. URL: https://www.investopedia.com/terms/p/perceived-value.asp (acedido em 07/01/2019).

Koen, Peter A. (24 de mai. de 2004). «Front End of Innovation: Effective Methods, Tools and Techniques Workshop». Em: *Stevens Institute of Technology*, pp. 11–31. URL: https://web.stevens.edu/cce/NEW/PDFs/FEIEffectiveMethodstoolWorkshop.pdf.

Larman, Craig (30 de out. de 2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Ed. por Prentice Hall. 3rd Edition. ISBN: 978-0131489066.

Locker, Anatol (12 de jul. de 2019). «2019 Best 3D Slicer Software for 3D Printers (Most are Free)». Em: *All3DP*. URL: https://all3dp.com/1/best-3d-slicer-software-3d-printer/ (acedido em 16/09/2019).

Makerbot (2019). *3D Printers For Educators Or Professionals*. Makerbot. URL: https://www.makerbot.com/ (acedido em 15/09/2019).

Moon, Mariella (20 de jun. de 2014). «What you need to know about 3D-printed organs». Em: *Engadget*: *Medicine*. URL: https://www.engadget.com/2014/06/20/3d-printed-organ-explainer/?guccounter=2 (acedido em 15/09/2019).

Nick Rich BSc MBAMatthias Holweg, Dipl.-Wirtschaftsing.(FH) MSc (2000). «Value Analysis Value Engineering». Em: *Institute of value management Australia*. URL: http://www.

ivma.org.au/images/articlepdfs/Other/Value_Analysis_Value_Engineering_-_Rich_and_Holweg_2000.pdf.

Octoprint (2019). *Octoprint*. Octoprint. URL: https://octoprint.org/ (acedido em 16/09/2019).

Organovo (21 de mar. de 2011). *History*. Organovo. URL: https://organovo.com/about/history/ (acedido em 15/09/2019).

Paukstelis, Paul (8 de nov. de 2018). *OctoPrint-Cancelobject*. OctoPrint. URL: https://plugins.octoprint.org/plugins/cancelobject/ (acedido em 15/11/2018).

Quick, Darren (2 de nov. de 2010). «The Urbee hybrid: the world's first 3D printed car». Em: *New Atlas*: *Automotive*. URL: https://newatlas.com/urbee-3d-printed-car/16795/ (acedido em 15/09/2019).

RepRap (23 de mai. de 2015a). *RepRapOneDarwin*. RepRap. URL: https://reprap.org/wiki/RepRapOneDarwin (acedido em 15/09/2019).

– (16 de jul. de 2015b). *RRD Full Graphic Smart Controler Info*. RepRap. URL: https://reprap.org/wiki/File:RRD_FULL_GRAPHIC_SMART_CONTROLER_INFO.JPG (acedido em 15/09/2019).

– (24 de set. de 2019a). *G-code*. RepRap. URL: https://reprap.org/wiki/G-code (acedido em 03/10/2019).

– (17 de mar. de 2019b). *Welcome to RepRap.org*. RepRap. URL: https://reprap.org/wiki/RepRap (acedido em 15/09/2019).

Shapeways (2019). *3D Printing Service*. Shapeways. URL: https://www.shapeways.com/ (acedido em 15/09/2019).

Simplify3D (2019). *Simplify3D Version 4.1*. Simplify3D. URL: https://www.simplify3d.com/software/release-notes/version-4-1-0/ (acedido em 16/09/2019).

Slic3r (2019). *About*. Slic3r. URL: https://slic3r.org/about/ (acedido em 16/09/2019).

Story, New (2019). *We Pioneer Solutions To End Global Homelessness*. New Story. URL: https://newstorycharity.org/ (acedido em 15/09/2019).

Woodall, Tony (2003). «Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis». Em: *Academy of Marketing Science Review*.

Apêndice A

Engenharia de Requisitos

Nesta secção vamos encontrar a identificação dos requisitos de hardware, funcionais e não-funcionais. Começamos pela visão da solução na secção A.1, passando para o glossário da terminologia utilizada na secção A.2. Os casos de uso onde são registados os requisitos são apresentados na secção A.3. Na secção A.4 são apresentados os restantes requisitos que não foram capturados pelos casos de uso. Por fim, na secção A.5 é apresentada a gestão de risco.

A.1 Visão

Desenvolver a próxima geração de interfaces gráficas com o Utilizador para impressoras 3D, tolerante a falhas, com boa usabilidade e que permita ao utilizador remover um objeto no qual tenha ocorrido algum problema durante a impressão.

A.1.1 Posicionamento

As interfaces gráficas com o utilizador atuais das impressoras 3D são pouco intuitivos e não permitem remover objetos individuais durante a impressão, ou imprime-se toda a Cena 3D até ao final da impressão ou se cancela-se toda a impressão (incluindo os objetos que estão a ser impressos corretamente).

Devido à fraca usabilidade das interfaces gráficas com o utilizador atuais, as impressoras 3D têm um tempo de uso baixo e obrigam à utilização de um computador para manipular a impressora. Estas interfaces também não permitem a remoção de apenas um objeto da impressão levando assim a um maior tempo de impressão e desperdício de material (atualmente, quando um objeto falha as opções disponíveis são: cancelar tudo ou imprimir tudo até ao final. Qualquer uma das soluções não é aceitável pois, se cancelarmos a impressão perdemos o tempo de impressão até ao momento de cancelamento bem como todo o material já utilizado nos objetos que estão a ser impressos, se deixarmos imprimir tudo até ao fim, todo o material usado no objeto com defeito será desperdício e o tempo de impressão também será penalizado pois a impressora vai imprimir o objeto até ao fim necessitando assim do tempo normal da impressão desse objeto).

Este produto destina-se a impressoras 3D direcionadas para a Indústria, terá funcionalidades diferenciadoras como remoção de um objeto durante a impressão e vai se distinguir da concorrência não só por estas funcionalidades inovadores, mas também pela sua usabilidade.

A.1.2 Objetivos

- **Preparação de impressão:** Permitir ao utilizador importar um modelo 3D e prepará-lo para o processo de impressão. Permitir ao utilizador ajustar os parâmetros de impressão como, resolução, densidade de preenchimento ou material de suporte.
- **Iniciar Impressão:** Transferir para a impressora um trabalho de impressão e iniciar o processo de impressão.
- **Monitorar a Impressão:** Permitir ao utilizador monitorar o processo de impressão. Permitir ao Utilizador ter acesso à imagem em *streaming* do trabalho de impressão e verificar o estado do mesmo. Deverá também ser possível realizar algumas ações como cancelar ou meter a impressora em *pause*.
- **Fila de Impressão:** Permitir ao Utilizador adicionar e gerir trabalhos de impressão.
- **Mudar Filamento:** Este serviço permite ao utilizador substituir o filamento carregado na impressora. Operações de sistema para este Use Case: iniciar aquecimento de extrusor, cancelar aquecimento de extrusor, descarregar filamento, carregar filamento, ler informação de filamento, escrever informação de filamento.
- **Mudar Extrusor:** Ester serviço permite ao utilizador substituir o extrusor carregado na impressora. Operações de sistema para este Use Case: iniciar aquecimento de extrusor, cancelar aquecimento de extrusor, libertar extrusor, ler informação de extrusor, escrever informação de extrusor.
- **Imprimir Ficheiro:** Permitir ao utilizador imprimir ficheiros a partir do armazenamento local da impressora, como o cartão SD interno da impressora ou de uma drive USB. Operações de sistema para este Use Case: iniciar impressão a partir de USB, iniciar impressão de um ficheiro recente, listar ficheiros, remover ficheiros, adicionar ficheiros.

A.1.3 Descrição genérica do Produto

A GUI BEE2B será embutida na impressora 3D BEE2B e irá proporcionar aos utilizadores uma interação com a impressora 3D mais fluída e intuitiva.

O BEECura, irá possibilitar aos utilizadores importar um modelo 3D, prepará-lo para o processo de impressão, emparelhar uma impressora BEE2B presente na rede local, iniciar uma impressão 3D e gerir a sua fila de impressão.

Como se verifica na Figura A.1 estas Interfaces irão colaborar com o Servidor de Impressão da BEE2B (BEE2BServer) que por sua vez irá colaborar com o BEEStats.

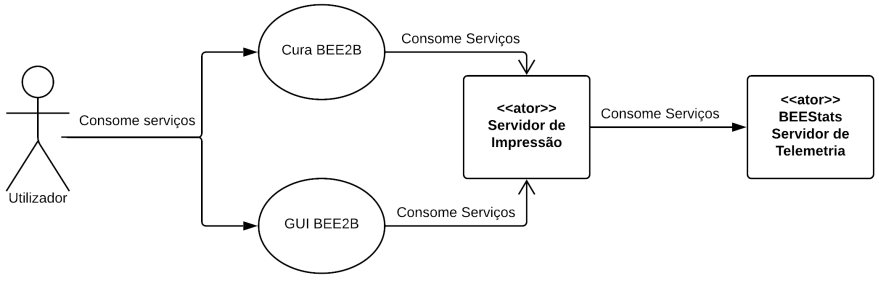


FIGURA A.1: Visão da solução.

Tabela A.1: Benefícios do produto.

Funcionalidade	**Benefício**
Remover objeto de impressão	Redução de desperdício de material e aumento de tempo de impressão

A.2 Glossário do Produto

Este glossário descreve alguns termos utilizados no domínio da impressão 3D.

Tabela A.2: Glossário do produto.

Cena 3D	Conjunto de modelos 3d organizados no tabuleiro de impressão.
Cura	Software de Impressão 3D da Ultimaker.
Aquecimento da impressora	Processo de aquecimento da mesa de impressão e dos vários extrusores que serão utilizados na próxima impressão.
Limpar Mesa	Processo de limpeza de mesa de impressão, este processo é indispensável para que as futuras impressões sejam realizadas com sucesso.
Filamento	Consumível de uma impressora 3D. Pode ser constituído por vários materiais, sendo o Poliácido Láctico (PLA) o mais habitual.
Extrusor	é a parte da impressora 3D responsável pela fusão da matéria-prima e a sua formação num perfil contínuo.
PLA	O PLA é um polímero constituído por moléculas de ácido láctico, com propriedades semelhantes as do Polietileno Tereftalato (PET) que é utilizado para fabricar envases, mas que também é biodegradável. Degrada-se facilmente em água e dióxido de carbono.
Hotend	É a parte do extrusor que derrete o filamento para extrusão e ajuda a manter uma temperatura consistente e precisa para impressões bem sucedidas.
Fila de Impressão	Uma fila de Impressão é uma coleção de trabalhos de Impressão cuja ordem é mantida e as principais operações na coleção são a adição de entidades à posição final da coleção e a remoção de entidades do fim da fila. Isto torna a fila de impressão uma estrutura de dados *First In, First Out* (FIFO).
Impressão 3D	É uma forma de tecnologia de fabricação aditiva onde um modelo tridimensional é criado por sucessivas camadas de material.
Extrusão	É o processo usado para criar objetos numa impressora 3D, consiste em fundir o material, para posteriormente o depositar na coordenada desejada e arrefecê-lo depois de depositado para voltar ao estado sólido.
Trabalho de Impressão	É um objeto ou conjunto de objetos (Cena 3D) que foi enviado para impressão.
FIFO	Método para organizar e manipular um *buffer* de dados, onde a entrada mais antiga é processada primeiro. É análogo ao processamento de uma fila com o comportamento de primeiro a chegar, primeiro a ser servido: onde as pessoas saem da fila na ordem em que chegam.

A.3 Casos de Uso

Na Figura A.2, é apresentado o diagrama de casos de uso da solução.

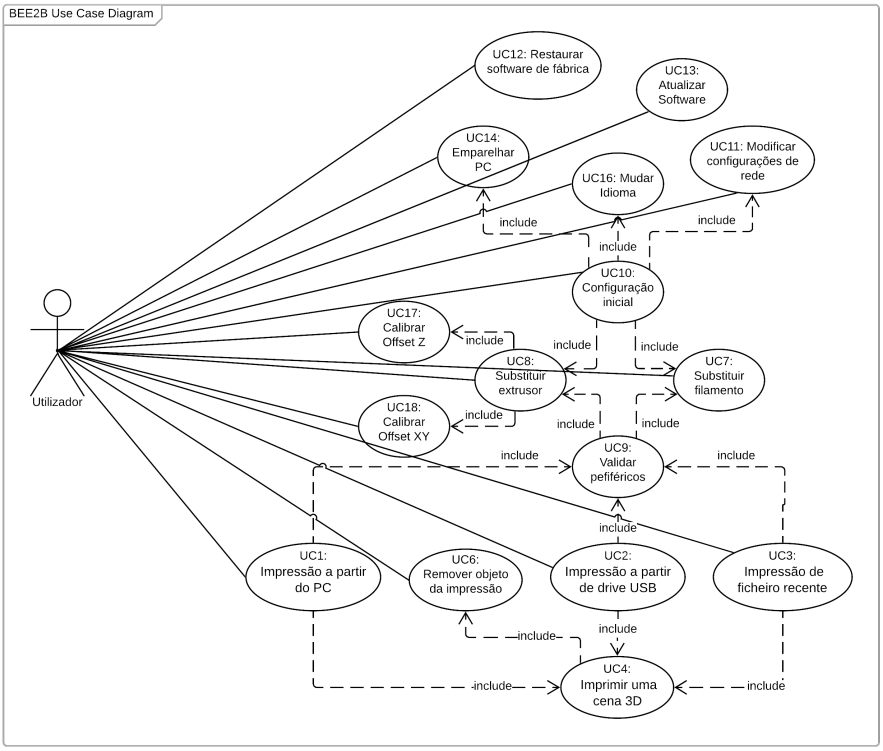


FIGURA A.2: Diagrama de casos de uso.

A.3.1 UC01: Iniciar Impressão a Partir do PC

A.3.1.1 Formato Breve

O utilizador inicia a impressão de uma cena 3d a partir do software Cura BEE2B. O Sistema inicia o processo de validação dos periféricos da impressora (filamento, extrusores). É iniciado o processo de aquecimento da impressora. Quando a impressora estiver à temperatura ideal dá-se o início da impressão. Depois da impressão estar concluída começa o processo de arrefecimento da impressora. Depois da impressora arrefecer o suficiente o sistema apresenta o relatório da impressão, pede ao utilizador para retirar a impressão da mesa e proceder à limpeza da mesma.

A.3.1.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: processo completamente automático depois do utilizador mandar imprimir.

Pré-condições

A impressora está no estado **Ligado**. A cena 3D está organizada com os objetos que o Utilizador deseja imprimir.

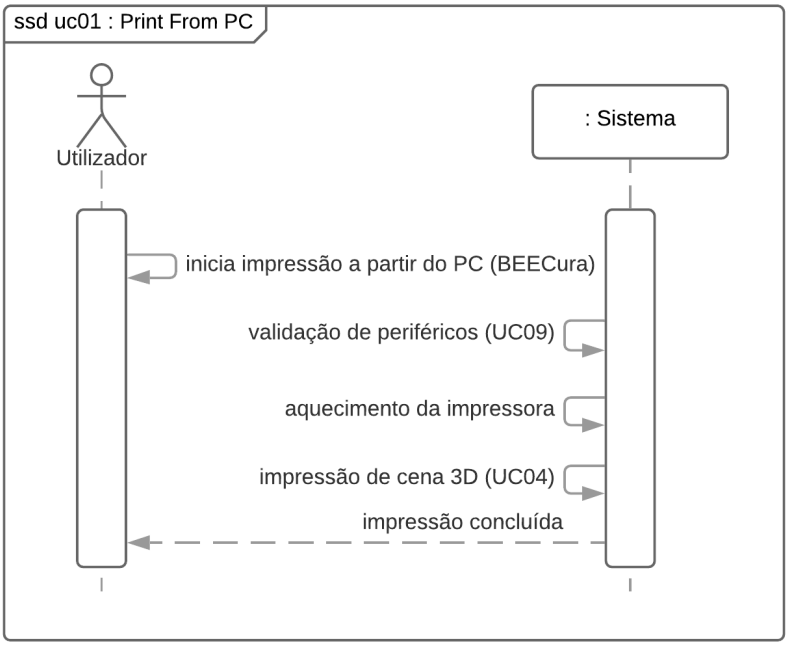


FIGURA A.3: *System Sequence Diagrams* (SSD) do UC01: Iniciar Impressão a Partir do PC.

Cenário de sucesso principal (ou fluxo básico)

1. O utilizador inicia no software CURA o processo de impressão.
2. O sistema inicia o processo de validação dos periféricos (UC9 - A.3.9).
3. O Sistema inicia o processo de aquecimento da impressora.
4. O Sistema inicia o processo de impressão da cena 3D (UC4 - A.3.4).
5. O Sistema concluí o processo de impressão;
6. O Sistema inicia o processo de arrefecimento da impressora e informa o utilizador do resumo da impressão e informações adicionais/complementares.
7. O Sistema apresenta uma mensagem para informar o utilizador que pode retirar a impressão e limpar a mesa;
8. O utilizador remove a impressão e limpa a mesa e confirma a operação;

Extensões (ou fluxos alternativos)

a. A qualquer momento o utilizador solicita o cancelamento da impressão.

   i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

Muito frequente: será naturalmente o caso de uso mais frequente;

Questões em aberto

- Como é que o utilizador irá ser notificado da conclusão da impressão?
- A notificação será enviada por correio eletrónico?
- Que dados devem constar da mensagem a enviar?

A.3.2 UC02: Iniciar Impressão a Partir de Drive USB

A.3.2.1 Formato Breve

O Utilizador inicia a impressão a partir do aplicação gráfica da impressora escolhendo a opção USB. O Sistema apresenta uma lista de ficheiros disponíveis na pen USB. O Utilizador escolhe o ficheiro que deseja imprimir. O Sistema apresenta a informação completa do ficheiro selecionado e pede confirmação. O utilizador confirma o ficheiro selecionado. O Sistema inicia o processo de validação dos periféricos da impressora(filamento, extrusores). É iniciado o processo de aquecimento da impressora. Quando a impressora estiver à temperatura ideal dá-se o início da impressão. Depois da impressão estar concluída começa o processo de arrefecimento da impressora. Depois da impressora arrefecer o suficiente o sistema apresenta o relatório da impressão, pede ao utilizador para retirar a impressão da mesa e proceder à limpeza da mesma.

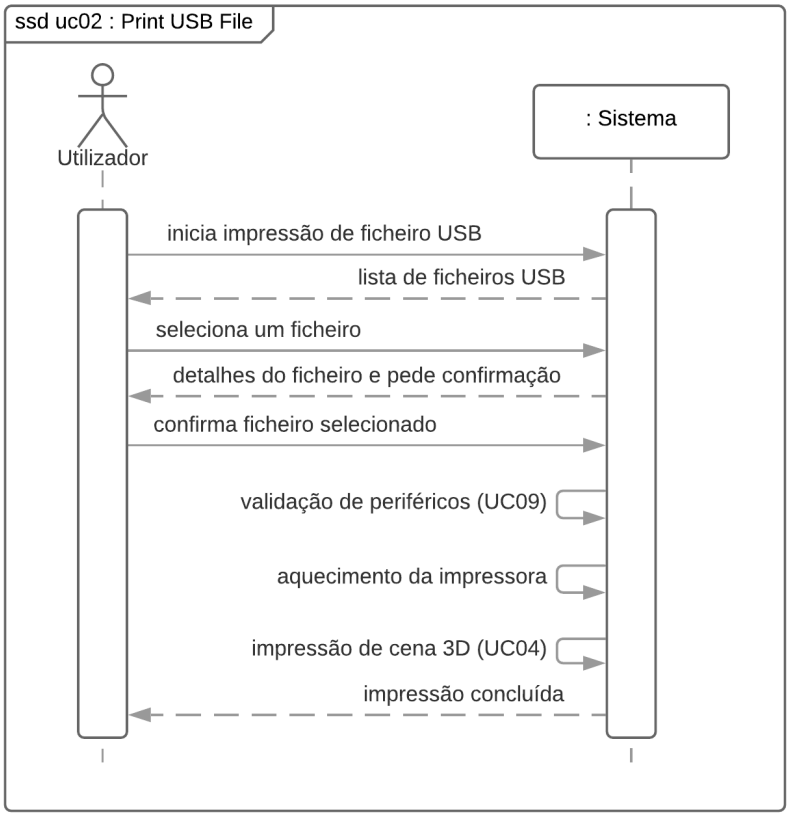


FIGURA A.4: SSD do UC02: Iniciar Impressão a Partir de uma pen USB.

A.3.2.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: processo completamente automático depois do utilizador mandar imprimir.

Pré-condições

A impressora está no estado **Ligado**. A impressora têm uma pen USB ligada. A pen USB tem pelo menos um cena 3D já organizada com os objetos que o Utilizador deseja imprimir.

Cenário de sucesso principal (ou fluxo básico)

1. O utilizador escolhe a opção imprimir, USB utilizando a GUI BEE2B.
2. O Sistema apresenta uma lista de ficheiros disponíveis na pen USB.
3. O Utilizador escolhe o ficheiro que deseja imprimir.
4. O Sistema apresenta a informação completa do ficheiro selecionado e pede confirmação.
5. O utilizador confirma o ficheiro selecionado.
6. O sistema inicia o processo de validação dos periféricos (UC9 - A.3.9).
7. O Sistema inicia o processo de aquecimento da impressora.
8. O Sistema inicia o processo de impressão da cena 3D (UC4 - A.3.4).
9. O Sistema concluí o processo de impressão;
10. O Sistema inicia o processo de arrefecimento da impressora e informa o utilizador do resumo da impressão e informações adicionais/complementares.
11. O Sistema apresenta uma mensagem para informar o utilizador que pode retirar a impressão e limpar a mesa;
12. O utilizador remove a impressão e limpa a mesa e confirma a operação;

Extensões (ou fluxos alternativos)

a. A qualquer momento o utilizador solicita o cancelamento da impressão.

i. O caso de uso termina.

b. A qualquer momento o utilizador clica no botão *home*.

i. Apresentada a vista *home*, mas a impressão continua normalmente.

ii. O Sistema possibilita ao utilizador adicionar uma nova impressão à lista de impressão ou ver o estado da impressão atual.

1a. Durante a impressão a opção imprimir passa a "adicionar à fila de impressão".

i. O caso de uso continua nos pontos 2, 3, 4 e 5. 2. O caso de uso continua no ponto 6. quando o trabalho de impressão iniciar a impressão.

1b. O utilizador escolhe a opção "ver impressão".

i. É mostrado o ecrã correspondente ao estado da impressão (Em Aquecimento, A Imprimir, Em Arrefecimento)

2a. Não existem ficheiros na drive USB.

i. O caso de uso termina.

12a. Se existir alguma trabalho de impressão na fila de impressão.

i. O caso de uso continua no ponto 6.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

frequente: este caso de uso é um dos mais frequentes;

Questões em aberto

- Como é que o utilizador irá ser notificado da conclusão da impressão?
- A notificação será enviada por correio eletrónico?
- Que dados devem constar da mensagem a enviar?

A.3.3 UC03: Iniciar impressão de um ficheiro recente

A.3.3.1 Formato Breve

O Utilizador inicia a impressão a partir do aplicação gráfica da impressora escolhendo a opção Ficheiros Recentes. O Sistema apresenta uma lista de ficheiros disponíveis no cartão SD da impressora. O Utilizador escolhe o ficheiro que deseja imprimir. O Sistema apresenta a informação completa do ficheiro selecionado e pede confirmação. O utilizador confirma o ficheiro selecionado. O Sistema inicia o processo de validação dos periféricos da impressora(filamento, extrusores). É iniciado o processo de aquecimento da impressora. Quando a impressora estiver à temperatura ideal dá-se o início da impressão. Depois da impressão estar concluída começa o processo de arrefecimento da impressora. Depois da impressora arrefecer o suficiente o sistema apresenta o relatório da impressão, pede ao utilizador para retirar a impressão da mesa e proceder à limpeza da mesma.

A.3.3.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: processo completamente automático depois do utilizador mandar imprimir.

Pré-condições

A impressora está no estado **Ligado**. O cartão SD da impressora tem pelo menos uma cena 3D já organizada com os objetos que o Utilizador deseja imprimir.

Cenário de sucesso principal (ou fluxo básico)

1. O utilizador escolhe a opção imprimir, Ficheiros Recentes utilizando a GUI BEE2B.
2. O Sistema apresenta uma lista de ficheiros disponíveis no cartão SD.
3. O Utilizador escolhe o ficheiro que deseja imprimir.

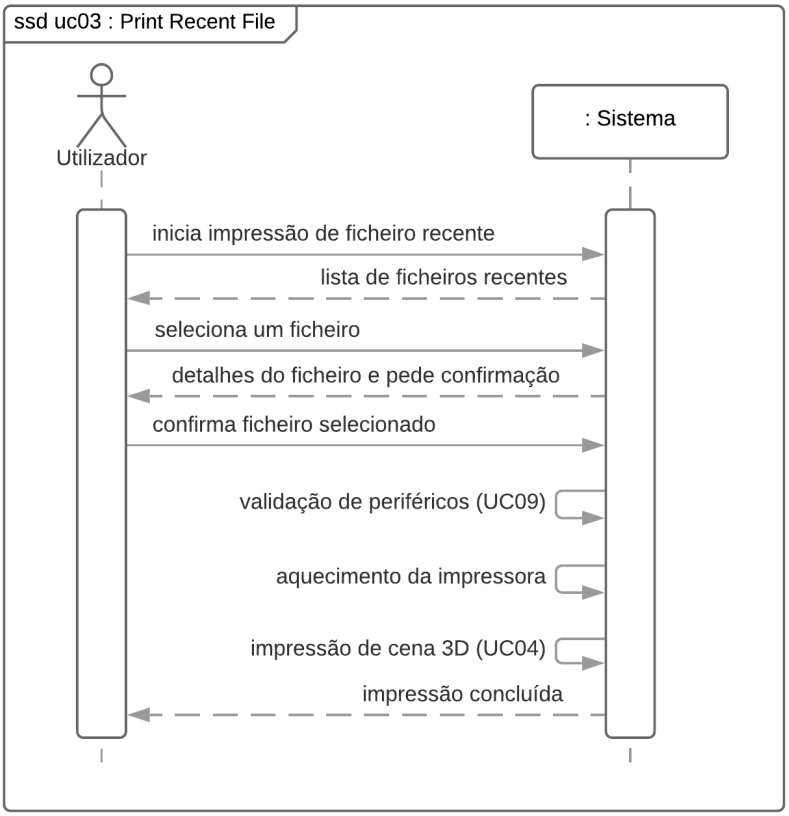


FIGURA A.5: SSD do UC03: Iniciar impressão de um ficheiro recente.

4. O Sistema apresenta a informação completa do ficheiro selecionado e pede confirmação.
5. O utilizador confirma o ficheiro selecionado.
6. O sistema inicia o processo de validação dos periféricos (UC9 - A.3.9).
7. O Sistema inicia o processo de aquecimento da impressora.
8. O Sistema inicia o processo de impressão da cena 3D (UC4 - A.3.4).
9. O Sistema concluí o processo de impressão;
10. O Sistema inicia o processo de arrefecimento da impressora e informa o utilizador do resumo da impressão e informações adicionais/complementares.
11. O Sistema apresenta uma mensagem para informar o utilizador que pode retirar a impressão e limpar a mesa;
12. O utilizador remove a impressão e limpa a mesa e confirma a operação;

Extensões (ou fluxos alternativos)

a. A qualquer momento o utilizador solicita o cancelamento da impressão.

   i. O caso de uso termina.

b. A qualquer momento o utilizador clica no botão *home*.

i. Apresentada a vista *home*, mas a impressão continua normalmente.

ii. O Sistema possibilita ao utilizador adicionar uma nova impressão à lista de impressão ou ver o estado da impressão atual.

1a. Durante a impressão a opção imprimir passa a "adicionar à fila de impressão".

i. O caso de uso continua nos pontos 2, 3, 4 e 5. 2. O caso de uso continua no ponto 6. quando o trabalho de impressão iniciar a impressão.

1b. O utilizador escolhe a opção "ver impressão".

i. É mostrado o ecrã correspondente ao estado da impressão (Em Aquecimento, A Imprimir, Em Arrefecimento)

2a. Não existem ficheiros na drive USB.

i. O caso de uso termina.

12a. Se existir alguma trabalho de impressão na fila de impressão.

i. O caso de uso continua no ponto 6.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

frequente: este UC é um dos mais frequentes;

Questões em aberto

- Como é que o utilizador irá ser notificado da conclusão da impressão?
- A notificação será enviada por correio eletrónico?
- Que dados devem constar da mensagem a enviar?

A.3.4 UC04: Impressão de uma cena 3D

A.3.4.1 Formato Breve

O Sistema inicia a impressão da cena 3D. O Sistema renderiza a cena 3D e mostra-a ao utilizador. O Sistema termina a impressão da cena 3D.

A.3.4.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: visualização da renderização da cena 3D para poder remover um objeto da impressão.

Pré-condições

- A impressora está no estado **Ligado**.
- A cena 3D já está carregada no cartão SD da impressora e as ferramentas da impressora já estão à temperatura correta

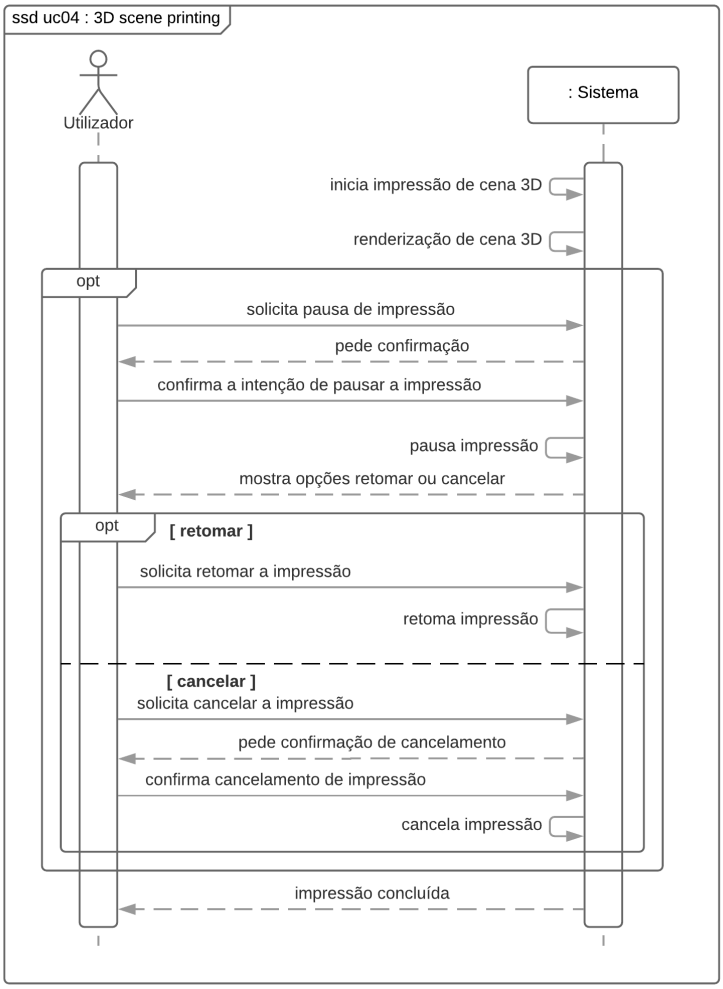


FIGURA A.6: SSD do UC04: Impressão de uma cena 3D.

Cenário de sucesso principal (ou fluxo básico)

1. O Sistema inicia a impressão da cena 3D.
2. O Sistema renderiza a cena 3D.
3. O Sistema mostra a cena 3D ao utilizador.
4. O Sistema termina a impressão da cena 3D.

Extensões (ou fluxos alternativos)

3a. O utilizador solicita o cancelamento da impressão.

i. O sistema pede confirmação ao utilizador.

ii. O utilizador confirma a intenção de cancelar a impressão.

iii. A impressão é cancelada.

iv. O caso de uso termina.

3b. O utilizador solicita a pausa da impressão.

i. O Sistema coloca a impressão em pausa.

ii. O Sistema apresenta as opções de cancelar ou retomar ao utilizador.

iii. O utilizador escolhe retomar a impressão.

iv. O Sistema retoma a impressão anteriormente pausada.

3c. O utilizador solicita a remoção do objeto selecionado da impressão (UC06 - A.3.6).

i. O Sistema coloca a impressão em pausa.

ii. O Sistema apresenta as opções de cancelar ou retomar ao utilizador.

iii. O utilizador escolhe retomar a impressão.

iv. O Sistema retoma a impressão anteriormente pausada.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

- dimensões reais da mesa de impressão: 350x300x300 (x,y,z)
- coordenada (0,0,0) situada no canto inferior esquerdo da mesa de impressão

Frequência de Ocorrência

- Muito frequente;
- Este use case será executado no use case: iniciar impressão a partir do PC (UC01 - A.3.1)
- Este use case será executado no use case: iniciar impressão a partir de uma drive USB (UC02 - A.3.2)
- Este use case será executado no use case: iniciar impressão de um ficheiro recente (UC03 - A.3.3)

Questões em aberto

A.3.5 UC05: Gerir Fila de Impressão

Este requisito deixou de pertencer à GUI BEE2B e passou a integrar o pluggin BEECura, assim será descrito na secção A.3.18.4.

A.3.6 UC06: Remover Objeto da Impressão

A.3.6.1 Formato Breve

O Utilizador seleciona um objeto da cena 3D renderizada. O Sistema apresenta a informação detalhada do objeto selecionado. O Utilizador solicita o cancelamento do objeto selecionado. O Sistema pede confirmação da intenção de cancelamento. O utilizador confirma a intenção de cancelar a impressão do objeto selecionado. O Sistema remove o objeto da impressão atual e continua normalmente com a impressão dos restantes objetos na cena 3D.

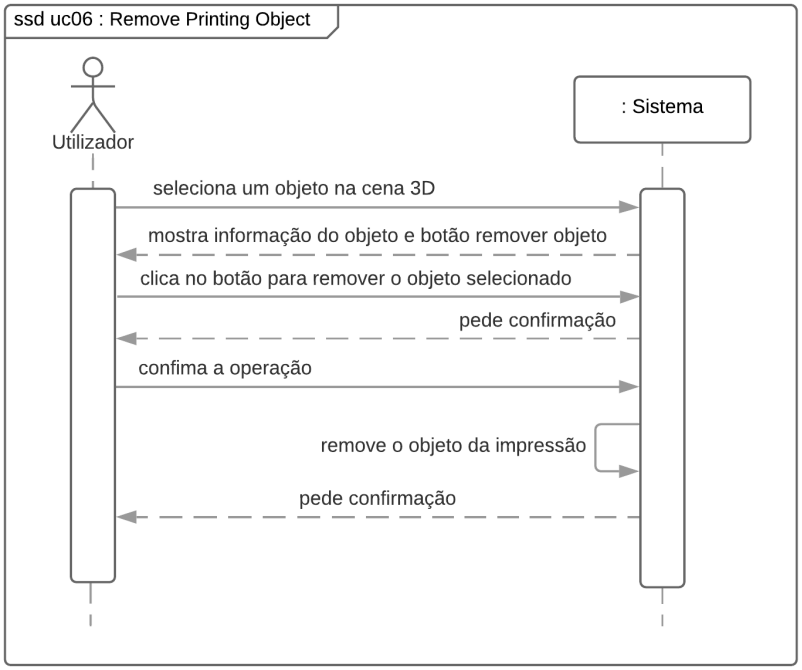


FIGURA A.7: SSD do UC06: Remover Objeto da Impressão.

A.3.6.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Em caso de falha parcial da impressão o Utilizador quer remover o objeto que está a falhar sem cancelar toda a impressão.

Pré-condições

- A impressora está no estado **A Imprimir**.
- A cena 3D da impressão em curso contém mais do que um objeto.
- O Interface Gráfica encontra-se na vista "A Imprimir"

Cenário de sucesso principal (ou fluxo básico)

1. O utilizador seleciona o objeto que deseja remover.
2. O Sistema apresenta informação detalhada desse objeto e mostra um botão para remove-lo da impressão.
3. O Utilizador clica no botão para remover o objeto da impressão.
4. O Sistema pede confirmação.
5. O utilizador confirma a operação.
6. O sistema remove o objeto da impressão, recria a renderização/tabuleiro de impressão e mostra o novo progresso.

Extensões (ou fluxos alternativos)

5a. O utilizador cancela a operação.

i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- quando existir uma falha parcial de impressão.
- Este use case pode ser executado no use case: impressão de cena 3D (UC04 - A.3.4)

Questões em aberto

A.3.7 UC07: Substituir Filamento

A.3.7.1 Formato Breve

O Utilizador seleciona um filamento. O Sistema mostra informação detalhada do filamento selecionado O Utilizador solicita a substituição de filamento. O Sistema pede confirmação da operação O Utilizador confirma a intenção de continuar com a operação. O sistema aquece o extrusor do filamento selecionado. O sistema descarrega o filamento do extrusor assim que este estiver à temperatura indicada. O utilizador coloca o novo filamento na impressora- O sistema deteta o novo filamento e inicia o seu carregamento. No final do carregamento do filamento o Sistema pede ao utilizador para remover o material extrudido que se encontra na mesa de impressão. O Utilizador remove o material extrudido da mesa de impressão e conclui o processo de troca de filamento.

A.3.7.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja trocar o filamento de um determinado extrusor de forma rápida e segura

Pré-condições

- A impressora está no estado **Ligado**.
- A impressora têm pelo menos um extrusor carregado, ou seja, está no estado **Pronto**.
- O Utilizador está na Vista Manutenção.

Cenário de sucesso principal (ou fluxo básico)

1. O utilizador seleciona o filamento que deseja carregar ou substituir.
2. O Sistema apresenta informação detalhada desse filamento e mostra um botão para poder substituí-lo.
3. O Utilizador clica no botão para substituir o filamento selecionado.
4. O Sistema pede confirmação.
5. O utilizador confirma a operação.

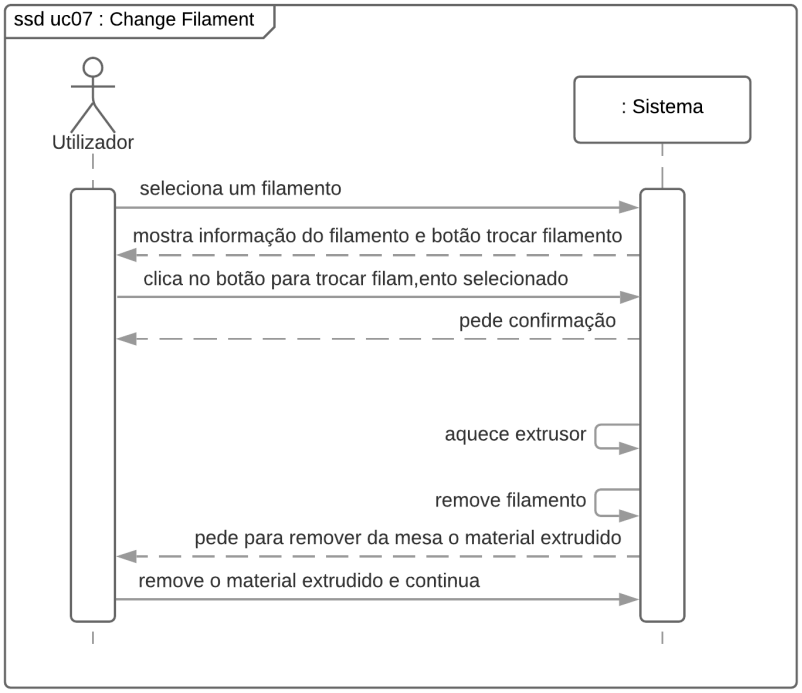


FIGURA A.8: SSD do UC07: Substituir Filamento.

6. O sistema aquece o extrusor do filamento selecionado e inicia a a sua remoção quando estiver à temperatura ideal.
7. O Utilizador coloca o novo filamento na impressora.
8. O Sistema deteta o novo filamento e inicia o seu carregamento.
9. No final do carregamento o Sistema pede ao Utilizador para remover o material extrudido que se encontra na mesa de impressão.
10. O Utilizador remove o material extrudido da mesa de impressão e conclui o processo de troca de filamento.

Extensões (ou fluxos alternativos)

3a. Não existe nenhum filamento carregado no extrusor.

i. O caso de uso continua no ponto 7.

4a. O Utilizador cancela a operação.

8a. O novo filamento não é automaticamente detetado.

i. O Utilizador seleciona o filamento introduzido na impressora.

ii. O Sistema inicia o carregamento do novo filamento.

iii. O caso de uso continua no ponto 9.

8b. O tipo de material do novo filamento é diferente do material anteriormente utilizado pelo extrusor.

i. O Sistema alerta o Utilizador que o material que este está a tentar carregar é diferente do último material utilizado e pede confirmação.

ii. O Utilizador confirma a intenção de continuar a operação.

iii. O Sistema inicia o carregamento do novo filamento.

iv. O caso de uso continua no ponto 9.

iia. O Utilizador cancela a operação.

1. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre que o utilizador desejar trocar de filamento.
- Este use case será executado no assistente de substituição de Extrusor (UC8 - A.3.8)
- Este use case será executado no assistente de configuração inicial da impressora (UC10 - A.3.10)

Questões em aberto

A.3.8 UC08: Substituir Extrusor

A.3.8.1 Formato Breve

O Utilizador seleciona um extrusor. O Sistema mostra informação detalhada do extrusor selecionado. O Utilizador solicita a substituição de extrusor. O Sistema pede confirmação da operação O Utilizador confirma a intenção de continuar com a operação. O Sistema aquece o extrusor selecionado. O Sistema descarrega o filamento do extrusor assim que este estiver à temperatura indicada. O Sistema prepara a remoção do extrusor e pede ao Utilizador para substituir o extrusor. O Utilizador remove o extrusor da Impressora . O Utilizador insere o novo extrusor na Impressora e clica OK (indicando que já não vai mexer na mecânica da máquina). O Sistema deteta o novo extrusor e inicia um assistente de substituição de Filamento (UC7 - A.3.7). No final do assistente de troca de filamento o Utilizador clica continuar. O Sistema inicia um assistente de calibração *offset* Z (UC17 - A.3.16) para o novo extrusor. No final do processo de calibração do *offset* Z do novo extrusor o Utilizador clica continuar. O Sistema inicia um assistente de calibração *offset* XY (UC18 - A.3.17) para o novo extrusor.

A.3.8.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja trocar o extrusor de forma rápida e segura.

Pré-condições

- A impressora está no estado **Ligado**.

Cenário de sucesso principal (ou fluxo básico)

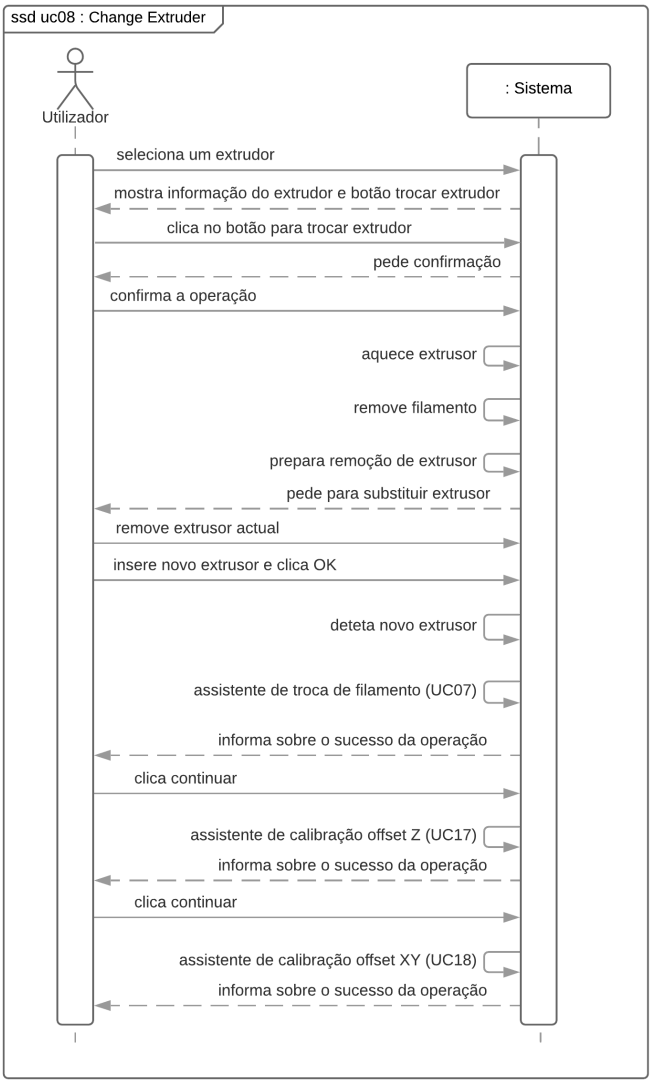


FIGURA A.9: SSD do UC08: Substituir Extrusor.

1. O Utilizador seleciona o extrusor que deseja carregar ou substituir.
2. O Sistema apresenta informação detalhada desse extrusor e mostra um botão para poder substituí-lo.
3. O Utilizador clica no botão para substituir o extrusor selecionado.
4. O Sistema pede confirmação.
5. O Utilizador confirma a operação.
6. O Sistema aquece o extrusor selecionado e inicia a sua remoção do filamento nele carregado quando estiver à temperatura ideal.
7. O Sistema prepara a remoção do extrusor e pede ao Utilizador para substituir o extrusor.
8. O Utilizador remove o extrusor da Impressora.

9. O Utilizador insere o novo extrusor na Impressora e clica OK (indicando que já não vai mexer na mecânica da máquina).
10. O Sistema deteta o novo extrusor e inicia um assistente de substituição de Filamento (UC7 - A.3.7)
11. O Sistema informa sobre o sucesso da operação
12. O Utilizador clica continuar.
13. O Sistema inicia um assistente de calibração *offset* Z (UC17 - A.3.16) e informa o Utilizador sobre o sucesso da operação.
14. O Utilizador clica continuar.
15. O Sistema inicia um assistente de calibração *offset* XY (UC18 - A.3.17) e informa o Utilizador sobre o sucesso da operação.

Extensões (ou fluxos alternativos)

3a. Não existe nenhum extrusor carregado no *slot* selecionado.

i. O Sistema prepara o *slot* do extrusor e pede ao Utilizador para inserir o extrusor.

ii. O caso de uso continua no ponto 9.

4a. O Utilizador cancela a operação.

i. O caso de uso termina.

6a. Não existe nenhum filamento carregado no extrusor selecionado.

i. O caso de uso continua no ponto 7.

10a. Se for a configuração inicial da impressora

i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre que o utilizador desejar trocar de extrusor.
- Este use case será executado no assistente de configuração inicial da impressora (UC10 - A.3.10)

Questões em aberto

A.3.9 UC09: Validação de Periféricos

A.3.9.1 Formato Breve

O Sistema valida os extrusores Instalados. O Sistema valida os Filamentos Instalados. O Sistema apresenta um resumo da validação caso seja necessária alguma alteração e pede confirmação. O Utilizador confirma a intenção de continuar a operação. O Sistema apresenta um assistente de substituição de Extrusor (UC8 - A.3.8) para cada extrusor incorreto. O Sistema apresenta um assistente de substituição de Filamento (UC7 - A.3.7) para cada filamento incorreto.

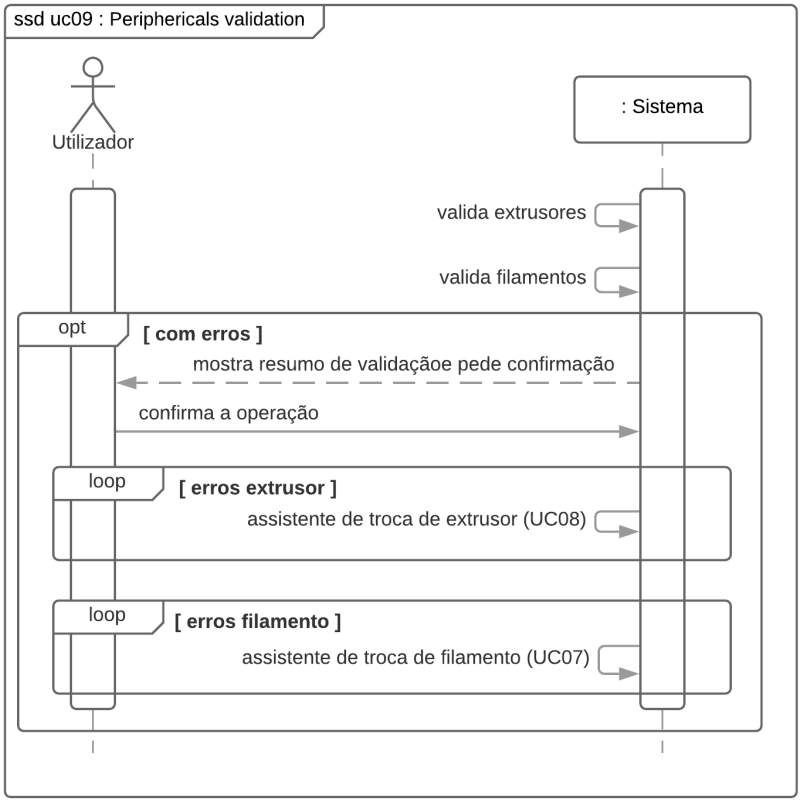


FIGURA A.10: SSD do UC09: Validação de Periféricos.

A.3.9.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja ser notificado caso algum periférico não esteja correto (extrusor ou filamento).

Pré-condições

- Uma impressão foi iniciado pelo Utilizador

Cenário de sucesso principal (ou fluxo básico)

1. O Sistema valida os extrusores Instalados.
2. O Sistema valida os filamentos Instalados.
3. O Sistema apresenta um resumo da validação caso seja necessária alguma alteração e pede confirmação.
4. O Utilizador confirma a operação.
5. O Sistema apresenta um assistente de substituição de extrusor (UC8 - A.3.8) para cada extrusor incorreto.

6. O Sistema apresenta um assistente de substituição de filamento (UC7 - A.3.7) para cada filamento incorreto.

Extensões (ou fluxos alternativos)

4a. O Utilizador clica no botão de troca filamento de um extrusor.

i. O Sistema apresenta um assistente de substituição de Filamento (UC7 - A.3.7) para o filamento selecionado.

ii. O caso de uso continua no ponto 3.

4b. O Utilizador cancela a operação.

i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre que o for iniciada uma impressão.
- Este use case será executado no use case: iniciar impressão a partir do PC (UC01 - A.3.1)
- Este use case será executado no use case: iniciar impressão a partir de uma drive USB (UC02 - A.3.2)
- Este use case será executado no use case: iniciar impressão de um ficheiro recente (UC03 - A.3.3)

Questões em aberto

A.3.10 UC10: Configuração Incial da Impressora

A.3.10.1 Formato Breve

O Utilizador liga a impressora pela primeira vez. O Sistema inicia o processo de mudança de idioma (UC16 - A.3.15) O Sistema apresenta o ecrã de boas vindas. O Sistema inicia um assistente de substituição de extrusor (UC8 - A.3.8) para cada extrusor. O Sistema inicia um assistente de substituição de filamento (UC7 - A.3.7) para cada extrusor instalado. O Sistema inicia um assistente de calibração *offset* Z (UC17 - A.3.7) para cada extrusor instalado. O Sistema inicia um assistente de calibração *offset* XY (UC18 - A.3.17) para cada extrusor instalado. O Sistema inicia o assistente de configuração de rede (UC11 - A.3.11). O Sistema pergunta ao Utilizador se quer emparelhar algum PC O Utilizador inicia o assistente de emparelhamento do PC (UC14 - A.3.14). O Sistema pergunta ao Utilizador se quer fazer uma impressão de teste. O Utilizador inicia uma impressão de teste. O Sistema pergunta ao Utilizador se quer repetir a impressão de teste.

A.3.10.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja configurar a impressora de forma assistida e intuitiva.

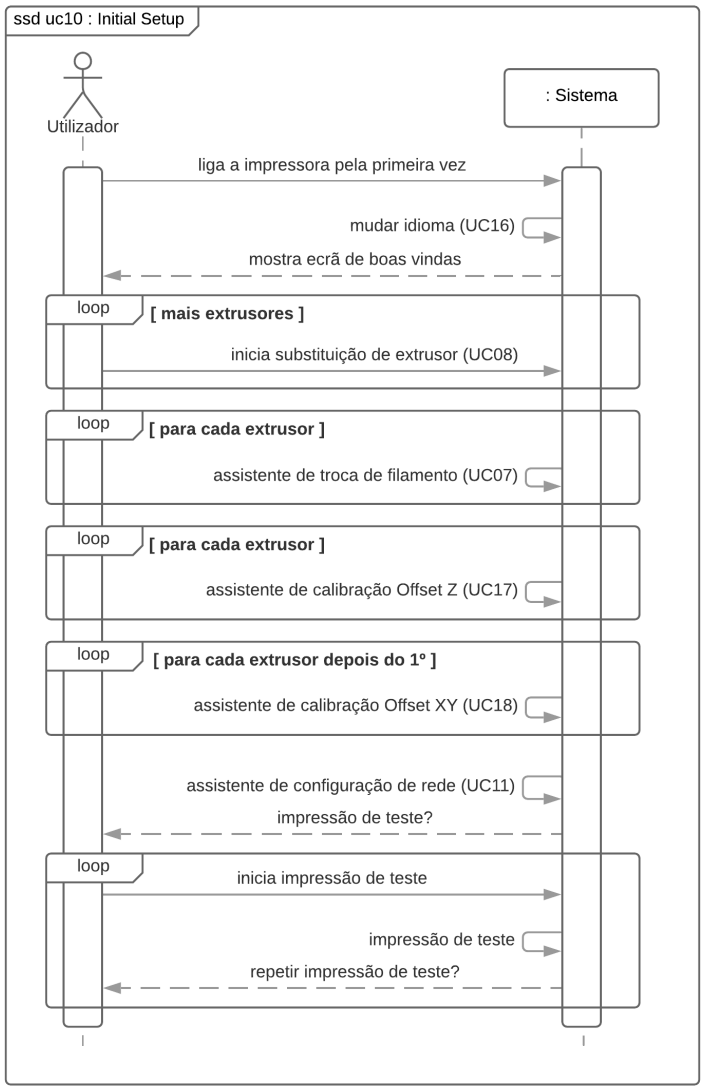


FIGURA A.11: SSD do UC10: Configuração Incial da Impressora.

Pré-condições

- É a primeira utilização da impressora.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador liga a impressora pela primeira vez.
2. O Sistema inicia o processo de mudança de idioma (UC16 - A.3.15)
3. O Sistema apresenta o ecrã de boas vindas.
4. O Sistema inicia um assistente de substituição de extrusor (UC8 - A.3.8). O Utilizador repete o passo 4. até ter no máximo 4 extrusores instalados
5. O Sistema inicia um assistente de substituição de filamento (UC7 - A.3.7). O Sistema repete o passo 5. para cada extrusor instalado.

6. O Sistema inicia um assistente de calibração *offset* Z (UC17 - A.3.16) O Sistema repete o passo 6. para cada extrusor instalado.
7. O Sistema inicia um assistente de calibração *offset* XY (UC18 - A.3.17) O Sistema repete o passo 7. para cada extrusor instalado depois do primeiro.
8. O Sistema inicia o assistente de configuração de rede (UC11 - A.3.11).
9. O Sistema pergunta ao Utilizador se quer emparelhar algum PC
10. O Utilizador inicia o assistente de emparelhamento do PC (UC14 - A.3.14).
11. O Sistema pergunta ao Utilizador se quer fazer uma impressão de teste.
12. O Utilizador inicia uma impressão de teste.
13. O Sistema inicia uma impressão de teste.
14. O Sistema pergunta ao Utilizador se quer repetir a impressão de teste.
15. O Utilizador termina a configuração inicial.

Extensões (ou fluxos alternativos)

1a. Não é a primeira utilização da impressora.

i. O caso de uso termina.

4a. Depois do utilizador instalar o primeiro extrusor decide não adicionar mais nenhum extrusor

i. O caso de uso continua no ponto 5.

10a. O Utilizador decide não emparelhar nenhum PC.

i. O caso de uso continua no ponto 11.

12a. O Utilizador decide não efetuar a impressão de teste.

i. O caso de uso termina.

15a. O Utilizador decide repetir a impressão de teste.

i. O caso de uso continua no ponto 13.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- pouco frequente: apenas na primeira utilização da impressora.

Questões em aberto

A.3.11 UC11: Modificar configurações de rede

A.3.11.1 Formato Breve

O Utilizador inicia o assistente de configuração de Rede. O Sistema mostra o nome da impressora, informação de rede e PCs emparelhados. O Utilizador inicia a ligação a uma rede. O Sistema gera um *hotspot* e apresenta o endereço e password de acesso ao Utilizador. O Utilizador liga-se à rede através de um *browser*. O Sistema apresenta o sucesso da operação.

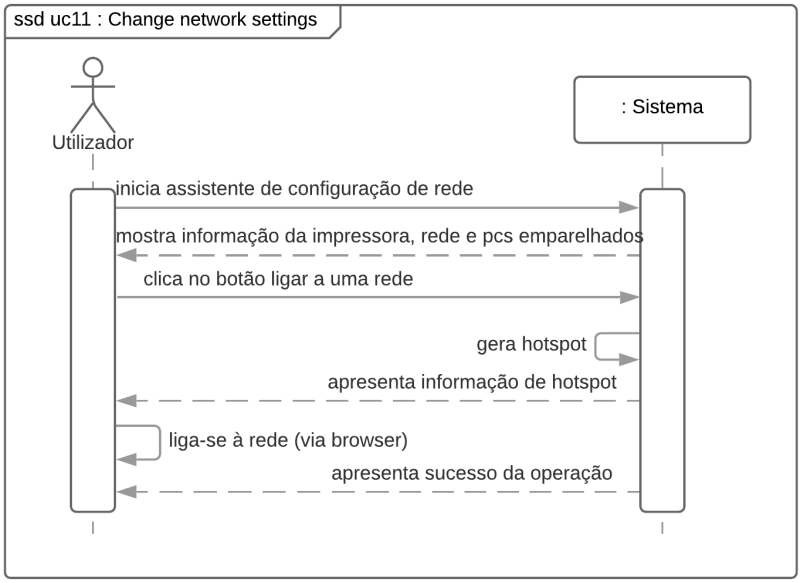


FIGURA A.12: SSD do UC11: Modificar configurações de rede.

A.3.11.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja ligar a impressora de forma fácil e sem usar um teclado de ecrã.

Utilizador: Deseja ter informação sobre a rede a que a impressora está ligada.

Utilizador: Deseja ter informação sobre os PCs emparelhados com a impressora.

Pré-condições

- A impressora está no estado **Ligado**.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o assistente de configuração de Rede.
2. O Sistema mostra o nome da impressora, informação de rede e PCs emparelhados.
3. O Utilizador clica em ligar a uma rede.
4. O Sistema gera um *hotspot* e apresenta o endereço e password de acesso ao Utilizador.
5. O Utilizador liga-se à rede através de um *browser*
6. O Sistema apresenta o sucesso da operação.

Extensões (ou fluxos alternativos)

3a. A impressora já está ligada a uma rede

i. O Utilizador clica em ligar a outra rede. ii. O Sistema avisa que irá perder a ligação à rede atual e pede confirmação. iii. O Utilizador confirma a operação. iv. O caso de uso continua no ponto 4.

4a. O Utilizador cancela a operação

i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre que for necessário ligar a uma rede Wi-Fi

Questões em aberto

A.3.12 UC12: Restaurar software de fábrica

A.3.12.1 Formato Breve

O Utilizador inicia o processo de reposição de fábrica. O Sistema alerta para os danos do processo e pede confirmação. O Utilizador confirma a intenção da reposição de fábrica. O Sistema restaura o software de fábrica.

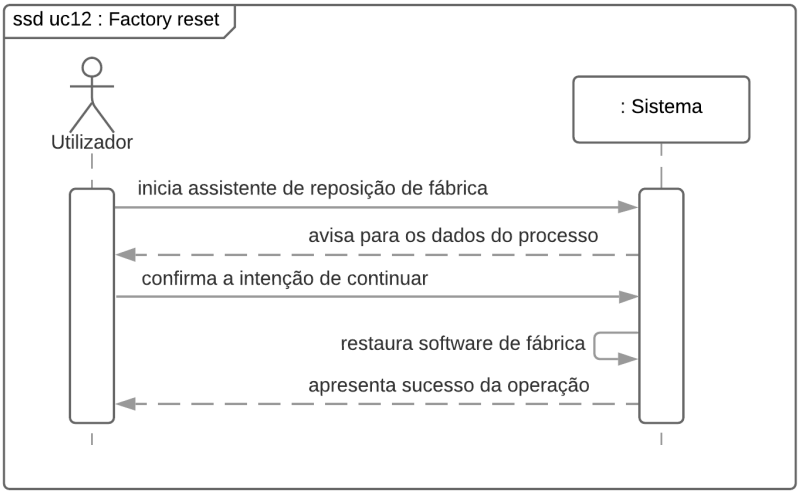


FIGURA A.13: SSD do UC12: Restaurar software de fábrica.

A.3.12.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja restaurar o software de forma fácil.

Pré-condições

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o assistente de reposição de fábrica.
2. O Sistema alerta para os danos deste processo e pede confirmação.
3. O Utilizador confirma a intenção de continuar a operação
4. O Sistema restaura o software de fábrica.
5. O Sistema informa o Utilizado do sucesso da operação.

Extensões (ou fluxos alternativos)

a. A qualquer momento o utilizador solicita o cancelamento da operação.

i. O caso de uso termina.

5a. A operação não é concluída ou é concluída com erros.

i. O Sistema repõe o software atual.

ii. O Sistema alerta o utilizador da falha da operação.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- pouco frequente: este use case será apenas utilizado pontualmente para resolver problemas inesperados no software.

Questões em aberto

A.3.13 UC13: Atualizar software

A.3.13.1 Formato Breve

O Sistema deteta automaticamente uma nova versão e faz a transferência da mesma. O Sistema notifica o utilizador que existe uma atualização nova. O Utilizador inicia o assistente de atualização de software. O Sistema dá informação sobre o processo e pede confirmação. O Utilizador confirma a intenção de atualizar o software. O Sistema faz a atualização do software. O Sistema informa o Utilizador do sucesso da Operação.

A.3.13.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja atualizar o software de forma rápida, segura e sem perda de informação.

Pré-condições

- Existe uma nova versão do software.
- O software atual já efetuou a transferência da nova versão.

Cenário de sucesso principal (ou fluxo básico)

1. O Sistema deteta automaticamente uma nova versão e faz a transferência da mesma.
2. O Sistema notifica o utilizador que existe uma atualização nova.

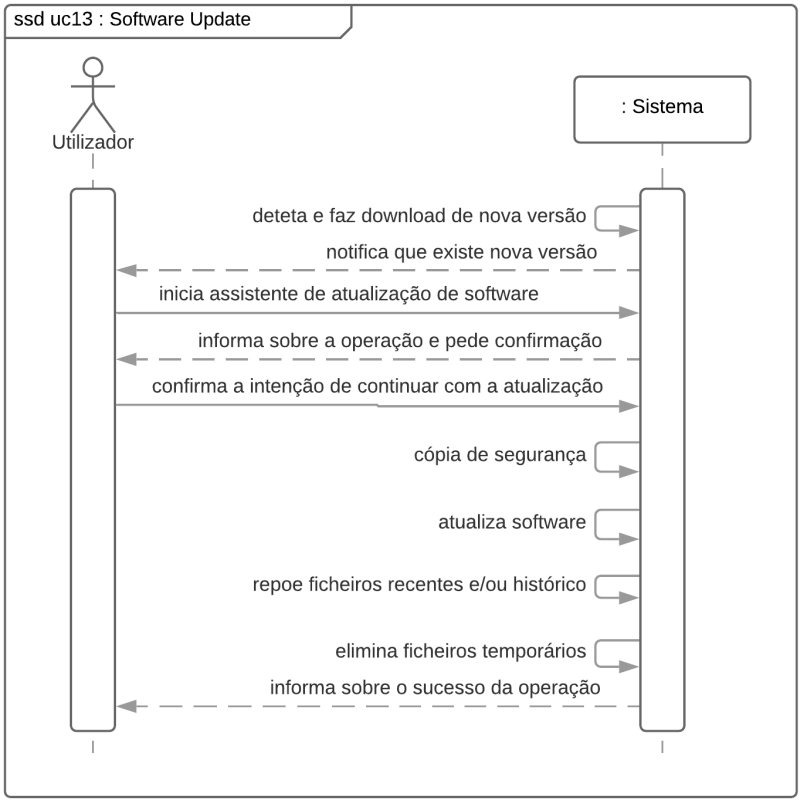


FIGURA A.14: SSD do UC13: Atualizar software.

3. O Utilizador inicia o assistente de atualização de software.
4. O Sistema dá informação sobre o processo e pede confirmação.
5. O Utilizador confirma a intenção de atualizar o software.
6. O Sistema faz uma cópia de segurança do software atual.
7. O Sistema faz a atualização do software.
8. O Sistema repõe os ficheiros recentes e histórico da impressora.
9. O Sistema elimina os ficheiros temporários.
10. O Sistema informa o Utilizador do sucesso da Operação.

Extensões (ou fluxos alternativos)

4a. Não existe nenhuma atualização disponível.

i. O Sistema avisa o utilizador que o software já está atualizado. ii. O caso de uso termina.

5a. O Utilizador cancela a operação

i. O caso de uso termina.

6a. O Sistema falha a efetuar a cópia de segurança.

i. O Sistema alerta o utilizador que o software não pode ser atualizado devido a uma falha na cópia de segurança.

ii. O caso de uso termina.

7a. O Sistema falha a efetuar a atualização do software.

i. O Sistema alerta o utilizador que ocorreu um erro ao tentar atualizar o software.

ii. O caso de uso termina.

8a. O Sistema falha a reposição dos ficheiros recentes e/ou histórico.

i. O Sistema alerta o utilizador que ocorreu um erro ao tentar repor o histórico e/ou ficheiros recentes e pede confirmação para continuar.

ii. O Utilizador confirma a intenção de continuar a operação.

iii. O caso de uso continua no ponto 9.

iia. O Utilizador cancela a operação.

1. O Sistema repõe o software a versão atual e informa o utilizador que a operação foi cancelada.

2. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre que existir uma nova atualização do software.

Questões em aberto

A.3.14 UC14: Emparelhar PC

A.3.14.1 Formato Breve

O utilizador inicia processo de emparelhamento na GUI BEE2B. O sistema mete a impressora em modo de emparelhamento e fica à espera de ligações. O Utilizador efetua o emparelhamento no PC selecionando a impressora pretendida. O Sistema pede confirmação ao utilizador. O utilizador confirma a intenção de emparelhar a impressora na GUI BEE2B. O sistema emparelha o PC com a Impressora.

A.3.14.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja emparelhar o PC com a impressora de forma simples e segura.

Pré-condições

- A impressora e o PC estão ligados à mesma rede.

Cenário de sucesso principal (ou fluxo básico)

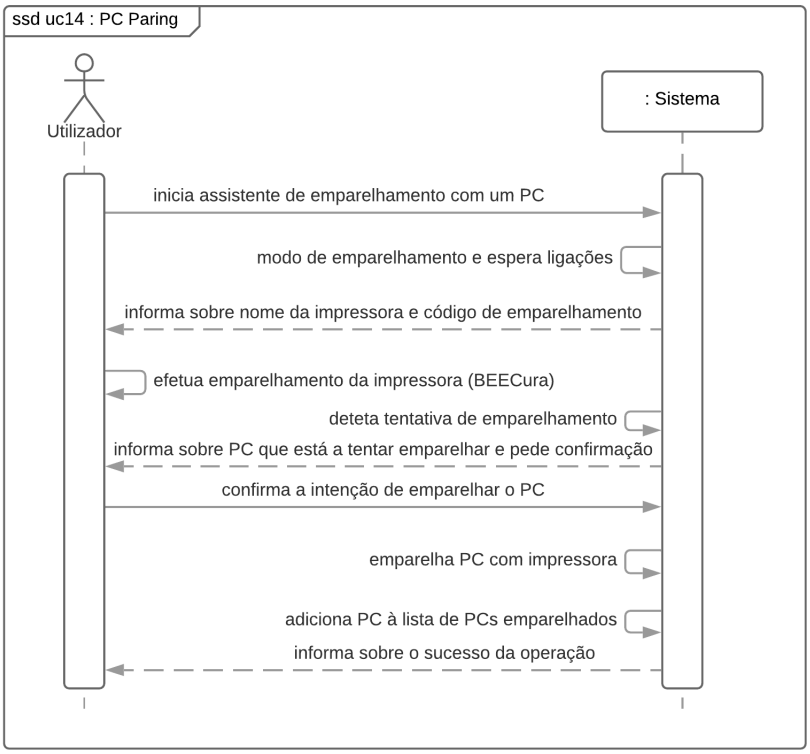


FIGURA A.15: SSD do UC14: Emparelhar PC.

1. O utilizador inicia o assistente de emparelhamento com o PC.
2. O sistema coloca a impressora em modo de emparelhamento e fica à espera de ligações.
3. O Sistema informa o utilizador sobre o nome da impressora e o código de emparelhamento.
4. O Utilizador efetua o emparelhamento no PC selecionando a impressora pretendida e
5. inserindo o código da mesma (BEECura).
6. O Sistema deteta a tentativa de emparelhamento.
7. O Sistema informa o utilizador que um determinado PC está a tentar o emparelhamento com a impressora e pede confirmação ao Utilizador.
8. O Utilizador confirma a intenção de emparelhar a impressora com o PC.
9. O Sistema emparelha o PC com a Impressora e adiciona-o à lista de PCs emparelhados.
10. O Sistema informa o utilizador do sucesso da operação.

Extensões (ou fluxos alternativos)

2a. Não existe nenhuma tentativa de ligação durante um minuto.

i. O caso de uso termina.

7a. O Utilizador cancela a operação

i. O caso de uso termina.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Poderá ser executado no assistente de configuração inicial da impressora (UC10 - A.3.10)

Questões em aberto

- Quanto tempo o Sistema fica à espera no ponto 2.?
  - para já um minuto
- Ao mudar de rede elimina os emparelhamentos anteriores?
  - manter PCs emparelhados independentemente da rede

A.3.15 UC16: Mudar Idioma

A.3.15.1 Formato Breve

O Utilizador inicia a mudança de idioma. O Sistema muda os seus diálogos para o idioma pretendido.

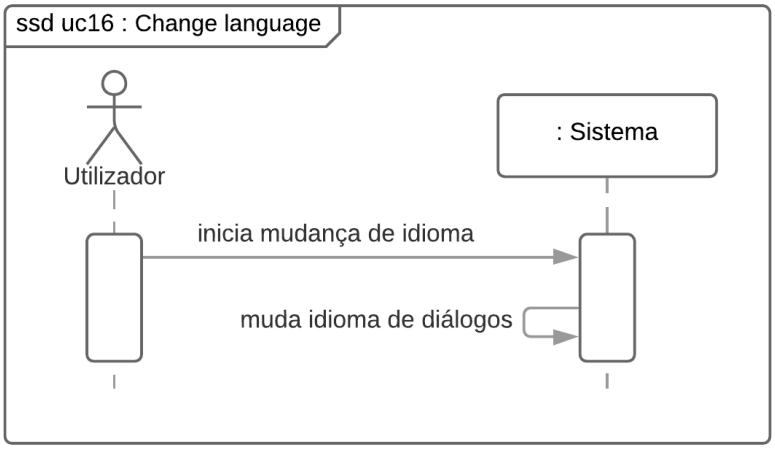


FIGURA A.16: SSD do UC16: Mudar Idioma.

A.3.15.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja utilizar o Sistema em Português ou em Inglês.

Pré-condições

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia a mudança de idioma.
2. O Sistema muda os seus diálogos para o idioma pretendido.

Extensões (ou fluxos alternativos)

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- pouco frequente: este use case será apenas utilizado pontualmente.
- Este use case será executado no assistente de configuração inicial da impressora (UC10 - A.3.10)

Questões em aberto

A.3.16 UC17: Calibrar *offset* Z

A.3.16.1 Formato Breve

O Utilizador inicia o assistente de calibração *offset* Z. O Sistema mostra as instruções da calibração ao utilizador. O Utilizador efetua a calibração do *offset* Z e confirma a operação. O Sistema informa o Utilizador do sucesso da operação.

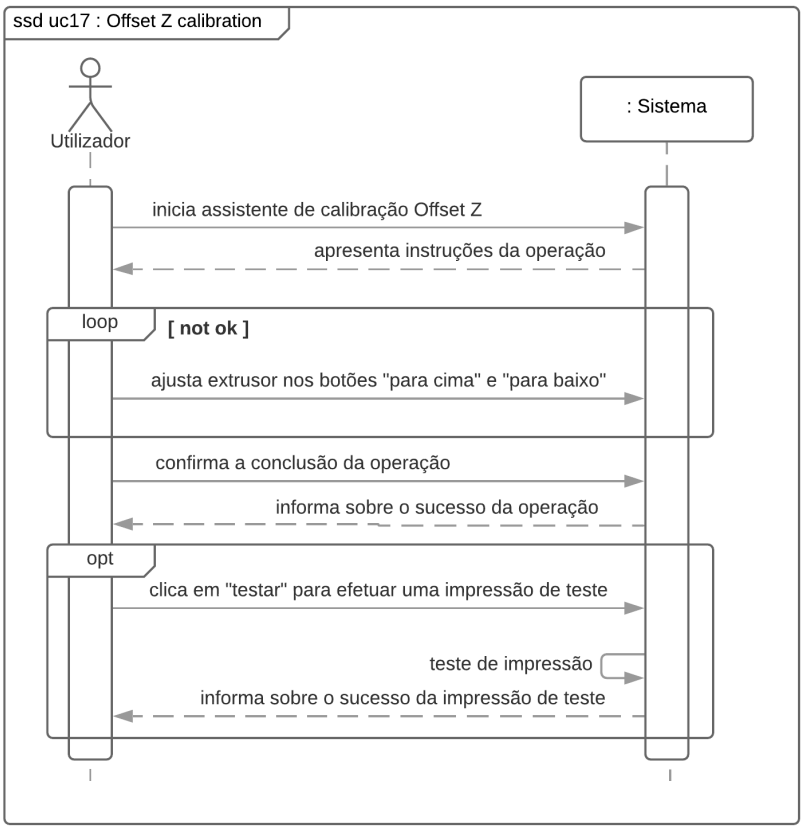


FIGURA A.17: SSD do UC17: Calibrar Offset Z.

A.3.16.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja calibrar o *offset* Z de forma correta e intuitiva para conseguir melhores desempenhos de impressão.

Pré-condições

- A impressora têm pelo menos um extrusor instalado.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o assistente de calibração *offset* Z.
2. O Sistema mostra as instruções da calibração ao utilizador.
3. O Utilizador efetua a calibração do *offset* Z clicando nos botões "para cima"ou "para baixo".
4. O Utilizador confirma a operação.
5. O Sistema informa o Utilizador do sucesso da operação e permite ao utilizador testar a nova calibração.
6. O Utilizador clica em testar para efetuar uma impressão de teste.
7. O Sistema inicia impressão de teste e mostra progresso.
8. O Sistema mostra sucesso de impressão e permite repetir o processo de teste do ponto 6.

Extensões (ou fluxos alternativos)

1a. Não existe nenhum extrusor instalado.

i. O caso de uso termina.

6a. O Utilizador decide não efetuar o teste de impressão.

i. O caso de uso termina.

8a. O Utilizador decide efetuar novo teste de impressão.

i. O caso de uso continua no ponto 7.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre depois do Utilizador efetuar um assistente de substituição de extrusor (UC8 - A.3.8)
- Este use case será executado no assistente de configuração inicial da impressora (UC10 - A.3.10) para cada extrusor instalado
- pouco frequente: quando o Utilizador verificar fracos desempenhos de impressão.

Questões em aberto

A.3.17 UC18: Calibrar *offset* XY

A.3.17.1 Formato Breve

O Utilizador inicia o assistente de calibração *offset* XY. O Sistema imprime réguas de calibração e mostra as instruções da calibração ao utilizador e progresso de impressão. O Sistema mostra as instruções da calibração do eixo X ao utilizador. O Utilizador efetua a calibração do eixo X e confirma a operação. O Sistema mostra as instruções da calibração do eixo Y ao utilizador. O Utilizador efetua a calibração do eixo Y e confirma a operação. O Sistema informa o Utilizador do sucesso da operação.

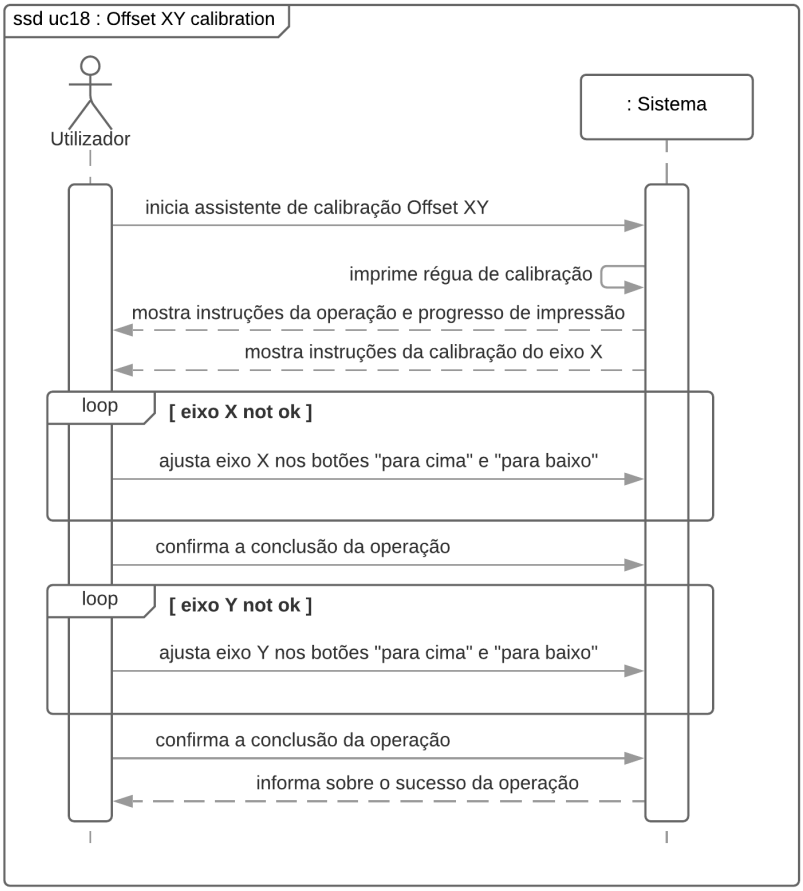


FIGURA A.18: SSD do UC18: Calibrar *offset* XY.

A.3.17.2 Formato Completo

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja calibrar o *offset* XY de forma correta e intuitiva para conseguir melhores desempenhos de impressão.

Pré-condições

- A impressora têm pelo menos dois extrusores instalados.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o assistente de calibração *offset* XY.
2. O Sistema inicia a impressão de régua de calibração.
3. O Sistema mostra as instruções da calibração XY e progresso de impressão ao utilizador.
4. O Sistema mostra as instruções da calibração do eixo X ao utilizador.
5. O Utilizador efetua a calibração do *offset* X clicando nos botões "para cima"ou "para baixo".
6. O Utilizador confirma a operação.
7. O Sistema mostra as instruções da calibração do eixo Y ao utilizador.
8. O Utilizador efetua a calibração do *offset* Y clicando nos botões "para cima"ou "para baixo".
9. O Utilizador confirma a operação.
10. O Sistema informa o Utilizador do sucesso da operação.

Extensões (ou fluxos alternativos)

1a. Não estão instalados pelo menos dois extrusores.

i. O caso de uso termina.

3a. O Utilizador cancela a impressão.

i. O sistema alerta para a impossibilidade de utilização do extrusor caso não termine a calibração e pede confirmação.

ii. O utilizador confirma a intenção de cancelar a impressão.

iii. O caso de uso termina.

iia. O Utilizador desiste de cancelar a impressão.

1. O caso de uso continua no ponto 3.

6a. O Utilizador cancela a calibração.

i. O sistema alerta para a impossibilidade de utilização do extrusor caso não termine a calibração e pede confirmação.

ii. O utilizador confirma a intenção de cancelar a calibração.

iii. O caso de uso termina.

iia. O Utilizador desiste de cancelar a calibração.

1. O use case continua no ponto 6.

8a. O Utilizador cancela a calibração.

i. O sistema alerta para a impossibilidade de utilização do extrusor caso não termine a calibração e pede confirmação.

ii. O utilizador confirma a intenção de cancelar a calibração.

iii. O caso de uso termina.

iia. O Utilizador desiste de cancelar a calibração.

1. O use case continua no ponto 8.

Requisitos especiais

Tecnologia e Lista de Variações dos Dados

Frequência de Ocorrência

- Sempre depois do Utilizador efetuar um assistente de substituição de extrusor (UC8 - A.3.8).
- Este use case será executado no assistente de configuração inicial da impressora (UC10 - A.3.10) para cada extrusor instalado.
- pouco frequente: quando o Utilizador verificar fracos desempenhos de impressão.

Questões em aberto

A.3.18 Casos de Uso - BEECura

Na Figura A.19, é apresentado o diagrama de casos de uso do *plugin* BEECura.

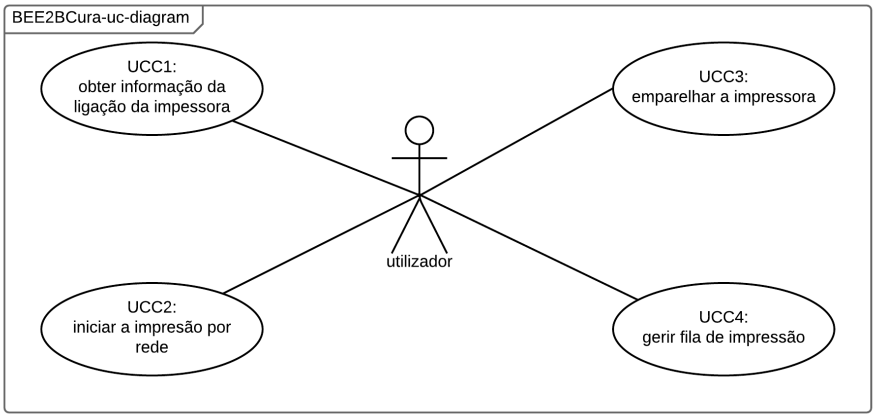


FIGURA A.19: Diagrama de casos de uso.

A.3.18.1 UCC1 - Obter a informação da ligação da impressora

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja saber o estado atual da impressora, para perceber o que está a acontecer a cada momento.

Pré-condições

- O PC está emparelhada com a impressora 3D.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o Sistema.
2. O Sistema inicia processo de atualização da informação da ligação da impressora;.

3. O Sistema apresenta a informação da ligação da impressora ao Utilizador.

Frequência de Ocorrência

- Este caso de uso ocorre constantemente, com um intervalo de um a cinco segundos.

A.3.18.2 UCC2 - Iniciar uma impressão por rede

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja imprimir a partir da rede local.

Pré-condições

- O PC está emparelhada com a impressora 3D.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia a impressão por rede.
2. O Sistema gera um ficheiro *.zip com a informação da cena 3D atual no programa cura.
3. O Sistema envia o pedido de impressão para a impressora.

Frequência de Ocorrência

- Este caso de uso ocorre com muita frequência porque é principal função da impressora 3D.

A.3.18.3 UCC3 - Emparelhar o PC com uma impressora BEE2B

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja imprimir a partir da rede local.

Pré-condições

- o Utilizador iniciou o processo de emparelhamento da impressora na GUI
- o PC e a impressora 3D estão ligados à mesma rede local.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia o processo de emparelhamento da impressora do Sistema;
2. O Sistema verifica que impressoras BEE2B estão disponíveis na rede (à espera de emparelhamento)
3. O Sistema mostra ao utilizador a lista de impressoras BEE2B disponíveis na rede atual;
4. O Utilizador seleciona a impressora BEE2B que deseja emparelhar;
5. O Sistema pede ao utilizador o código de emparelhamento e e-mail para notificações;
6. O Utilizador insere o código de emparelhamento e e-mail;
7. O Sistema efetua o emparelhamento do PC do utilizador com a impressora BEE2B.
8. A GUI BEE2B e o BEECura mostram ao utilizador o sucesso da operação.

Extensões (ou fluxos alternativos)

4a. Não existem impressoras BEE2B disponíveis.

i. O caso de uso termina.

7a. O código de emparelhamento não está correto.

i. O caso de uso continua no ponto 2.

Frequência de Ocorrência

- Este caso de uso ocorre com pouca frequência porque para cada impressora 3D apenas é necessário emparelhar uma vez o PC.

A.3.18.4 UCC4 - Gerir fila de impressão

Ator principal: Utilizador

Partes interessadas e seus interesses

Utilizador: Deseja gerir a fila de impressão da impressora BEE2B para remover ou reordenar algum trabalho de impressão.

Pré-condições

- O PC está emparelhada com a impressora 3D.
- A impressora 3D está em processo de impressão.

Cenário de sucesso principal (ou fluxo básico)

1. O Utilizador inicia processo de gestão de fila de impressão;
2. O Sistema apresenta ao utilizador a lista de cenas 3D presentes na fila de impressão, juntamente com a impressão atual;
3. O Utilizador escolhe a cena 3D que deseja remover ou ordenar da fila de impressão;
4. O Sistema apresenta informação detalhada da cena 3D selecionada e pede confirmação ao utilizador;
5. O Utilizador confirma a intenção de remover (ou ordenar) da fila de impressão a cena 3D selecionada;
6. O Sistema efetua o pedido de gestão de fila de impressão ao servidor de impressão ;
7. O Servidor processa o pedido e responde ao Sistema com o sucesso da operação;
8. O Sistema apresenta ao Utilizador o sucesso da operação.

Extensões (ou fluxos alternativos)

5a. O Utilizador não confirma a intenção de remover a cena 3D da fila de impressão.

i. O caso de uso continua no ponto 4.

Frequência de Ocorrência

- Este caso de uso ocorre com alguma frequência.

A.4 Especificação suplementar

Esta secção é um repositório de todos os requisitos não capturados nos casos de uso.

A.4.1 Funcionalidades

Auditoria: As ações do utilizador devem ser registadas para futura análise.

Local/Linguagem: Deve ser implementado nas línguas português e inglês.

Log de dados e tratamento de erros: registar todos os erros em armazenamento persistente.

A.4.2 Usabilidade

- A interação entre os utilizadores e o sistema deve ser simples, intuitiva e completamente adaptada à ação em causa.
- A utilização de cores não deve limitar a utilização da aplicação por pessoas daltónicas.
- O software desenvolvido deve reduzir o tempo de inatividade (*downtime*) da impressora 3D, incluindo assistentes de funcionalidades que reduzem o tempo e o material gasto em ações e falhas de manutenção.
- O software desenvolvido deve aumentar o tempo de atividade (*uptime*) da impressora 3D, simplificando os procedimentos e auxiliando o utilizador na manutenção da impressora.

A.4.3 Desempenho

Nenhum dos elementos gráficos deve reduzir a frame rate abaixo dos 20 FPS.

A.4.4 Suportabilidade

Testabilidade: Todos os componentes desenvolvidos devem ser testados com testes unitários.

Manutenção:

- Usar boas práticas de análise e design orientado a objetos.
- As atualizações do sistema devem ser efetuadas preferencialmente através de uma ligação à Internet, mas deve ser possível também efetuar esta atualização com uma drive USB.
- As novas funcionalidades devem estar disponíveis através da atualização do sistema.

Adaptabilidade: A GUI BEE2B deve trabalhar com ou sem ligação à rede.

A.4.5 Restrições de design

- Adoção do processo de desenvolvimento de software iterativo e incremental.
- Adoção de boas práticas de design, nomeadamente padrões *General Responsibility Assignment Software Patterns* (GRASP), *Single responsability, Open-closed, Liskov substitution, Interface segregation and Dependency inversion* (SOLID) e MVC.

A.4.6 Restrições de implementação

- O núcleo principal do software deve ser desenvolvido em python.
- Adoção de normas de codificação, (ex. PEP8).
- Adoção de normas de controlo de versões, (ex. GIT).
- Todo o código deve ser devidamente documentado com pydoc.
- A GUI BEE2B irá correr sobre um sistema operativo Linux.
- O Cura BEE2B poderá correr em Windows, Linux ou macOS.

A.4.7 Componentes *Open Source*

Em Geral, será maximizado o uso da tecnologia Open Source python neste projeto. Apesar de ainda ser prematuro para desenhar e escolher os componentes, as seguintes bibliotecas são grandes candidatas:

- PyQt5 para interface gráfica;
- pi3d para renderização de cena 3D;
- pyTest para testes unitários;
- coverage para cobertura do código;
- pytest-cov para cobertura dos testes unitários;
- pyTest-qt para testes unitários da interface gráfica;
- Django para o servidor de impressão;
- DjangoREST para a API do servidor de impressão;

A.4.8 Restrições de interface

A aplicação deve ter uma interface gráfica implementada em pyQt5, e assegurada a separação clara entre as responsabilidades da a GUI e das classes de negócio.

A.4.9 Restrições físicas

- Ecrã Tátil de 7"com resolução de 800X480px.
- Raspberry Pi 3B+ com distribuição Linux ligado ao ecrã tátil.

A.4.10 Interfaces de Software

- **BEE2B PrintServer** - A aplicação BEE2B GUI deve comunicar com o Servidor de Impressão atravez da sua REST API.
- **BEEStats** - O Servidor de Impressão deve comunicar com o Servidor de Telemetria BEEStats através da sua REST API.

A.5 Gestão de Risco

TABELA A.3: Tabela de Risco.

Descrição do Risco	**Probabilidade de Ocorrência**
Fraca performance de Raspberry Pi para fazer render da cena 3D.	Baixa
Cena 3D demasiado grande/detalhada para ser renderizada na Raspberry pi.	Média

TABELA A.4: Tabela de Risco.

Descrição do Risco	**Plano de mitigação**
Fraca performance de Raspberry Pi para fazer render da cena 3D.	Substituir a renderização da cena 3D por cena 2D interativa.
Cena 3D demasiado grande/detalhada para ser renderizada na Raspberry pi.	Converter modelos da cena 3D para modelos de poucos polígonos.

Apêndice B

REST API - *PrintServer* BEE2B

Este apêndice contém a documentação da REST API do PrintServer BEE2B gerada automaticamente pelo servidor de entrega contínua. Na secção 7.2 podemos obter mais informação sobre este servidor.

BEE2B - Print Server API

Internal and External API from BEE2B Print Server

Print Server Variables

List of Printer States:

```
CONNECTING
DISCONNECTED
ERROR
CONNECTED
HEATING
TRANSFERRING
PRINTING
SHUTDOWN
RESUMING
PAUSED
```

Tools ID - Hexadecimal numeration:

```
id00 - id0F : Extruders
id10 - id1F : Bed
id20 - id2F : Others
```

External API

External_API - cancel_print

With no arguments, cancels an ongoing print job.
Cancel print job with print_job_id
Return command result description and True if print job was cancelled successfully or False if not.

GET

```
/cancel_print/{print_job_id}
```

Parameter

Field		Type	Description
print_job_id	optional	string	Optional Id of the print job.

- Request: [#parameter-examples-External_API-cancel_print-0_1_0-0]

```
{
    "printJobId": 2
}
```

- Success-Response: [#success-examples-External_API-cancel_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": True,
    "description": "the print job 3 was cancelled".
```

```
}
```

Error 20X

Name	Description
`PrintJobIdNotFound`	The print job was not found
`NotPrinting`	The printer was not corrently printing

- Error-Response(PrintJobIdNotFound): [#error-examples-External_API-cancel_print-0_1_0-0]
- Error-Response(NotPrinting): [#error-examples-External_API-cancel_print-0_1_0-1]

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "print job not found"
}
```

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "not currently printing"
}
```

External_API - get_filaments

With no arguments, returns Information from available filaments in spools.
Return available information from filament in spool with id toolId.

GET

```
/get_filaments
```

Parameter

Field	Type	Description
`toolId` `optional`	string	Optional Id of the tool.

- Request: [#parameter-examples-External_API-get_filaments-0_1_0-0]

```
{
"toolId": id02
}
```

- Success-Response: [#success-examples-External_API-get_filaments-0_1_0-0]

```
HTTP/1.1 200 OK
{
    id00: {
        "name": "Filament A",
        "quantity": 210.4,
        "material": PLA
    },
    id01: {
        "name": "Filament B",
        "quantity": 310.3,
        "material": PLA
    },
```

```
    id02: {
        "name": "Filament C",
        "quantity": 10.1,
        "material": PETG
    },
    id03: {
        "name": "Filament D",
        "quantity": 50.9,
        "material": PLA
    }
}
```

External_API - get_file_list

Returns the list and their information of files stored in the printer's internal SD card.

GET

```
/get_file_list
```

- Success-Response: [#success-examples-External_API-get_file_list-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "file01": {
        "name": "File A",
        "path": "home/BEE2B-rpi/BEE2B/Files",
        "filament": PLA,
        "filamentQuantity": 32.5,
        "printingTime": 135.6
    },
    "file02": {
        "name": "File B",
        "path": "home/BEE2B-rpi/BEE2B/Files",
        "filament": PLA,
        "filamentQuantity": 34.1,
        "printingTime": 112.3
    },
    "file03": {
        "name": "File C",
        "path": "home/BEE2B-rpi/BEE2B/Files",
        "filament": PETG,
        "filamentQuantity": 13.8,
        "printingTime": 85.2
    },
    "file04": {
        "name": "File D",
        "path": "home/BEE2B-rpi/BEE2B/Files",
        "filament": PLA,
        "filamentQuantity": 63.5,
        "printingTime": 201.3
    }
}
```

External_API - get_state

Returns the current printer state.

GET

```
/get_state
```

- Success-Response: [#success-examples-External_API-get_state-0_1_0-0]

```
HTTP/1.1 200 OK
```

```
{
    "state": "printer.printing"
}
```

External_API - get_stream

Get url for streaming printer camera.

GET

```
/get_stream
```

- Success-Response: [#success-examples-External_API-Get_stream-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "streamUrl": "//rpi-BEE2B/home/BEE2B/camera.py"
}
```

Error 4xx

Name	Description
StreamNotFound	The stream was not found

- Error-Response(StreamNotFound): [#error-examples-External_API-Get_stream-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": StreamNotFound
}
```

External_API - get_temperature

Gets the current temperature and target temperature from all the tools.

GET

```
/get_temperature
```

- Success-Response: [#success-examples-External_API-get_temperature-0_1_0-0]

```
HTTP/1.1 200 OK
{
    id00 : {
        "temperature": 34.5,
        "target": 65
    },
    id01 : {
        "temperature": 24.5,
        "target": 65
    },
    id02 : {
        "temperature": 32.4,
        "target": 65
    },
    id03 : {
        "temperature": 44.5,
        "target": 70
    },
    id10 : {
```

```
        "temperature": 33.9,
        "target": 40
    }
}
```

External_API - get_tools

With no Parameters, returns the name, working hours and nozzle diameter of the extruders.
With toolId, return the name, working hours and nozzle diameter of that tool.

GET

```
/get_tools
```

Parameter

Field		Type	Description
tool_id	optional	string	Optional Id of the tool.

- Request: [#parameter-examples-External_API-get_tools-0_1_0-0]

```
{
"toolId": id02
}
```

- Success-Response: [#success-examples-External_API-get_tools-0_1_0-0]

```
HTTP/1.1 200 OK
{
    id00: {
        "name": "Extruder A",
        "size": 0.2,
        "working_hours": 221.4
    },
    id01: {
        "name": "Extruder B",
        "size": 0.2,
        "working_hours": 221.4
    },
    id02: {
        "name": "Extruder C",
        "size": 0.4,
        "working_hours": 134.6
    },
    id03: {
        "name": "Extruder D",
        "size": 0.2,
        "working_hours": 221.4
    }
}
```

External_API - pause_print

Pause an ongoing print job.
Return command result description and True if print job was paused successfully or False if not.

GET

```
/pause_print}
```

- Success-Response: [#success-examples-External_API-pause_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": True,
    "description": "the print job 3 was paused".
}
```

Error 20X

Name	Description
`NotPrinting`	The printer was not corrently printing

- Error-Response(NotPrinting): [#error-examples-External_API-pause_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "not currently printing"
}
```

External_API - request_authorization

Return the token key for current computer.

POST

```
/request_authorization
```

Parameter

Field	Type	Description
`computer_id`	string	Mandatory Unique id that identify current computer.

- Request: [#parameter-examples-External_API-request_authorization-0_1_0-0]

```
{
    "client_id": CGTR-ASDF7-CSA5-23KD
}
```

- Success-Response: [#success-examples-External_API-request_authorization-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "token": 812DSAD657DF
}
```

External_API - resume_print

Resume a paused print job.
Return command result description and True if print job was resumed successfully or False if not.

GET

```
/resume_print}
```

- Success-Response: [#success-examples-External_API-resume_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": True,
    "description": "the print job 3 was resumed".
}
```

Error 20X

Name	Description
`NotPrinting`	The printer was not corrently printing

- Error-Response(NotPrinting): [#error-examples-External_API-resume_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "not currently printing"
}
```

External_API - set_temperature

Starts heating the printer tools. Returns the current temperature and target temperature from all the tools.

POST

```
/set_temperature
```

Parameter

Field	Type	Description
`tool`	string	Mandatory ID of the tool, one entry for each tool.
`target`	float	Mandatory Temperature target to the tool, one entry for each tool.

- Request: [#parameter-examples-External_API-set_temperature-0_1_0-0]

```
[
    {
        "tool": "id02",
        "target": 65
    },
    {
        "tool": "id03",
        "target": 65
    },
    {
        "tool": "id10",
        "target": 40
    }
]
```

- Success-Response: [#success-examples-External_API-set_temperature-0_1_0-0]

```
HTTP/1.1 200 OK
{
    id00 : {
        "temperature": 34.5,
        "target": 65
```

```
    },
    id01 : {
        "temperature": 24.5,
        "target": 65
    },
    id10 : {
        "temperature": 33.9,
        "target": 40
    }
}
```

Error 4xx

Name	Description
`ToolNotFound`	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-External_API-set_temperature-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

External_API - start_print

This method is used to submit print jobs.
Return the job id and job position if the operation is successful or None if not.

PUT

```
/start_print/{filename}
```

Parameter

Field	Type	Description
`filename`	string	Mandatory Filename of print file.

- Request: [#parameter-examples-External_API-start_print-0_1_0-0]

```
{
    "file_obj": bee.zip
}
```

- Success-Response: [#success-examples-External_API-start_print-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "job_id": 12,
    "job_position": 3
}
```

Error 4xx

Name	Description
`FileNotFound`	The file was not found

- Error-Response(FileNotFound): [#error-examples-External_API-start_print-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": FileNotFound
}
```

Internal API

Internal_API - add_extruder

Add new extruder.
Return True if extruder is added and False if is not.

`GET`

```
/extruder/add
```

Parameter

Field	Type	Description
`toolId`	String	Mandatory Id of the tool.

- Request: [#parameter-examples-Internal_API-add_extruder-0_1_0-0]

```
{
"toolId": id03
}
```

Success 200

Field	Type	Description
`isExtruderAdded`	bool	True if extruder is added and False if is not.

- Success-Response: [#success-examples-Internal_API-add_extruder-0_1_0-0]

```
HTTP/1.1 200 OK
{
"isExtruderAdded": True
}
```

Error 4xx

Name	Description
`ToolNotFound`	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-Internal_API-add_extruder-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

Internal_API - cancel_extruder

Cancel extruder operation.
Return True if extruder operation is canceled and False if is not.

`GET`

```
/extruder/cancel
```

Parameter

Field	Type	Description
`toolId`	String	Mandatory Id of the tool.

- Request: [#parameter-examples-Internal_API-cancel_extruder-0_1_0-0]

```
{
"toolId": id03
}
```

Success 200

Field	Type	Description
`isExtruderAdded`	bool	True if extruder is added and False if is not.

- Success-Response: [#success-examples-Internal_API-cancel_extruder-0_1_0-0]

```
HTTP/1.1 200 OK
{
"isExtruderCanceled": True
}
```

Error 4xx

Name	Description
`ToolNotFound`	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-Internal_API-cancel_extruder-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

Internal_API - connect

This method is used to force a connection with printer, in cases where the initial connection failed or the connection was manually terminated.
Returns the current printer state.

`GET`

```
/connect
```

- Success-Response: [#success-examples-Internal_API-connect-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "state": "printer.connected"
}
```

Internal_API - disconnect

Disconnects the current active connection to the printer.
Returns the current printer state.

GET

```
/disconnect
```

- Success-Response: [#success-examples-Internal_API-disconnect-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "state": "printer.disconnected"
}
```

Internal_API - get_current_job

This method is used to return the progress of current printing

GET

```
/get_current_job
```

- Success-Response: [#success-examples-Internal_API-get_current_job-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": "true",
    "printing_progress": 15,
    "file": beeX2
}
```

Internal_API - get_working_hours

Returns the total printer Working Hours.

GET

```
/get_working_hours
```

- Success-Response: [#success-examples-Internal_API-get_working_hours-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "workingHours": 153.4
}
```

Internal_API - go_to

Go to defined x, y, z position.
Return command result description and True if printer is at requested position and False if is not.

GET

```
/go_to
```

Parameter

Field	Type	Description
x	Float	Mandatory x coordinates.

- Request: [#parameter-examples-Internal_API-go_to-0_1_0-0]

```
{
    "x": 10,
    "y": 20,
    "z": 30
}
```

- Success-Response: [#success-examples-Internal_API-go_to-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": True,
    "description": "printer at requested position"
}
```

Error 20X

Name	Description
`PrinterInProgress`	The printer has an ongoing printing

- Error-Response(PrinterInProgress): [#error-examples-Internal_API-go_to-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "printing in progress"
}
```

Internal_API - home

Sends the command to the printer to move all the axis to their home position.
Return a command result description and True if printer is at home position and False if is not.

GET

```
/home
```

- Success-Response: [#success-examples-Internal_API-home-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": true,
    "description": "printer at home position"
}
```

Error 20X

Name	Description
`PrinterInProgress`	The printer has an ongoing printing

- Error-Response(PrinterInProgress): [#error-examples-Internal_API-home-0_1_0-0]

```
HTTP/1.1 200 OK
{
    "response": False,
    "description": "printing in progress"
}
```

Internal_API - load_filament

Start loading filament.
Return True if filament is complete loaded and False if is not.

PUT

```
/load_filament
```

Parameter

Field	Type	Description
`toolId`	String	Mandatory Id of the tool.

- Request: [#parameter-examples-Internal_API-load_filament-0_1_0-0]

```
{
"toolId": id03
}
```

- Success-Response: [#success-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 200 OK
{
"isFilamentLoaded": True
}
```

Error 4xx

Name	Description
`ToolNotFound`	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

Internal_API - load_filament

Start loading filament.
Return True if filament is complete loaded and False if is not.

PUT

```
/load_filament
```

Parameter

Field	Type	Description
`toolId`	String	Mandatory Id of the tool.

- Request: [#parameter-examples-Internal_API-load_filament-0_1_0-0]

```
{
"toolId": id03
}
```

- Success-Response: [#success-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 200 OK
{
"isFilamentLoaded": True
}
```

Error 4xx

Name	Description
`ToolNotFound`	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

Internal_API - load_filament

Start loading filament.

Return True if filament is complete loaded and False if is not.

PUT

```
/load_filament
```

Parameter

Field	Type	Description
toolId	String	Mandatory Id of the tool.

- Request: [#parameter-examples-Internal_API-load_filament-0_1_0-0]

```
{
"toolId": id03
}
```

- Success-Response: [#success-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 200 OK
{
"isFilamentLoaded": True
}
```

Error 4xx

Name	Description
ToolNotFound	The tool was not found

- Error-Response(ToolNotFound): [#error-examples-Internal_API-load_filament-0_1_0-0]

```
HTTP/1.1 404 Not Found
{
    "error": ToolNotFound
}
```

Internal_API - move

Move to the given coordinates relative to the current point.
Return the command result description and True if printer is at requested position and False if is not.

GET

```
/move
```

Parameter

Field	Type	Description
x	Float	Mandatory x coordinates.
y	Float	Mandatory y coordinates.
z	Float	Mandatory z coordinates.

Apêndice C

Questionário de qualidade

O questionário de qualidade apresentado neste apêndice será efetuado pelos utilizadores durante os testes alpha e beta.

Questionário de Qualidade BEE2B

*Obrigatório

Informação Pessoal

Esta secção é relacionada com as características pessoais que definem o utilizador

1. **Género**
   Marcar apenas uma oval.

   - ◯ Masculino
   - ◯ Feminino
   - ◯ outro
   - ◯ Outra: _____

2. **Idade**

   _____

3. **Tipo de Utilizador** *
   Marcar apenas uma oval.

   - ◯ Utilizador normal
   - ◯ Utilizador Técnico

4. **Qual a sua profissão**

   _____

5. **Têm algum problema de daltonismo** *
   Marcar apenas uma oval.

   - ◯ Sim *Passe para a pergunta 6.*
   - ◯ Não *Passe para a pergunta 8.*

Daltonismo

6. **1.1 - Acha que a Interface gráfica está preparada para pessoas com problemas de daltonismo?**
   Marcar apenas uma oval.

   - ◯ Sim *Após a última pergunta desta secção, passe para a pergunta 8.*
   - ◯ Não

7. **1.2 - Que melhorias sugere para melhorar a interface gráfica para pessoas com daltonismo?**

Interface Gráfico BEE2B

Para todas as questões pode avaliar com uma pontuação de 1 a 5, sendo 1 muito fraco e 5 muito bom.

8. **2.1 - Como avalia a dificuldade da configuração inicial da Interface Gráfica BEE2B ?**
   Marcar apenas uma oval.

	1	2	3	4	5	
muito fácil	◯	◯	◯	◯	◯	muito dificil

9. **2.2 - Quantas vezes usou a interface gráfica?**
   Marcar apenas uma oval.

   - ◯ apenas 1
   - ◯ Entre 2 a 5 vezes
   - ◯ Mais de 5 vezes

10. **2.3 - Selecione as funcionalidades que teve dificuldade em encontrar:**
    Marcar tudo o que for aplicável.

    - ☐ Imprimir a partir de uma drive USB
    - ☐ Imprimir um ficheiro recente
    - ☐ Imprimir a partir de um PC (Cura BEE2B)
    - ☐ Cancelar uma impressão
    - ☐ Pausar e retomar um impressão
    - ☐ Mudar de Filamento
    - ☐ Mudar de Extrusor/Hotend
    - ☐ Remover um objeto da impressão
    - ☐ Configurações de Rede
    - ☐ Restaurar Software
    - ☐ Atualizar Software

11. **2.4 - Considera fácil aceder às principais funcionalidades da impressora?**
Marcar apenas uma oval.

- Sim
- Não

12. **2.5 - Considera fácil iniciar uma impressão?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito dificil						muito fácil

13. **2.6 - Considera a Interface Gráfica intuitiva?**
Marcar apenas uma oval.

	1	2	3	4	5	
pouco intuitiva						muito intuitiva

14. **2.7 - Como avalia a usabilidade da Interface gráfica?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito fraca						muito boa

15. **2.8 - Como classifica a performance da Interface Gráfica?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito fraca						muito boa

16. **2.9 - Como avalia a necessidade de um manual de utilizador para utilizar a interface gráfica?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito necessário						desnecessário

17. **2.10 - Considera os procedimentos de manutenção simples e fáceis de executar?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito complexo						muito simples

18. **2.11 - Como avalia a dificuldade de atualizar o software da Interface Gráfica BEE2B ?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito fácil	◯	◯	◯	◯	◯	muito dificil

19. **2.12 - Quantos erros apareceram durante a utilização da Interface Gráfica? (excluindo os alertas de falta de filamento e outros relacionados com a impressora)**
Marcar apenas uma oval.

- ◯ 0
- ◯ Entre 1 e 5
- ◯ Mais de 5

20. **2.13 - O que melhoraria na próxima versão da Interface Gráfica BEE2B**

Software Cura BEE2B

21. **3.1 - Como avalia a dificuldade de instalação do software Cura BEE2B**
Marcar apenas uma oval.

	1	2	3	4	5	
muito dificil	◯	◯	◯	◯	◯	muito fácil

22. **3.2 - Como avalia a dificuldade de efetuar a ligação à impressora com o software Cura BEE2B e realizar o emparelhamento com a mesma?**
Marcar apenas uma oval.

	1	2	3	4	5	
muito difícil	◯	◯	◯	◯	◯	muito fácil

23. **3.3 - O que melhoraria na próxima versão do Cura BEE2B?**

Apêndice D

Respostas ao questionário de qualidade - Testes Alpha

Este apêndice é um resumo dos resultados dos questionários de qualidade efetuados nos teste alpha.

A média de pontuação das respostas, das seguintes perguntas, são utilizadas como input para a avaliação no âmbito do QEF 8.4.

- **Pergunta 2.5** - Média de pontuação de resposta **4.29**
- **Pergunta 2.6** - Média **4.15**
- **Pergunta 2.7** - Média **4**
- **Pergunta 2.9** - Média **3.58**
- **Pergunta 2.10** - Média **3.57**

D.1 Informação Pessoal

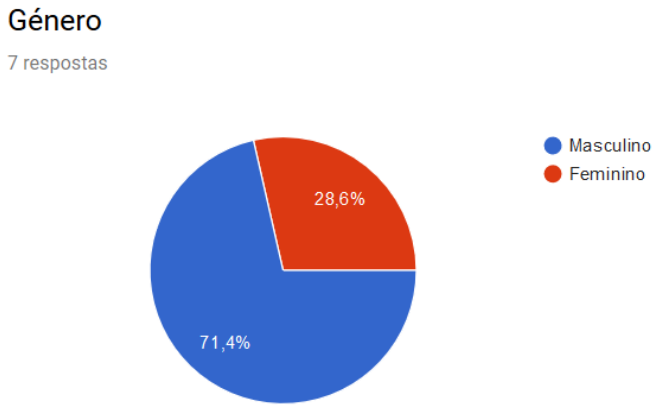


FIGURA D.1: Respostas do questionário nos testes alpha - Género .

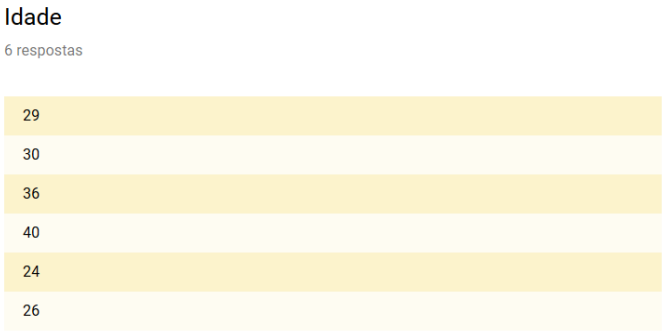


FIGURA D.2: Respostas do questionário nos testes alpha - Idade .

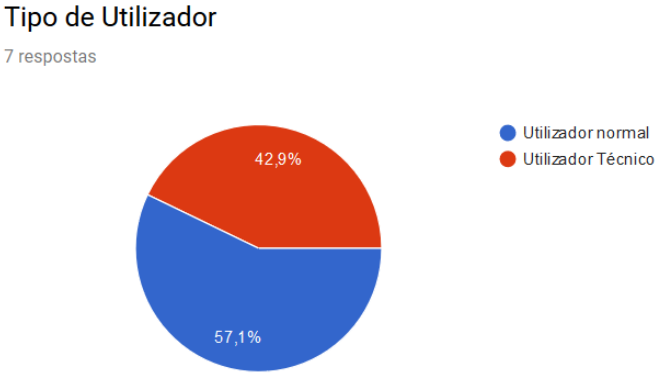


FIGURA D.3: Respostas do questionário nos testes alpha - Tipo de Utilizador.

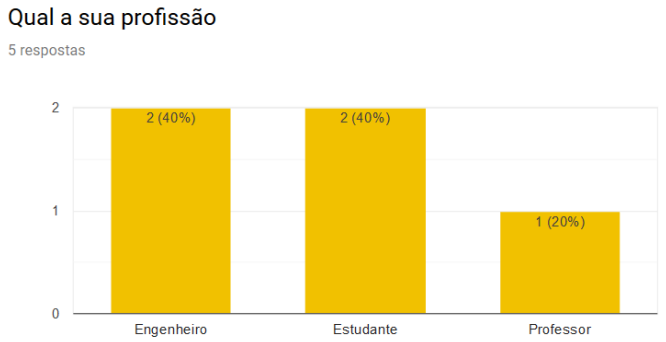


FIGURA D.4: Respostas do questionário nos testes alpha - Profissao .

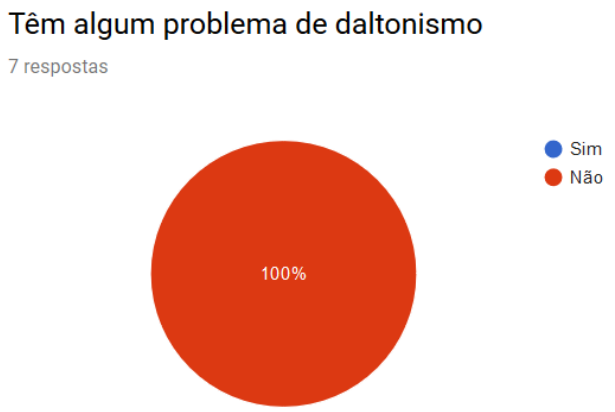


FIGURA D.5: Respostas do questionário nos testes alpha - Daltonismo .

D.2 Daltonismo

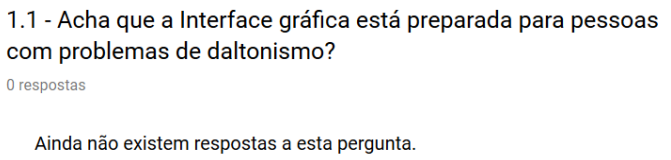


FIGURA D.6: Respostas do questionário nos testes alpha - daltonismo 1.

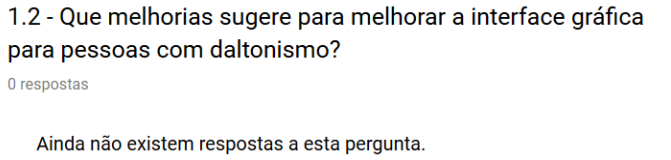


FIGURA D.7: Respostas do questionário nos testes alpha - Daltonismo 2.

D.3 Interface Gráfico BEE2B

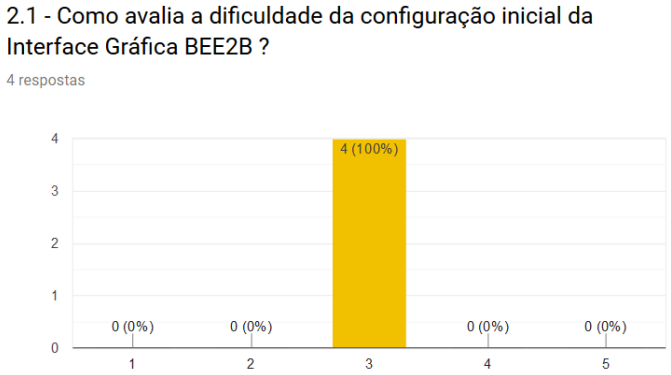


FIGURA D.8: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 1.

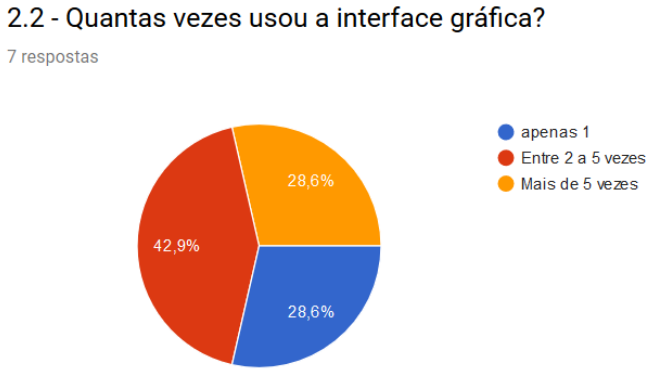


FIGURA D.9: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 2.

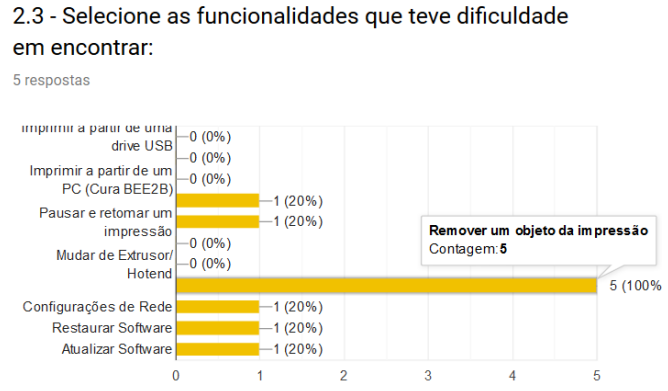


FIGURA D.10: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 3.

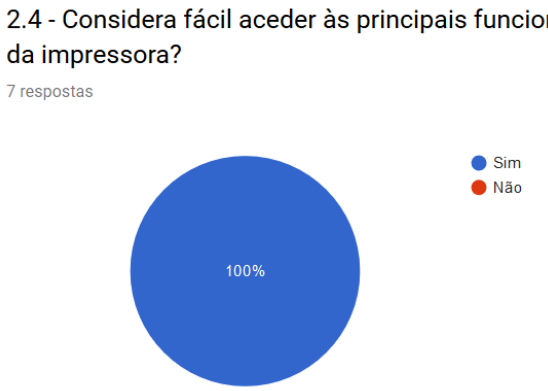


FIGURA D.11: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 4.

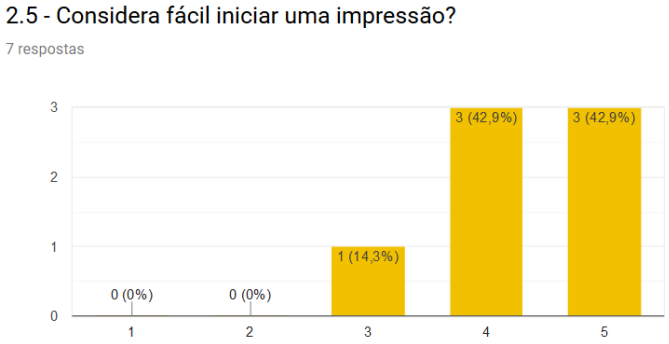


FIGURA D.12: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 5.

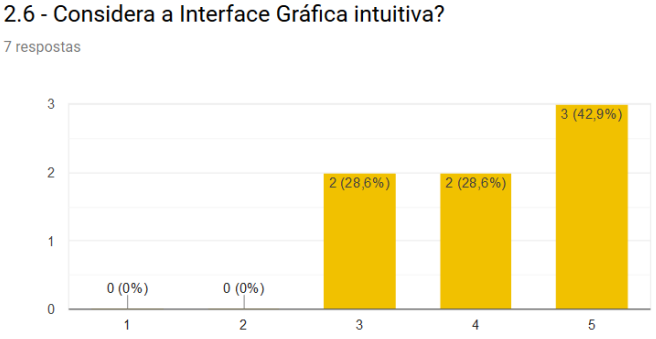


FIGURA D.13: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 6.

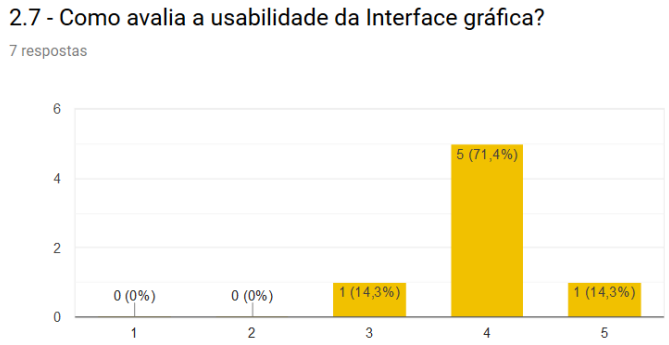


FIGURA D.14: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 7.

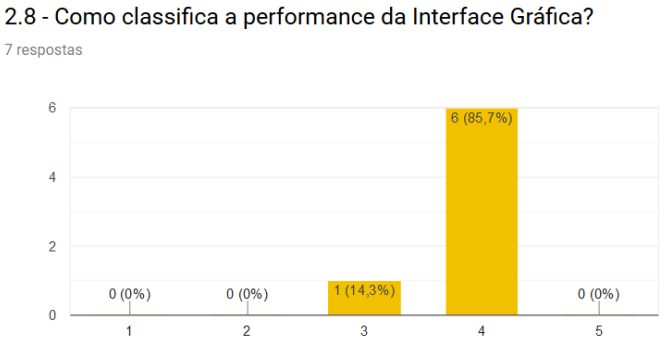


FIGURA D.15: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 8.

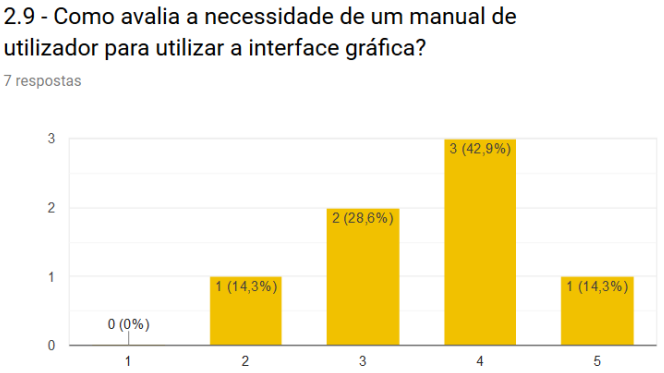


FIGURA D.16: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 9.

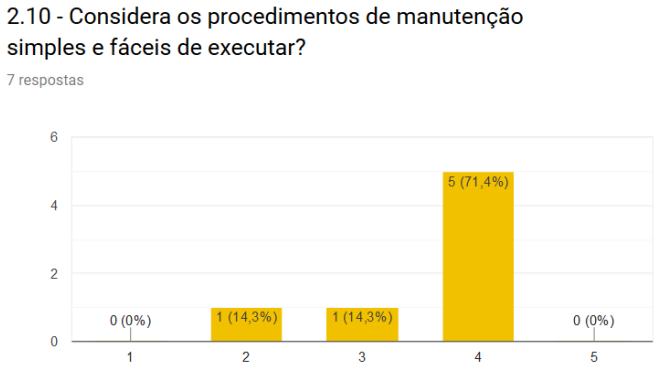


FIGURA D.17: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 10.

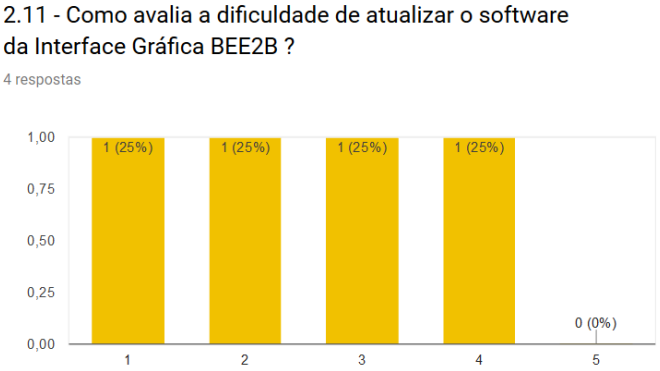


FIGURA D.18: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 11.

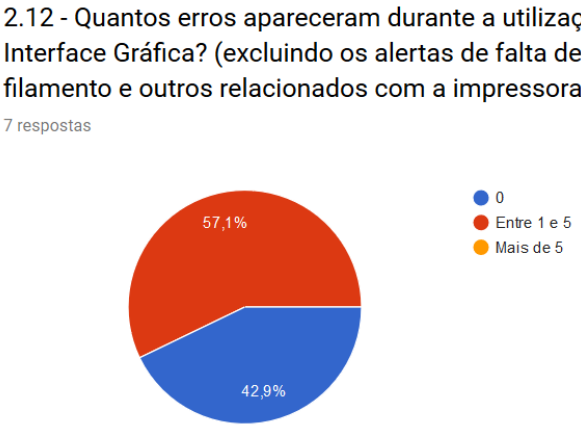


FIGURA D.19: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 12.

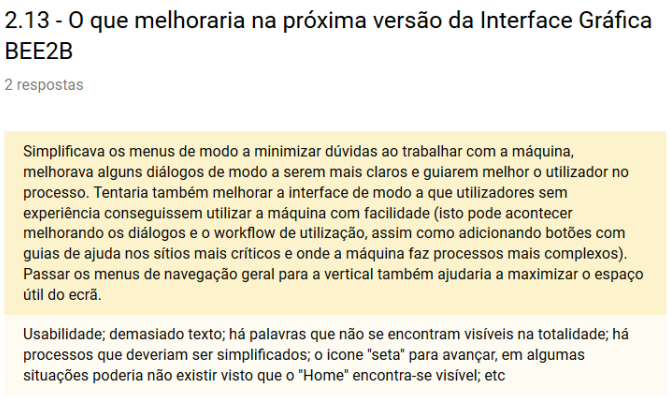


FIGURA D.20: Respostas do questionário nos testes alpha - Interface Gráfico BEE2B 13.

D.4 Software Cura BEE2B (BEECura)

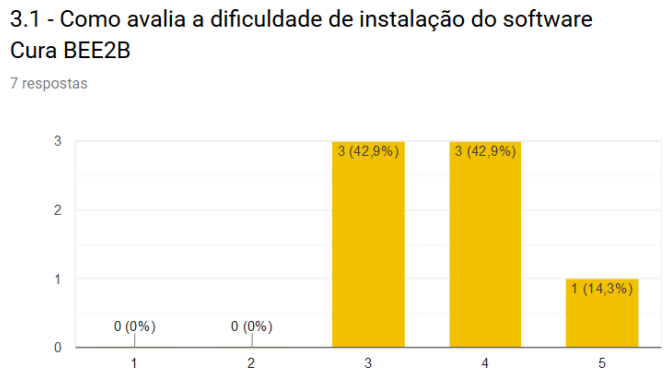


FIGURA D.21: Respostas do questionário nos testes alpha - BEECura 1.

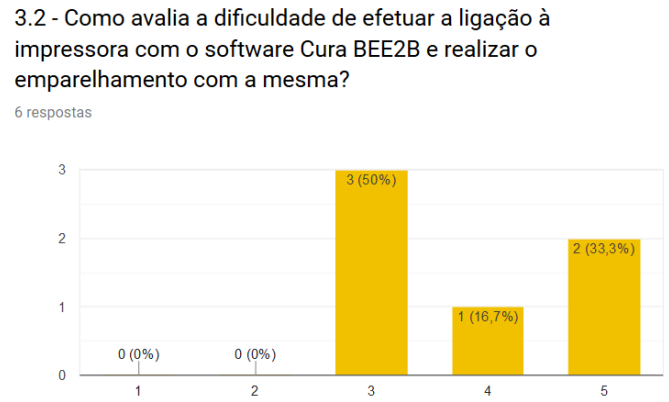


FIGURA D.22: Respostas do questionário nos testes alpha - BEECura 2.

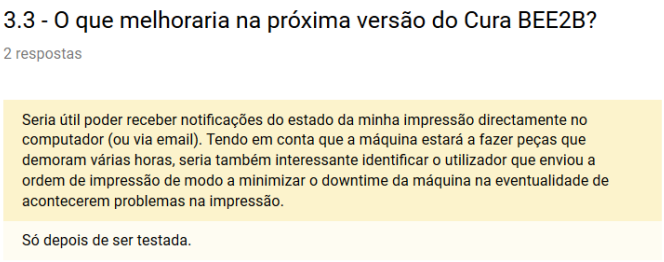


FIGURA D.23: Respostas do questionário nos testes alpha - BEECura 3.

Apêndice E

QEF - Testes *Alpha*

Este apêndice mostra o preenchimento do QEF após a realização dos testes *alpha*.

De notar que, mesmo a seis meses da conclusão do projeto, já se obtém uma avaliação de 73%.

Para obter mais informação relativamente às métricas de avaliação dos diversos requisitos ver a secção 8.4.2.

q	D	Qi	Dimension	Qj	Wij (Factor Weight j in Dim i) [0,1]	Factor	rwjk (requirement weight k in Factor j) {2, 4, 6, 8, 10}	Requirement	wfk % requirement fulfillment k) [0,100]
		42,647	Funcionalidade	46,875	0,471	Funcional (Use Cases)	10	FF01 - Iniciar impressão a partir do PC (Cura BEE2B)	100
							10	FF02 - Iniciar impressão a partir de uma drive USB	100
							10	FF03 - Iniciar impressão de um ficheiro recente	100
							10	FF04 - Imprimir de uma cena 3D (com renderização de cena 3D/2D)	100
							10	FF06 - Remover objeto da impressão	50
							10	FF07 - Substituir Filamento	100
							10	FF08 - Substituir Extrusor/Hotend	100
							10	FF09 - Validar de periféricos	0
							10	FF10 - Configuração Inicial da Impressora	0
							10	FF11 - Modificar configurações de rede	0
							10	FF12 - Restaurar software de fábrica	0
							10	FF13 - Atualizar Software	0
							10	FF14 - Emparelhar PC com impressora	0
							10	FF16 - Mudar idioma	100
							10	FF17 - Calibrar offset Z	0
							10	FF18 - Calibrar offset XY	0
				25	0,353	PrintServer API	10	FP01 - Gerir impressora (connect, disconnect e get_state, request_authorization, get_printer_info)	75
							10	FP02 - Gerir Movimento (go_to, home e move)	50
							10	FP03 - Gerir Filamento (get_filaments, filament_identify, add_filament, load e unload)	25
							10	FP04 - Gerir Ferramentas (set_tool, get_tools, periph_validation, get_stream, get_temperature, set_temperature)	50
							10	FP05 - Gerir Ficheiros (remove_file, get_file_list, get_file_settings, set_file_settings)	25
							10	FP06 - Gerir Extrusor (prepare_extruder, remove_extruder, new_extruder, add_extruder, cancel_extruder)	0

						PrintServer API	10	FP07 - Gerir impressão(cancel_print, start_print, pause, resume, print_recent_job, remove_object)	25
							10	FP08 - Gestão de Fila de Impressão (remove_from_queue, get_current_job, sort_queue, get_queue)	0
							10	FP09 - Gerir Rede(start_network_config, cancel_network_config, get_networks, connect_network)	0
							10	FP10 - Gerir Emparelhamento(start_pairing, pair, cancel_paring, pairing_list, unpair)	0
							10	FP11 - Gerir Calibrações(start_calibrate_z, add_calibrate_z, stop_calibrate_z, start_calibrate_xy, add_calibrate_xy, stop_calibrate_xy)	0
							10	FP12 - Impressora Virtual	50
				66,6667	0,176	Cura Plugins	10	FC01 - Skin Cura BEEVERYCREATIVE	100
							10	FC02 - Plugin: Impressão por rede	100
							10	FC03 - Plugin: Fila de Impressão	50
							10	FC04 - Plugin: Infomarção da ligação da impressora	100
							10	FC05 - Plugin: Emparelhamento da impressora	50
							10	FC06 - Plugin: Perfis de Impressão	0
73%	0,70	102,14	Adaptabilidade	83,3333	0,14	Versatibilidade	10	AV01 - linguagem do GUI de acordo com a linguagem selecionada (Inglês ou Português)	100
							10	AV02 - A GUI BEE2B trabalha normalmente sem ligação à rede	100
							10	AV03- Cura BEE2B executa sem problemas em todas as plataformas (Windows, Linux e macOS)	50
				92,5	0,59	Usabilidade	10	AU01 - O utilizador consegue facilmente começar a imprimir	100
							10	AU02 - O utilizador consegue utilizar a GUI sem ler o manual	75
							10	AU03 - Símbolos e metáforas da GUI são apropriados e consistêntes	100
							10	AU04 - A GUI é intuitiva	100
							10	AU05 - O utilizador recebe feedback imediato pelas suas acções	100
							10	AU06 - Tarefas semelhantes tem controlos semelhantes	100
							10	AU07 - O Design da GUI é homogéneo	100
							10	AU08 - A GUI tem uma boa usabilidade	75
							10	AU09 - Assistentes para tarefas de manutenção (reduzir downtime)	100

						10	AU10 -tarefas de manutenção com procedimentos simples e com ajuda presente par auxiliar o utilizador (aumentar uptime)	75
				88,8889	0,41	Manutenção	10	AM1 - São aplicados padrões de design sempre que é apropriado	100
							10	AM02 - A aplicação não guarda dados sensíveis do utilizador	100
							10	AM03- As ações do utilizador são registadas para futura análise	100
							10	AM04 - Todos os erros são registados em armazenamento permanente	100
							10	AM05 - Código devidamente documentado com pydoc	0
							10	AM06 - código segundo normas de codificação pep8	100
							10	AM07 - Cobertura de testes unitários superior a 70%	100
							10	AM08 - Clara separação entre responsabilidades da GUI e das classes de negócio	100
							10	AM09 - Existe um glossário com a terminologia e definições de conceito.	100
		59,704	Eficiência	100	0,25	Navegação	10	EN01 - GUI com uma boa estrutura e permite aos utilizadores aceder às principais funcionalidades de uma forma intuitiva	100
							10	EN02 - GUI rápido e simples	100
							10	EN03 - a GUI corre sem erros e os erros inesperados são tratados	100
							10	EN04 - o sistema de navegação da GUI mete o utilizador no controlo	100
				100	0,32	Qualidade do Conteúdo	10	EQ01 - Texto bem escrito e com todas as frases a fazer sentido	100
							10	EQ02 - Todas as mensagens são fáceis de entender e são apropriadas para humanos	100
							10	EQ03 - O conteúdo da GUI é claro para daltóticos	100
							10	EQ04 - Nenhuma mensagem deve conter conteúdo racista, xenófabo ou anti-semita	100
							10	EQ05 - Frame Rate da cena 3D da GUI sempre superior a 20fps	100
							10	EQ06 - Todos os conteúdos são relacionados com a impressora 3D BEE2B	100
				8,33333	0,38	Suporte	10	ES01 - Configureação inicial é fácil para o público alvo	0
							10	ES02 - A GUI é estável e corre sistematicamente sem falhas	50
							10	ES03 - Estratégia de backup da base de dados do PrintServer.	0
							10	ES04 - Procedimento de recuperação de falha da base de dados do PrintServer	0

10	ES05 - O Software é facilmente atualizado.	0
10	ES06 - Manual do utilizador em ambas as línguas (PT e ENG)	0

Assumptions, 2 User Types:	**Administrator** - User who can administrate the whole application and has access to statistical graphs
	Guest - User that uses the application. Only unable to access statistical data.