INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Monitorização remota de óleos usados baseada em sensores

NUNO JORGE ANTUNES DE LIMA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Smart Depot
Monitorização remota de óleos usados baseada em sensores

Nuno Jorge Antunes de Lima

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: António Manuel Cardoso da Costa
Co-Orientador: Nuno Alexandre Magalhães Pereira

Júri:
Presidente:

Vogais:

Porto, 22 de Outubro de 2017

Dedicatória

Dedico este trabalho aos meus filhos, Sofia e Miguel, e à minha esposa que sempre me apoiaram, incentivaram e deram a força adicional necessária para que fosse possível o meu regresso, passado uma década, à vida académica novamente como trabalhador-estudante.

Foram sempre vós a minha maior força.

Resumo

A génese deste projeto partiu da intenção de melhorar o processo de recolha dos óleos usados. Surgiu da observação de um problema recorrente das organizações que têm depósitos de óleo para os quais não encontram mais utilidade após a finalidade para a qual o adquiriram. Frequentemente, estes depósitos não são corretamente monitorizados, o que leva a se passem vários dias desde que ficam completamente cheios até à sua recolha.

O projeto Smart Depot pretende trazer eficiência ao processo, oferecendo às empresas de recolha uma solução inteligente de monitorização, na qual se podem basear para consultar a informação do estado dos depósitos de óleo usado existentes nos seus pontos de recolha.

Palavras-chave: Monitorização, Óleo, Reciclagem, Recolha, Resíduos, Sensores

Abstract

The genesis of this project was based on the intention to improve the process of collection of waste oils. It arose from the observation of a recurring problem of organizations which have oil deposits that are no longer useful after the purpose for which they have acquired it. Often, these deposits are not properly monitored, which will take several days from being completely filled until collected.

The Smart Depot project aims to bring efficiency to the process by providing collection companies with an intelligent monitoring solution that can be used to consult information on the state of used oil deposits at their collection points.

Agradecimentos

Aos meus colegas de curso e amigos (da B308), Ana Almeida, Bruno Gomes, Ricardo Costa e Viviana Batista, o meu sincero obrigado pelo companheirismo, motivação e amizade.

Ao amigo Pedro Strecht agradeço toda a paciência e atenção prestada às minhas longas palestras sobre quais as melhores abordagens técnicas aos problemas tratados neste trabalho.

Agradeço a dedicação de todos os docentes do curso de mestrado que me permitiram adquirir novos conhecimentos nesta área que tanto aprecio.

Dirijo especial agradecimento aos Professores António Costa e Nuno Pereira, que me concederam o privilégio de os ter como orientadores deste projeto.

"So the problem is not so much to see what nobody has yet seen, as to think what nobody has yet thought concerning that which everybody sees."

— Arthur Schopenhauer

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ACS	*Ant Colony System*.
ADI	Agência de Inovação.
AHP	*Analitic Hierarchy Process*.
AMA	Aplicação Móvel Android.
ANOVA	Análise de Variância.
AP	*Access Point*.
API	*Application Programming Interface*.
BA	*Basic Authentication*.
BLE	*Bluetooth Low-Energy*.
BPMN	*Business Process Model and Notation*.
CA	*Certificate Authority*.
CAS	*Central Authentication Service*.
CES	*Consumer Eletronics Show*.
CPU	*Central Processing Unit*.
ER	Entidade Relacionamento.
ESB	*Enterprise Service Bus*.
FEE	*Fuzzy Front End*.
FOSS	*Free and Open-Source Software*.
GPRS	*General Packet Radio Service*.
GPS	*Global Positioning System*.
GSM	*Global System for Mobile Communications*.
HORECA	Sector da Hotelaria e Restauração.
HTTP	*Hyper Text Transfer Protocol*.
HTTPS	*Hyper Text Transfer Protocol Secure*.
IC	Circuito Integrado.
ID	*Unique Identifier*.
IoT	*Internet of Things*.
JWT	*JSON Web Token*.
LGPL	*GNU Lesser General Public License*.
M2M	*Machine-to-Machine*.
MCU	Microcontrolador.

NCD	*New Concept Development*.
NPD	*New Product Development*.
OAU	Óleos Alimentares Usados.
OO	*Object-Oriented*.
ORM	*Object-Relational Mapping*.
OU	Óleos Lubrificantes Usados.
PKA	*Public Key Authentication*.
QREN	Quadro de Referência Estratégica Nacional.
REST	*Representational State Transfer*.
SGBD	Sistema de Gestão de Base de Dados.
SGC	Sistema de Gestão Centralizado.
SMR	Sistema de Monitorização Remota.
SO	Sistema Operativo.
SOGILUB	Sociedade de Gestão Integrada de Óleos Lubrificantes Usados, Lda..
SSH	*Secure Shell*.
SSL	*Secure Sockets Layer*.
TCP/IP	*Transmission Control Protocol/Internet Protocol*.
TSP	*Travelling Salesman Problem*.
UML	*Unified Modeling Language*.
URL	*Uniform Resource Locator*.
UTAD	Universidade de Trás-os-Montes e Alto Douro.
VRP	*Vehicle Routing Problem*.

Capítulo 1

Introdução

Neste primeiro capítulo será apresentado o problema tratado pelo projeto Smart Depot, o contexto onde este se insere, como se pretende abordar o problema, quais os objetivos que se pretende atingir e qual o valor acrescentado pelo projeto.

1.1 Contexto

A preservação do meio ambiente é um tema cada vez mais abordado nos dias que correm, sendo alvo de recorrentes estudos que tentam interpretar os reflexos da interação humana. Existem preocupações profundas com a poluição e alterações climáticas, sendo estas responsáveis pela morte de plantas, animais e humanos todos os dias.

O óleo usado é uma das fontes de poluição mais preocupantes pelo impacto que tem para o ambiente. A reciclagem deste resíduo é de extrema importância evitando a poluição das águas e do solo.

As necessidades de reciclagem prendem-se com as seguintes preocupações:

- Preservação do meio ambiente;
- Poluição e alterações climáticas;
- Respeitar legislações atuais;
- Gerir corretamente os recursos poluentes.

A Figura 1.1 pretende representar, de uma forma não técnica, o ciclo percorrido pelo óleo desde a produção até à sua valorização e re-introdução no mercado. Nesta figura

Os intervenientes deste mercado, identificados na Figura 1.1, são:

Produtores de óleos novos (1) Existem dois tipos de empresas produtoras de óleos novos, os alimentares e os lubrificantes. As empresas de óleos lubrificantes têm como principal cliente a indústria automóvel e estão obrigados por lei a contribuir financeiramente para o processo de recolha e tratamento depois dos óleos que vendem serem usados;

Consumidores de óleos alimentares e lubrificantes (2) Os grandes grupos de consumidores de óleos alimentares são a indústria alimentar, grupo Sector da Hotelaria e Restauração (HORECA) e os consumidores finais. Já os produtores de resíduos lubrificantes (Óleos Lubrificantes Usados (OU)) são essencialmente a indústria dos transportes e oficinas;

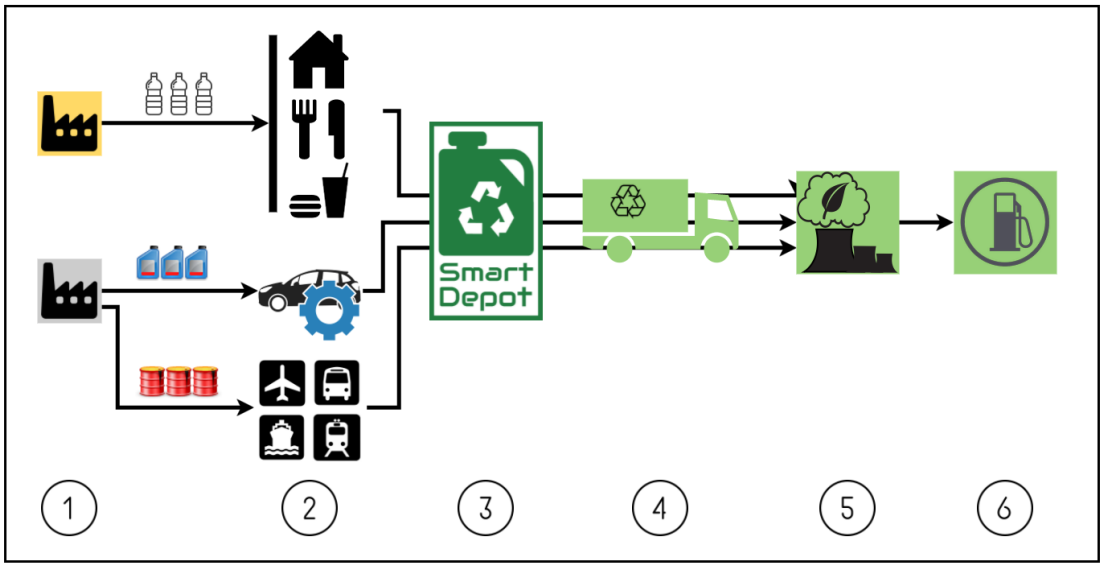


Figura 1.1: Ciclo do Óleo

Smart Depot (3) Este projeto pretende posicionar-se entre os consumidores/produtores dos resíduos e as empresas de recolha, monitorando de forma automática os depósitos de óleo usados e disponibilizando essa informação às empresas de recolha;

Empresas de recolha de Óleos Alimentares Usados (OAU) e de OU (4) As empresas de recolha de óleos estão segmentadas em dois grupos por restrições legais. A gestão da recolha dos OU é feita pela Sociedade de Gestão Integrada de Óleos Lubrificantes Usados, Lda. (SOGILUB), que representa a e gere esta rede de recolha em Portugal perante os produtores de óleos lubrificantes. As empresas de recolha de OAU são privadas, não tendo qualquer organização centralizadora que as represente. Estas empresas são normalmente contratadas pelos municípios para colmatar a área de recolha de resíduos destas, já que os municípios estão obrigados por lei a prestar este serviço;

Refinarias (5) Estas empresas compram os óleos usados recolhidos e fazem o seu tratamento e transformação em outros tipos de óleo lubrificantes, aditivos ou em biodiesel (6).

A valorização dos óleos usados encontra-se representada no ponto (6) da Figura 1.1 pela produção de biodiesel, por este ser a valorização preferencial dado ao seu retorno económico. No entanto, nem todos os óleos usados têm as características necessárias para ser transformados em biodiesel existindo outras alternativas. São exemplos de alternativas a incorporação em pasta de argila para a produção de argilas expandidas, a valorização energética através da combustão em fornos e caldeiras de várias indústrias, a produção de velas e a produção de sabonetes.

Apesar das leis aplicadas ao sector dos óleos, lubrificantes e alimentares, a adesão à reciclagem deste resíduo ainda é baixa.

1.2 Problema

A génese deste projeto partiu da intenção de melhorar o processo de recolha de óleos usados. Surgiu da observação de um problema recorrente em locais onde existem depósitos de óleo para o qual não encontram mais utilidade após a finalidade para a qual este foi inicialmente adquirido.

As empresas de recolha de óleos usados tem dois problemas centrais, eles são:

- Organizações têm dificuldades em gerir os seus recursos;
- Recolha de resíduos é pouco eficiente.

Nos pontos de recolha os depósitos de óleo usado não são monitorizados automaticamente, o que leva a que se passem vários dias sem ser recolhidos, apesar destes se encontrarem completamente cheios.

Estas empresas efetuam as recolhas com base na receção de chamadas telefónicas por parte dos produtores do resíduo ou então através de passagens frequentes das suas frotas pelos pontos de recolha. Apesar de existir a possibilidade de encontrarem resíduos a recolher, esta abordagem pode ter um elevado custo para a empresa caso não exista óleo usado a ser recolhido ou este seja insuficiente para justificar uma deslocação até aquele ponto.

1.3 Objetivos

O Smart Depot é um projeto que surge da necessidade de trazer eficiência ao processo de recolha dos OU e OAU. Pretende oferecer às empresas de recolha de óleos uma solução automática e inteligente de notificação sobre o melhor momento para recolha deste tipo de óleos. O processo torna-se mais eficiente, já que elimina a necessidade de telefonemas e evita passagens desnecessárias da recolha. Com uma reorganização da logística, prevê-se uma redução significativa dos custos de operacionalização do processo. A abordagem consiste em dispensar a intervenção humana na deteção de um depósito estar pronto a ser recolhido. Cada depósito terá um Sistema de Monitorização Remota (SMR) capaz de fornecer à empresa informação que permita agendar a sua recolha.

Um baixo custo de instalação deste sensor é um aspeto fundamental para a concretização e adesão ao Smart Depot por parte das empresas de recolha de óleos usados. Pela necessidade do SMR ter de ser instalado em todos os depósitos, o custo de cada sensor pode rapidamente inflacionar gravemente o custo global da instalação. A título de exemplo, refira-se que a terceira maior empresa de recolha de OAU em Portugal dispõe de 70.000 pontos de recolha. Uma forma de assegurar um custo final baixo dos SMR consiste na aquisição de componentes já fortemente implantados no mercado. Estes devem preferencialmente ser produzidos e vendidos por diferentes fabricantes e distribuidores, contudo, sem comprometer a sua qualidade.

Com a implementação deste projeto, espera-se obter os seguintes resultados:

1. Garantir que todas as rotas são constituídas apenas por pontos de recolha com níveis de óleo usado acima da percentagem esperada pela empresa de recolha a ele associada;
2. Evitar a necessidade dos produtores do resíduo contactar telefonicamente com a empresa de recolha para verem recolhido o seu óleo usado.

O projeto Smart Depot tem por missão desenvolver a tecnologia necessária (hardware e software) para monitorar remotamente o nível de fluidos em depósitos, eliminando a necessidade da intervenção humana, a qual usa leituras visuais aproximadas ou tardias. Alinhado com esta missão, o projeto terá que atingir os seguintes objetivos:

- Conceber e desenhar uma solução com a capacidade de medir o nível de fluidos usando sensores instalados em depósitos;
- Implementar um mecanismo que permita o envio das leituras dos SMR para um *Web Service* centralizador dos dados;
- Desenhar uma plataforma de gestão *online* capaz de receber e tratar os dados dos SMR, gerir contas e notificações de clientes e fornecer estatísticas;
- Implementar a solução desenhada para a gestão *online* da informação;
- Avaliar os resultados da solução desenhada para a leitura e comunicação dos níveis (*hardware*) e para a recolha, tratamento e notificação (API e portal web);
- Disponibilizar na sua *Application Programming Interface* (API) um serviço de planeamento otimizado de rotas baseado nos dados recolhidos pelos SMR.

1.4 Análise de Valor

A criação de valor é uma componente chave para as organizações que tentam fazê-lo através da introdução de novos produtos e serviços inovadores. No entanto o processo de inovação é bastante oneroso, constituindo um sério esforço mesmo para grandes empresas. Este esforço tem de ser bem avaliado, pesando os benefícios e os sacrifícios, internos com o desenvolvimento e externos de aceitação por parte dos clientes. Só as propostas onde os benefícios superam os sacrifícios é que podem aspirar a ser bem sucedidas, já as outras tendem a ficar-se pela ideia ou estão condenadas ao insucesso, levando muitas vezes à falência das próprias organizações.

Qualquer empresa, seja qual for a sua área de negócio, pretende que o valor dos seus produtos e serviços seja aceite e reconhecido pelos seus clientes, assim como, pelos seus próprios colaboradores (Sousa, 2010).

Apesar do Smart Depot não estar a ser desenvolvido no âmbito de uma empresa, não deixa de ser uma fonte de valor e de inovação. O valor central deste projeto encontra-se no SMR que possibilita a automatização da monitorização dos depósitos de óleo usado, evitando assim a intervenção humana nesta tarefa e as consequentes falhas ou atrasos de reação.

Como fator adicional de valor para este segmento de mercado, o projeto Smart Depot pretende também disponibilizar o serviço de planeamento otimizado de rotas, tirando partido da informação obtida a partir dos dados recolhidos pelo SMR.

1.5 Abordagem Preconizada

Pretende-se aplicar o conceito de "dividir para conquistar" ao problema, transformando-o em subproblemas menores, logo mais simples de resolver, aplicando a recursividade a fim de resolver o problema principal. Desta forma, serão identificados os requisitos dos subproblemas e procuradas tecnologias aplicáveis que ajudem a solucioná-los.

Dois destes subproblemas, facilmente identificados, são o desenvolvimento do *firmware* e a construção do SMR e o desenvolvimento do Sistema de Gestão Centralizado (SGC).

Será importante definir metodologias para a escolha dos seus componentes estruturais e eletrónicos do SMR, assim como para a escolha das ferramentas e linguagens para o desenvolvimento do código necessário ao funcionamento do SMR e do SGC.

Implementar sistemas de apoio ao versionamento do código, segurança nas comunicações e de acesso aos dados recolhidos, assim como, o planeamento e execução de testes, para a escolha de componentes e para aferir o bom funcionamento da solução desenvolvida, são fases que se pretende implementar transversalmente a todos os subproblemas.

Após a conclusão deste projeto serão identificados os objetivos atingidos, as dificuldades encontradas e será feita uma reflexão sobre o trabalho futuro necessário para dar seguimento ao trabalho desenvolvido.

1.6 Estrutura do Documento

Este documento está organizado pelos seguintes capítulos:

Introdução Neste capítulo é apresentado o problema abordado e a solução proposta por este projeto;

Contexto Este é o capítulo onde são apresentados detalhes do contexto envolvente ao mercado do tratamento de resíduos, com enfoque na recolha dos dois grupos de óleos usados, os OU e os OAU;

Estado da Arte Aqui é descrito o atual estado de outras abordagens e tecnologias aplicadas ou aplicáveis ao problema tratado por este projeto;

Avaliação das soluções e tecnologias existentes Neste capítulo são descritas as avaliações feitas as abordagens e as tecnologias existentes, identificadas no Capítulo 3 ;

Design Este capítulo descreve o design da solução proposta, identificação de requisitos, detalhes da arquitetura e modelo de dados, finalizando com a comparação e explicação das escolhas efetuadas;

Desenvolvimento e Implementação Este capítulo documenta o trabalho elaborado durante o desenvolvimento e a implementação das três aplicações tratadas por este projeto; o SMR, o Aplicação Móvel Android (AMA) e o SGC. Aqui são também abordados detalhes sobre a gestão do projeto e sobre a preparação do servidor para o SGC, em especial, detalhes relacionados com automatização da instalação das versões desenvolvidas e questões de segurança;

Testes e Resultados Os procedimentos de teste aplicados ao protótipo SMR e respetivos resultados são descritos neste capítulo;

Conclusão Neste capítulo são apresentados os objetivos atingidos e o qual o trabalho futuro necessário para dar continuidade a este projeto.

Capítulo 2

Contexto

Neste capítulo serão apresentados detalhes do contexto envolvente ao mercado do tratamento de resíduos, com mais enfoque na recolha dos dois grupos de óleos usados, os OU e os OAU. Também será esmiuçado o problema que este projeto pretende solucionar.

2.1 Detalhes do Contexto

A gestão de recursos poluentes em Portugal, em termos legislativos, é já tratada com bastante precaução. O objetivo é que haja o mínimo de poluição tanto no meio terrestre como em meio aquático.

"O Decreto-Lei n.º 239/97, de 9 de Setembro, veio estabelecer as regras básicas para a gestão de resíduos, designadamente para a sua recolha, transporte, armazenagem, tratamento, valorização e eliminação, por forma a evitar a produção de perigos ou de danos na saúde e no ambiente"(Ministério do Ambiente, 1997).

Embora já exista regime jurídico que regulamenta a forma como os óleos usados são tratados, é ainda pouco eficiente a forma como a sua gestão é feita.

A fim de reduzir o impacto ambiental provocado pela má gestão dos óleos usados, a legislação nacional estabelece um conjunto de regras que regulamentam a armazenagem, recolha, transporte e valorização deste resíduo.

Desenvolvendo-se este negócio em dois sub-segmentos distintos, o dos OU e o dos OAU, existem regimes jurídicos diferenciados para cada um deles. O Decreto-Lei n.º 153/2003(Ministérios das Cidades, 2003), de 11 de Julho (alterado pelo Decreto-Lei n.º 73/2011(Território, 2011), de 17 de Junho), estabelece o regime jurídico para a gestão de óleos lubrificantes novos e para a recolha e tratamento dos mesmos quando usados. Já o Decreto-Lei nr.º 75/2013(Global et al., 2013), de 12 de Setembro, obriga que todos os Municípios estabeleçam uma rede de recolha dos OAU, através da sua rede de recolha camarária ou sob contratação externa.

2.1.1 Restrições Existentes

Para além das restrições legais já abordadas, definidas por governos de cada país ou através de leis comunitárias, esta área da reciclagem tem também restrições específicas quanto ao armazenamento, transporte e reciclagem de tipos de óleo diferentes, sendo proibida a mistura dos dois tipos.

Exemplos destas restrições são:

OU Obrigação dos produtores de óleos lubrificantes, financiarem a sua recolha e tratamento;

OAU Obrigação dos municípios recolherem os OAU.

Devido à poluição dos recursos naturais originada pelo consumismo das sociedades mais desenvolvidas ou por questões económicas que levam ao desinvestimento na reciclagem, encontramos também restrições éticas associadas a esta área.

2.1.2 Pontos de Vista

O projeto Smart Depot pretende aplicar boas práticas aos problemas que tem para resolver. Desta forma, beneficiará do ponto de vista de outros profissionais e estudiosos sobre problemas semelhantes e recorrentes.

A ideia inicial do Smart Depot era unicamente ecológica. Contudo, logo que se identificou que a solução passava pelo desenvolvimento e produção de sistemas remotos tecnologicamente inteligentes, surgiu a necessidade de incluir também uma visão comercial.

A perspetiva comercial, despoletada pelas necessidades financeiras associadas ao projeto, obrigou a analisar a sua viabilidade, as necessidades e limitações do mercado alvo, e a consequente adaptação do Smart Depot.

Do ponto de vista dos clientes, é de extrema importância que o sacrifício do grande investimento inicial seja de retorno rápido, e para isso muito pode contribuir a capacidade de se conseguir um sistema inteligente, remoto e de baixo custo. As vantagens proporcionadas pelo Smart Depot são claras e evidentes para o cliente, já que resolvem problemas centrais ao seu negócio.

A regulação deste mercado é feita através das leis em vigor, que não se preocupam com este nível de detalhe do processo de recolha. O enfoque destas leis concentra-se principalmente que as recolhas estejam garantidas por empresas associadas à SOGILUB no que toca aos OU ou pelos municípios nos OAU, evitando assim descargas para o meio ambiente.

Para a população em geral, o conhecimento deste projeto poderá ser visto como um investimento tecnológico na área da reciclagem de resíduos, contribuindo para minimizar a pegada ambiental.

2.2 Detalhes do Problema

Os problemas identificados são:

- Organizações com dificuldades em gerir os seus recursos
- Processo de recolha de resíduos pouco eficiente

Os depósitos não são monitorizados automaticamente, o que leva a que se passem vários dias sem ser recolhidos, apesar de se encontrarem completamente cheios. As recolhas são efetuadas de forma periódica, podendo não coincidir necessariamente com a altura em que o depósito está cheio. A única forma de minimizar o problema é o produtor do resíduo requisitar a recolha por via telefónica, o que é ineficiente. Também o consumidor doméstico encontra problemas associados com a recolha dos OAU, já que esta é feita em pontos

específicos, os chamados de "óleões". Utilizando como exemplo a cidade do Porto, existem apenas 44 pontos de recolha em todo o município, levando à baixa participação da população nesta área da reciclagem.

Para além dos locais visitados após notificação telefónica, as frotas percorrem regularmente outros locais onde é expectável conseguir efetuar recolhas, baseado-se na periodicidade de visitas anteriores, embora sem garantias de sucesso. Esta abordagem, apesar de pontualmente ser eficaz, é extremamente ineficiente quando ocorrem visitas a locais da rota onde os depósitos ainda não estão em condições de serem recolhidos. Tal origina elevados custos com a frota (combustível, desgaste dos camiões, número de camiões), recursos humanos (número de motoristas e telefonistas) e desperdício de tempo.

2.2.1 Solução de Engenharia Proposta

O propósito desta solução é agilizar o processo de recolha de óleos usados. O Smart Depot pretende desenvolver os equipamentos e serviços necessários para que as empresas de recolha de óleos usados consigam saber atempadamente quando é o melhor momento para se deslocarem aos pontos de recolha a fim de encontrarem os depósitos num nível que lhes justifique os custos associados a essa deslocação.

Os clientes deste projeto estão no segmento de mercado da recolha de óleos usados, no entanto é necessário dividir este em dois sub-segmentos, o dos OAU e o dos OU. Esta divisão reflete-se também em diferenças nas características dos depósitos usados, periodicidade das recolhas, processo físico de recolha, tipo de sensores a utilizar e adaptação dos sensores aos depósitos.

A solução do Smart Depot responde a uma das principais necessidades dos seus clientes. No entanto, através da análise de oportunidades deste mercado, foi também identificada uma outra necessidade, o planeamento e otimização das rotas dos veículos de recolha. Com a inclusão de mais este serviço, também ele baseado nos dados recolhidos pelos sistemas inteligentes remotos, o sistema de informação irá satisfazer ainda melhor os requisitos do cliente.

Uma das questões que emerge perante a solução proposta pelo Smart Depot relaciona-se com a necessidade de uma reflexão cuidada sobre o custo unitário dos sensores inteligentes. Este ponto é de crucial importância para o sucesso comercial deste projeto.

A fim de conhecer melhor este mercado, visitaram-se e recolheram-se dados da empresa Filtapor que se classifica como a terceira maior do país no sector dos OAU, com cerca de 70.000 pontos de recolha, metade dos quais são visitados por empresas parceiras de menor dimensão que a representam.

Tomando como exemplo esta empresa, pode-se constatar que mesmo que esta pretendesse apenas equipar os seus locais de recolha, deixando os dos seus parceiros para uma segunda fase, seriam necessários sensores para 35.000 depósitos, o que se iria refletir num elevado investimento inicial.

Desta forma é fácil compreender que o custo do equipamento sensor é um dos pontos críticos para o sucesso do projeto, apesar da identificação da mais valia trazida com a adoção desta solução.

Um dos sócios da Filtapor, Sr. José Oliveira, informou que um outro projeto, Green box, desenvolvido em 2012 pela Universidade de Trás-os-Montes e Alto Douro (UTAD), não avançou para a comercialização precisamente por não conseguir garantir um valor aceitável por sensor remoto, inviabilizando assim a implementação do projeto neste mercado.

Apoiado na acelerada evolução tecnológica na área da computação ubíqua e consequente baixa dos custos de produção e aquisição de microcontroladores, sensores e outros componentes, é iminente o lançamento de uma solução remota de monitorização automática e notificação para resolver este problema nuclear desta área de negócio.

2.2.2 Pressupostos

O Smart Depot assume como viável a construção dos dispositivos remotos inteligentes, por considerar ter as competências técnicas necessárias e por existir de forma acessível a tecnologia necessária. O mercado tecnológico atual disponibiliza sensores capazes de efetuar as leituras de fluido pretendidas e módulos que garantem a comunicação dos dados, com diferentes tecnologias e em ambientes distintos.

Assume-se, baseada na atual conjuntura política, que se continuará a ter acesso ao mercado de consumo tecnológico asiático, pois é lá que estão concentrados os vários fornecedores dos componentes necessários à construção dos dispositivos sensores. Só dessa forma é que será possível garantir a viabilidade financeira do projeto.

O projeto assume o pressuposto de que atualmente não existe outra solução concorrente para este problema. Esta suposição é assumida com base em pesquisas feitas e na informação prestada pela empresa de recolha piloto (Filtapor), que procura ver este problema resolvido pelo menos desde 2012.

Dois fatores críticos, no que toca a pressupostos, são o já mencionado baixo custo de produção do dispositivo sensor e a confiança depositada pelos clientes do Smart Depot na ética e segurança da informação dos seus pontos de recolha, que são cruciais para a existência destas empresas.

2.2.3 Implicações

A existência de dois sub-segmentos no mercado da recolha de óleos usados faz com que o tipo de sensor de nível de fluido a usar nos SMR seja diferente, já que para a recolha dos OU são usados depósitos de 1000 L, enquanto que os depósitos usados para os OAU são de apenas 50 L.

Outra das implicações relaciona-se com a sensibilidade dos dados a que o projeto terá acesso. A informação da localização dos pontos de recolha é de extrema importância para as empresas clientes deste projeto, já que é a base de todo o seu funcionamento.

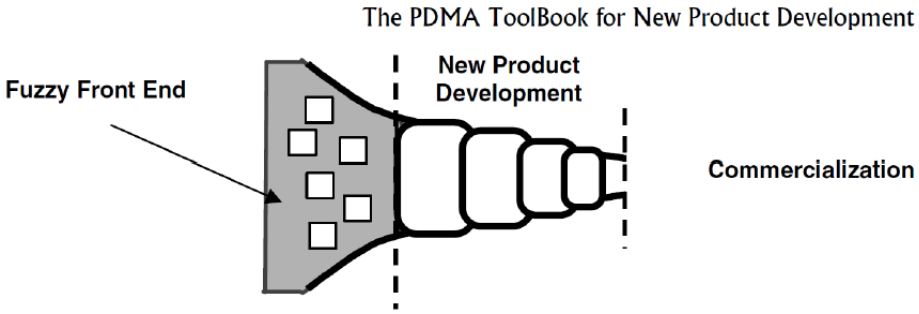


Figura 2.1: Três partes do processo de inovação (Sousa, 2010).

2.3 Análise de Valor

A corrida constante em busca de inovação, apesar de ser uma importante vantagem competitiva e diferenciadora, é uma tarefa extremamente complexa mesmo em grandes organizações.

Desenvolver e lançar novos produtos no mercado consome recursos e acarreta um elevado risco económico e da imagem da organização.

Para minimizar estes riscos, procura-se a melhoria contínua das técnicas utilizadas no processo de inovação.

Como se pode ver na Figura 2.1 este processo encontra-se dividido em três partes:

Fuzzy Front End (FEE) Onde se inicia a geração da ideia ou identificação da oportunidade;

New Product Development (NPD) desenvolvimento ou produção do conceito inovador;

Comercialização Promoção e Venda da metodologia ou produto inovador.

A primeira parte do processo de inovação, o FEE, é batizado (de Fuzzy) devido a ser nesta fase inicial onde surgem inúmeras dúvidas sobre as ideias geradas ou identificadas através das oportunidades observadas.

Para apresentar uma visão integral do processo percorrido desde a identificação do problema até à idealização do protótipo, será utilizada a notação e abordagem do modelo *New Concept Development* (NCD) de Peter Koen(Koen et al., 2001).

O NCD é um modelo que organiza o desenvolvimento do conceito no processo de inovação, define termos, identifica e ordena os vários estágios.

Este modelo divide esta fase inicial de inovação em três grupos:

- O motor do modelo
- Os cinco elementos da atividade inovadora
- Os fatores ambientais externos

Estes grupos, que se podem observar na Figura 2.2, serão detalhadas nos próximos tópicos.

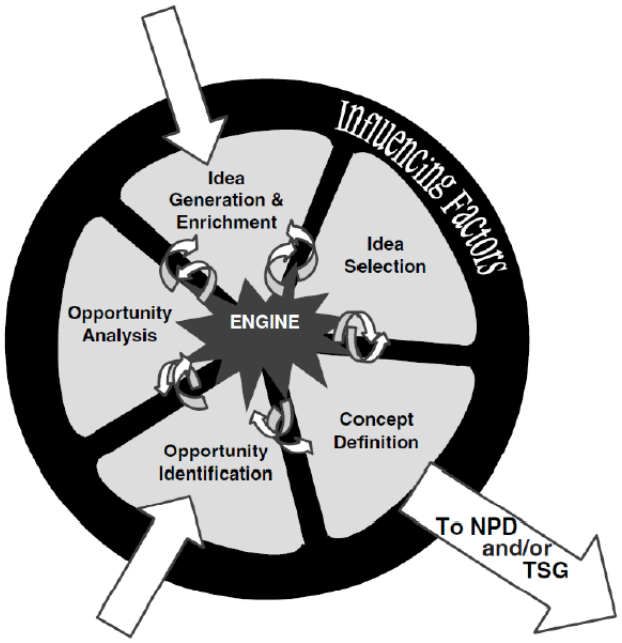


Figura 2.2: Modelo NCD (Sousa, 2010).

2.3.1 O Motor do Modelo

O motor do modelo é onde se explica a visão, estratégia e motivação para o processo inovador. É o núcleo agregador que concentra em si os fatores principais da abordagem ao problema.

2.3.2 Os Cinco Elementos da Atividade Inovadora

Os cinco elementos identificados pelo modelo NCD são:

- A identificação da oportunidade;
- Análise da oportunidade;
- Geração de ideias e o seu enriquecimento;
- Seleção das melhores ideias;
- Definição de conceito.

Identificação da Oportunidade

Procurar e compreender as necessidades e tendências das pessoas foi efetivamente o fator impulsionador da criação deste projeto.

A Smart Depot é um projeto que surge da necessidade de trazer eficiência ao processo de recolha de óleos usados (OU) e óleos alimentares usados (OAU).

Nesse contexto, a Smart Depot pretende oferecer às empresas de recolha de óleos uma solução inteligente de notificação sobre o melhor momento para recolha de óleos usados.

Este projeto partiu da intenção de melhorar o processo de recolha de óleos usados.

Figura 2.3: Contentor para OU.

Após a observação de alguns pontos de recolha de óleos usados, identificou-se que existem organizações e empresas que disponibilizam contentores, como o apresentado na Figura 2.3 onde os seus frequentadores, estudantes, docentes ou funcionários, podem colocar recipientes com óleo usado com o objetivo de ver este resíduo reciclado. No entanto constata-se que, apesar destes contentores se encontrarem cheios, decorriam vários dias até os mesmos serem recolhidos.

Procurou-se obter mais informações sobre o processo e verificou-se que a empresa responsável pela recolha realizava passagens periódicas, que nem sempre coincidiam com a existência de depósitos para recolher e que a única forma de minimizar o problema era alguém da organização avisar estas empresas telefonicamente.

As empresas de recolha, após serem avisadas da existência de resíduos, não levavam mais de 2 dias úteis para recolher os mesmos. Assim, pode-se concluir que não se tratava de um problema de eficácia, mas sim de eficiência.

Considera-se que atualmente, com a tecnologia disponível, é possível encontrar uma forma automática destas empresas serem notificadas da altura ideal para se deslocarem aos pontos de recolha de óleos usados. Isto tornaria todo o processo de reciclagem mais rápido e eficiente.

O acesso remoto à informação do nível de óleo existente nos depósitos iria melhorar o processo de recolha pois reduziria custos com telefonemas e com toda a logística despendida nas passagens desnecessárias em certos pontos de recolha.

As empresas de recolha de óleos usados (alimentares e lubrificantes) estão, na maior parte dos casos, dependentes dos contactos (normalmente telefónicos) por parte dos seus clientes para a atualização diária das rotas a percorrer pela sua frota de camiões.

Para além dos locais identificados pelos contactos, as frotas percorrem regularmente outros pontos onde empiricamente (por experiência) normalmente conseguem efetuar recolhas.

Este processo, apesar de pontualmente poder ser eficaz, é extremamente ineficiente, originando grandes custos com a frota (combustível, desgaste dos camiões, uma frota maior do que a realmente é necessária), recursos humanos (maior número de motoristas e telefonistas) e desperdício de tempo.

O problema identificado é a necessidade da empresa de recolha ser avisada, sem intervenção humana, logo que o contentor ou depósito está cheio.

Para este processo ser automático, será necessário equipar o contentor de recolha com um sensor, ou vários, que detetem o estado de cheio, e fazer com que essa informação chegue à empresa de recolha.

O desenvolvimento de um dispositivo sensor deste tipo só é interessante para as empresas de recolha de óleos usados se, para além de cumprir as funções pretendidas, tiver um custo de instalação em cada ponto de recolha baixo o suficiente para compensar a mudança do processo atual de recolha, baseado em passagens periódicas e em chamadas telefónicas por parte dos seus clientes.

Porque as empresas de recolha de óleos usados tem vários pontos de recolha para equipar, o preço final de cada depósito sensor de alerta é um fator crítico que pode inviabilizar a solução.

Na tentativa de conseguir um custo baixo nos sensores usados, estes devem ser componentes facilmente encontrados no mercado de consumo, preferencialmente produzidos e vendidos por diferentes fabricantes e distribuidores, não descurando a qualidade dos mesmos.

Foram analisados diferentes sensores de medição do nível de líquido nos depósitos. Sondas baseadas na condutividade elétrica(*Oil Condition Monitoring Using Electrical Conductivity*), sensores de ultrassom, por infravermelhos, eletromagnéticos e sensores de peso.

Análise da Oportunidade

De modo a sustentar, com base em números, a identificação da oportunidade atacada pelo Smart Depot, consultaram-se dados públicos dos dois sub-segmentos do mercado de recolha de óleos usados.

A SOGILUB disponibiliza anualmente os dados das recolhas efetuadas de OU. Pode-se observar na Figura 2.4 a evolução dos valores desde 2006 até 2015 (SOGILUB, 2015).

Como se pode ver pela análise do sub-segmento dos OU, foram recolhidos nos últimos nove anos uma média de 28 mil toneladas por ano, tendo sofrido uma queda nos últimos cinco anos, mas mantendo-se acima das 24 mil toneladas em 2015.

No sub-segmento dos OAU uma boa fonte de informação é o projeto Recoil (Recoil, 2017) co-financiado pelo *The Intelligent Energy Europe Programme* da União Europeia. No sítio deste projeto está disponível um estudo europeu (Mcatalao, 2013) que mostra, como é apresentado na Figura 2.5, que a principal forma de descartar os OAU é através da rede de saneamento (38,3%). Mas a segunda maior forma de descarte é efetivamente através dos mecanismos de reciclagem, com 22,3% dos casos.

Pelo elevado volume de óleo usado e também por existir a necessidade de o reciclar por imposição legal e em prol do ambiente, tornar a recolha deste resíduo mais eficiente é de facto uma necessidade urgente.

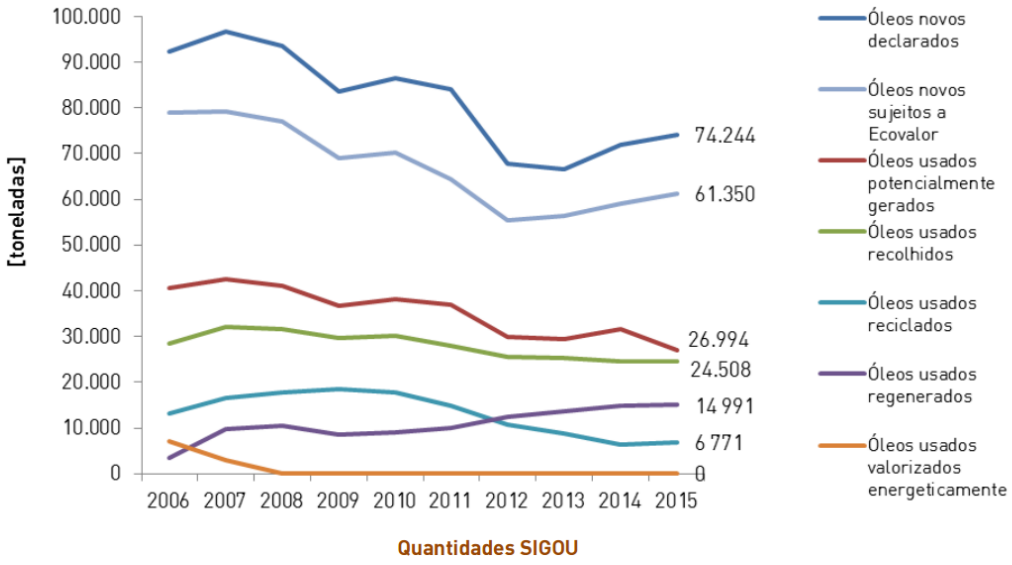


Figura 2.4: Dados SOGILUB 2006-2015 (SOGILUB, 2015).

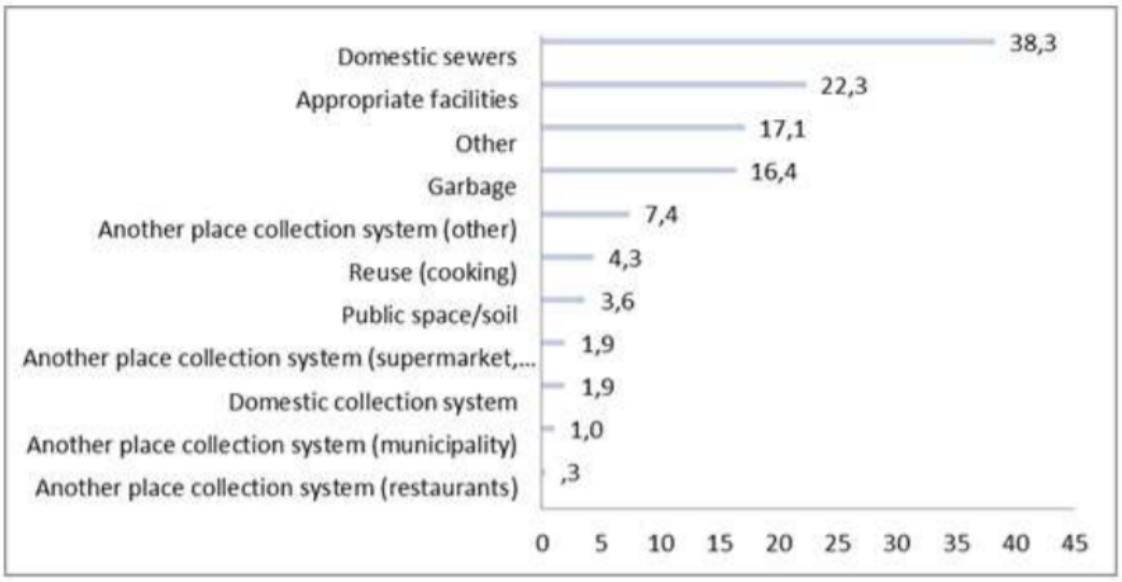


Figura 2.5: Formas de descarte dos OAU.

Geração de Ideias e o seu Enriquecimento

Após a análise do problema identificou-se a necessidade de conseguir efetuar a monitorização do nível dos depósitos sem intervenção humana. Para atingir esse objetivo foi necessário avaliar a oferta no mercado dos sensores de nível de fluidos e ponderar quais se adaptam melhor a cada um dos sub-segmentos (OAU e OU).

Para a seleção e enriquecimento das ideias foi importante a informação que se obteve nas reuniões com os representantes da empresa piloto (Filtapor), assim como a visita às suas instalações, onde foi possível ver o processo de recolha na prática.

Seleção das Melhores Ideias

Pretende-se encontrar a melhor solução, para cada sub-segmento, através da construção e testes de SMR protótipos aplicados aos depósitos habitualmente utilizados.

Definição de Conceito

Com o Smart Depot pretende-se que as aplicações de gestão das empresas de recolha possam aceder a um serviço online para obter a informação dos pontos de recolha que devem ser visitados, por já se encontrarem cheios com óleos usados.

2.3.3 Os Fatores Ambientais Externos

Fatores Sociais, Tecnológicos, Económicos, Ambientais e Políticos, devem ser tidos em conta na contextualização da solução inovadora idealizada. As soluções encontradas tiveram estes fatores de influência em conta.

Os fatores externos que influenciaram esta oportunidade são a massificada utilização de óleos, lubrificantes e alimentares, nas mais variadas indústrias, nos transportes, ferramentas e equipamentos domésticos e mesmo na nossa alimentação. Este elevado nível de consumo de óleo origina um grave problema de reciclagem, não só pela dificuldade do processo, mas especialmente pela baixa adesão à reciclagem deste tipo de resíduos.

Com o objetivo de tentar minimizar a pegada ambiental, o governo Português identificou dois dos mais significativos focos de poluição nesta área, os lubrificantes e os óleos alimentares usados na restauração e estabeleceu um regime jurídico com o Decreto-Lei n.º 153/2003, de 11 de Julho, baseado na Diretiva do Conselho da Comunidade Económica Europeia de 16 de Junho 1975 relativa à eliminação dos óleos usados (75/439/CEE).

O segmento dos óleos usados divide-se em dois grandes grupos, os óleos lubrificantes e os alimentares. Os óleos lubrificantes são utilizados na indústria e no sector automóvel e os óleos alimentares na restauração e mercado doméstico.

O tratamento e recolha é também diferenciado, estando proibida a mistura de óleos provenientes de grupos diferentes.

Tabela 2.1: Benefícios e Sacrifícios.

Domínio / mbito	Serviço	Relacionamento
Benefícios	Monitorização automática Sem intervenção humana Redução da dependência dos poluidores Redução de chamadas telefónicas Rapidez na recolha Elimina visitas desnecessárias Redução de custos Rotas planeadas e otimizadas Redução da pegada ambiental	Confiança Imagem Parceria Ambiental
Sacrifícios	Investimento inicial Custo do serviço Manutenção dos SMR Formação	

2.3.4 Conceitos de Valor do Projeto Smart Depot

Uma proposta de valor bem definida permite identificar claramente e inequivocamente aquilo que se vai oferecer ao cliente e deve, em termos ideais, ser associado a um benefício. Além de que, como já abordado na Secção 1.4 do capítulo anterior, a criação de valor é uma componente chave para sustentabilidade e crescimento de qualquer organização.

A solução proposta pelo Smart Depot aos seus clientes é simples e objetiva, indo de encontro a um dos seus principais problemas e resolvendo-o de forma automática, sem necessidade de intervenção humana. Com o SMR do Smart Depot os seus clientes conseguem saber o momento ideal para efetuar as recolhas dos óleos usados, permitindo-lhes uma economia de tempo e recursos, que se traduz numa significativa economia financeira.

Outro benefício que o Smart Depot pretende oferecer é o planeamento otimizado de rotas, também ele baseado nos dados recebidos dos SMR. Esta oferta adicional ataca outro dos problemas nucleares deste mercado, quais as rotas a percorrer diariamente.

Valor para o Cliente

Esta solução apresenta-se de elevado valor para o mercado da recolha de óleos usados, já que resolve dois problemas nucleares deste negócio. Com o Smart Depot estas empresas reduzem a dependência para com os produtores de resíduos, não necessitando da iniciativa destes para saber quando podem efetuar uma recolha com sucesso.

O elevado interesse demonstrado pelo representante da empresa piloto, Filtapor, é um bom indicativo da valorização que o cliente dá a este produto inovador e atualmente sem concorrência no mercado. Esta valorização por parte do cliente deve-se à clara identificação dos benefícios que esta solução traz ao seu negócio em comparação com os sacrifícios necessários.

Na Tabela 2.1 efetua-se uma comparação entre os benefícios e os sacrifícios para os serviços prestados pelo Smart Depot e também quanto aos benefícios da relação comercial entre os clientes e este projeto.

Valor Percecionado

O valor percecionado pelos clientes deste projeto, apesar de variar, aparenta tornar-se numa mais valia para as empresas deste mercado. Isto deve-se à diferença entre o reduzido nível de sacrifícios, em comparação com os benefícios trazidos por esta nova abordagem logística à recolha de resíduos.

A tecnologia implementada pelo Smart Depot proporciona uma elevada redução dos custos com mão de obra, recursos e também uma redução da atual elevada dependência destas empresas para com os seus clientes, produtores dos resíduos, levando à oferta de outros serviços para garantir o contacto telefónico ou a própria recolha e mesmo chegando, em casos de grandes produtores de resíduos, a negociarem a compra do óleo usado.

Perspetiva Longitudinal de Valor

Como forma de apresentar a perspetiva longitudinal de valor dos serviços Smart Depot serão apresentados os benefícios e sacrifícios do cliente nas quatro fase do ciclo comercial: antes da compra; no ato da compra; depois da compra; após usar os serviços.

Antes da compra As empresas de recolha de óleos usados conseguem identificar de forma bem clara as vantagens de adesão aos serviços Smart Depot, no entanto, na fase da análise da compra, necessitam avaliar especialmente o custo de implementação dos SMR e se o farão de forma faseada ou integral.

Na implementação Esta é a fase mais crítica para o cliente, já que será este o momento do principal investimento na aquisição dos dispositivos SMR e, quantos mais pontos de recolha pretenderem equipar, maior será o investimento. O cliente também necessitará investir na formação da sua equipa de colaboradores a fim destes se preparem para as alterações logísticas que serão introduzidas no processo. O preço unitário do SMR é um fator crítico desta fase.

Durante a utilização Depois de todos os pontos de recolha estarem equipados com os SMR, é o momento em que o cliente vê refletir-se no seu negócio os benefícios do Smart Depot.

Depois de experimentar Depois da experiência de utilização dos serviços Smart Depot, o mercado de recolha de óleos usados não será o mesmo, passando a existir as empresas que usam um sistema de monitorização remota e as restantes que não acompanharam ainda o desenvolvimento tecnológico. A adesão a este processo de automatização da monitorização pode levar a que as empresas que o fizerem primeiro possam alargar o seu número de pontos de recolha, criando sérias dificuldades de subsistência às restantes empresas deste mercado.

2.3.5 Proposta de Valor do Projeto Smart Depot

A solução Smart Depot promove, junto das empresas de recolha de óleos usados, uma nova abordagem do processo de recolha (*enabling function*), identificando o momento ideal para este ser efetuado. Isto é conseguido através da leitura automática do nível dos depósitos com base em sensores inteligentes. Esta nova abordagem proporciona também poupança de tempo e recursos da empresa o que se reflete em benefícios económicos.

Outra das mais-valias deste projeto é no campo ambiental já que, através da agilização do processo de recolha dos óleos usados, torna a recolha mais eficiente, contribuindo assim para a redução da pegada ambiental.

2.3.6 Modelo de Negócio do Projeto Smart Depot (CANVAS)

Na Tabela 2.2 está representado um possível modelo de negócio utilizando a estrutura *Business Model Canvas* proposta por Alexander Osterwalder (Alexander Osterwalder, 2008).

2.3.7 Rede de Valor

O Smart Depot é atualmente um projeto pessoal e académico, não estando integrado no âmbito de nenhuma empresa ou organização. Assim, não é possível analisar o seu valor numa componente de *networking* entre pessoas ou departamentos de uma organização.

No entanto, se este projeto estivesse enquadrado numa empresa, departamentos como o de "Inovação", "Desenvolvimento de produto" e "Marketing" trariam grandes vantagens de negócio. As vantagens seriam maiores, quanto maior fosse o intercâmbio de conhecimentos e ideias entre estes departamentos.

Há no entanto a criação da rede de valor envolvente ao mercado onde este projeto se insere, entre os seus intervenientes. Isto é feito através de parcerias com fornecedores, da fidelização de clientes e tornando mais fácil a aplicação de leis comunitárias.

Pode-se ainda incluir na rede de valor criada por este projeto o conhecimento transmitido aos meios académicos da engenharia e às comunidades ligadas à *Internet of Things* (IoT).

2.3.8 Analitic Hierarchy Process (AHP)

Um método utilizado no processo de avaliação multi-critério que é bastante flexível, por permitir o uso de critérios qualitativos e quantitativos, é o Método de Análise Hierárquica AHP, criado por Thoma L. Saaty (Saaty, 1990). Este método organiza os fatores numa estrutura hierárquica.

A utilização deste método prende-se também ao facto de se atribuir pesos diferentes para cada critério. Para classificar os níveis de importância nas comparações será usada a Escala Fundamental (Saaty, 1990).

No decorrer deste projeto pretende-se usar o método AHP para selecionar o tipo de tecnologia de comunicação, *Machine-to-Machine* (M2M), do SMR utilizando critérios, com pesos distintos, como: Fiabilidade; Custo; Consumo energético.

<table>
<tr>
<td rowspan="2">Parceiros-chave

Empresas que fazem os depósitos;

Acordo com empresas de componentes eletrónicas;

Câmaras municipais;

Associação Portuguesa de Hotelaria, Restauração e Turismo;

ANECRA (Associação, nacional, das empresas do comércio e reparação automóvel).</td>
<td>Atividades-chave

Sistema monitorização remota;

Recolha e tratamento dos dados;

Informar os clientes de quando tem de efetuar as recolhas;

Proposta de rotas otimizadas.</td>
<td rowspan="2">Proposta de valor

Para tornar mais eficiente a recolha de óleos usados é proposto, um sistema de informação baseado em sensores inteligentes que monitorizam o nível dos depósitos e transmitem essa informação para uma plataforma central.

A prestação deste serviço visa poupar tempo e dinheiro às empresas de recolha de resíduos, proporcionando uma nova forma de agilizar a recolha dos óleos, contribuindo também para a redução da pegada ambiental.</td>
<td>Relacionamento com clientes

Suporte;

Atendimento pós-venda;

Departamento comercial;

Ciclos de fidelização.</td>
<td rowspan="2">Segmentos de clientes

Empresas de recolha de óleos usados;

Empresas de recolha de óleos alimentares usados.</td>
</tr>
<tr>
<td>Recursos-chave

Plataforma tecnológica (backoffice);

Sensores;

Depósitos;

Internet.</td>
<td>Canais

Disponibilização de uma plataforma online;

Demonstrações nos clientes;

Feiras;

Flyers;

D2D – Door 2 door.</td>
</tr>
<tr>
<td colspan="3">Custos

Equipa de desenvolvimento de software; Na compra dos materiais necessários (componentes para os sensores); Deslocações ao cliente; Publicidade, marketing.</td>
<td colspan="2">Receitas

Venda do produto/serviço; Suporte/manutenção; O cliente pagar por mês ou por pedido.</td>
</tr>
</table>

Tabela 2.2: Modelo CANVAS

Capítulo 3

Estado da Arte

Neste capítulo é descrito o atual estado de outras abordagens e tecnologias aplicadas ou aplicáveis ao problema tratado por este projeto.

3.1 Soluções e Abordagens Existentes

Ao analisar o mercado foram identificados dois projetos que apesar de abordarem o problema aqui tratado não revelaram ser verdadeiros concorrentes do Smart Depot, como é apresentado no Capítulo 4.

1. Aplicação Android Smart Lubi, da SOGILUB;
2. Projeto académico Green Box, na UTAD (UTAD, 2012).

3.1.1 Smart Lubi

A SOGILUB, sociedade por quotas sem fins lucrativos, é a entidade gestora do Sistema Integrado de Gestão de Óleos (lubrificantes) Usados. Foi licenciada para o efeito por despacho conjunto dos Ministérios da Economia e do Ambiente e do Ordenamento do Território e Energia (Sogilub, 2017).

A SOGILUB tem de dar cumprimento às obrigações vigentes em matéria de gestão de óleos lubrificantes usados, facilitando o cumprimento das obrigações legais e ambientais das empresas produtoras aderentes ao sistema. Para garantir a Gestão de resíduos de óleos lubrificantes em Portugal, implementou um sistema integrado de gestão tendo lançado recentemente uma aplicação móvel.

A Smart Lubi é uma aplicação Android destinada a permitir a comunicação, via Internet, com a SOGILUB. Esta aplicação permite, aos produtores de OU associados à SOGILUB, informar o momento a partir do qual necessitam que os seus OU sejam recolhidos, evitando a necessidade de efetuar esse contacto pelo telefone.

Esta aplicação não resolve o problema das empresas de recolha de resíduos já que a notificação, alertando para a recolha, continua a necessitar da intervenção humana.

3.1.2 Green Box

O projeto Green Box foi desenvolvido em 2012 por um grupo de investigadores da UTAD, numa iniciativa Quadro de Referência Estratégica Nacional (QREN), financiada pela Agência de Inovação (ADI), Programa Operacional do Norte e cofinanciada pelo Fundo Europeu de Desenvolvimento Regional.

O foco do Green box é o segmento dos OAU e, dentro deste, apenas nos estabelecimentos HORECA e industriais, descurando o sector doméstico.

Tal como o Smart Depot, este projeto desenvolveu um dispositivo sensor remoto que envia dados, via SMS, para uma base de dados, mas neste caso, o sensor remoto encontra-se a equipar uma caixa separadora de óleos alimentares que é instalada no sistema de canalização do lava-loiça.

O Green box inclui um sistema de gestão e otimização de rotas.

Este projeto académico aparenta estar parado desde 2012, ano da entrega das dissertações de Mestrado em Engenharia Informática dos dois alunos intervenientes no projeto, André Sousa e José Faria.

3.2 Tecnologias de Conetividade Sem Fios para o SMR

O artigo *New Industrial Internet of Things Products* divulgado pela LinkLabs apresenta-se como um guia útil para engenheiros e decisores na seleção da tecnologia de comunicações M2M sem fios a utilizar em produtos de computação ubíqua aplicados à indústria (Link Labs, 2016).

Este artigo aborda várias opções de conetividade sem fios, apontando as suas vantagens, desvantagens e aplicações habituais dos mesmos.

Da lista abordada pelo artigo destacam-se, para possível utilização neste projeto, o Wi-Fi e o *Global System for Mobile Communications* (GSM). Outras tecnologias de comunicação sem fios que também se ponderou utilizar foram, o Bluetooth na sua versão *Bluetooth Low-Energy* (BLE) e o ZigBee. No entanto, verificou-se que nenhuma destas duas tecnologias se adequava ao projeto, pela curta distância de propagação e pelo elevado custo dos equipamentos respetivamente.

O Wi-Fi tem várias variantes ao seu standard 802.11, desde **b** até à **ac** de alta performance. A grande vantagem da utilização do Wi-Fi neste projeto é o facto desta se basear numa implementação completa do protocolo *Transmission Control Protocol/Internet Protocol* (TCP/IP) que permite uma conetividade direta com o sistema de informação.

Já o GSM, tem na sua maior força a sua maior fraqueza, já que, apesar de ter a vantagem da grande cobertura da sua rede celular, esta rede não permite uma ligação direta à rede TCP/IP. A utilização desta tecnologia pode ser muitas vezes a única solução de comunicação do SMR para uma dada localização. No entanto, a sua utilização pode ficar condicionada se o custo mensal a pagar à operadora GSM for demasiado alto para o orçamento do projeto, inviabilizando o mesmo.

	Realtek RTL8710	Espresif ESP8266
Package	QFN-48 (6×6 mm)	QFN-32 (5×5 mm)
CPU	ARM Cortex M3 @ 166 MHz	Tensilica LX106 @ 80 / 160 MHz
RAM	48KB available to user	36KB available to user
Flash	1MB Built-in	1, 2, 4, 8 or 16 MB
WiFi	802.11n up to 150 Mbps, 802.11g up to 54 Mbps	802.11n up to 65 Mbps, 802.11g up to 54 Mbps
GPIO	Up to 21	Up to 17
I2C	Up to 3	Up to 1
PCM	Up to 2	None
PWM	Up to 4	
UART	2x high-speed UART, 1x low-speed UART	Up to 2x UART
Power	Voltage: 3.0 to 3.6V; Current: 80 mA	
Temperature range	-40 to 125 °C	
Standard certifications	FCC/CE/TELEC/SRRC/ WiFi Alliance	FCC/CE/TELEC/SRRC

Tabela 3.1: Especificações do RTL8710 e do ESP8266

Relativamente aos *chipsets* destas duas tecnologias, o utilizado pelo GSM é significativamente mais caro que o do Wi-Fi, sendo também o do Wi-Fi menos exigente em termos de consumo energético.

3.2.1 Chipset Wi-Fi

Da oferta no mercado de baixo custo (abaixo dos dois dólares) para Microcontrolador (MCU) com capacidades Wi-Fi com pilha TCP/IP completa destacam-se o chipset ESP8266 da Espressif Systems (Espressif Systems, 2014) e o RTL8710 da Realtek Semiconductor Corp (Realtek Semiconductor Corp, 2017).

O ESP8266 foi lançado em 2014 invadindo mercado das aplicações IoT graças ao seu baixo custo, tendo-se tornado rapidamente o chipset dominante também no mercado amador de IoT graças à sua grande comunidade de programadores.

Devido a este grande sucesso de um chip produzido por um fabricante chinês, até então desconhecido, o gigante Realtek decidiu criar uma alternativa neste mercado, anunciando em janeiro deste ano (2017) no *Consumer Eletronics Show* (CES) o seu novo módulo Wi-Fi com capacidades de processamento, juntando ao RTL8710 um ARM Cortex-M3.

A Tabela 3.1 mostra um comparativo entre as especificações dos dois módulos, publicado pelo website chinês elecfans (Eelecfans, 2017).

3.2.2 Chipset GSM

As opções GSM de baixo custo limitando-se a três fabricantes. O chip N590 da Neoway Tech (Neoway Tech, 2010), o A6 GPRS/GSM da Ai Thinker (Ai Thinker, 2017) e o SIM800

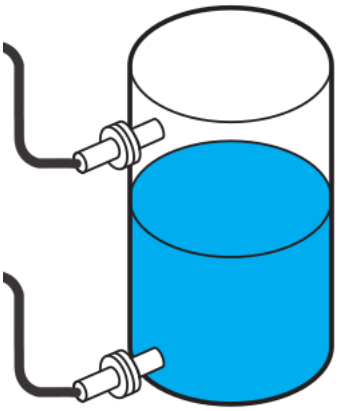


Figura 3.1: Sensor com interruptor de bóia (KING-GAGE, 2013).

da SIMcom (SIMcom, 2017). Os três módulos encontram-se no mercado com preços entre os quatro e os seis dólares.

Todos os módulos GSM tem variantes para as duas gamas de frequências usadas mundialmente, sendo que o SIM800 suporta Quad-band 850/900/1800/1900MHz, justificando assim ser o mais caro dos três.

3.3 Sensores para Leitura de Nível de Fluido

Os sistemas de medição de nível de líquidos podem utilizar dois tipos de sensores:

- Sensores de medição de nível de ponto
- Sensores de nível contínuo

Os sensores de medição de nível de ponto são utilizados para detetar uma única altura de fluido. Normalmente são utilizados para detetar situações limite, como o enchimento excessivo ou para dar o alerta de que foi atingido um nível demasiado baixo.

Os sensores de nível contínuo são mais complexos, normalmente envolvendo mais tecnologia na sua construção. Permitem uma monitorização constante, numa gama de valores, em vez de só num ponto, e estabelecem uma relação entre os valores de saída e o nível de fluido do depósito. Podem ser diferenciados, não só pela amplitude da gama de valores, mas também pela resolução que disponibilizam.

Dos sensores analisados, apenas os com interruptor de bóia e os baseados no efeito de Hall são usados para medição de nível de ponto. Todos os restantes podem ser usados para obter leituras de nível contínuas.

3.3.1 Sensores de Nível com Interruptor de Bóia

Estes sensores, representados na Figura 3.1, apenas tem dois estados, aberto ou fechado, funcionando com o auxílio de um interruptor comandado por uma boia.

Exemplos de áreas/problemas onde são utilizados:

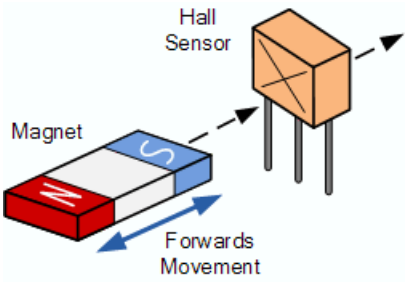


Figura 3.2: Efeito Hall (Electronics-tutorials, 2017).

- Deteção de recipiente vazio ou demasiado cheio;
- Deteção de pontos de nível intermédio.

3.3.2 Sensores de Nível por Efeito Hall

Estes sensores tiram partido do efeito magnético para controlar a abertura ou fecho do seu interruptor. Como apresentado na Figura 3.2, estes sensores funcionam por reação à proximidade quando detetam um campo magnético.

Exemplos de áreas/problemas onde são utilizados:

- Deteção de recipiente vazio ou demasiado cheio;
- Deteção de pontos de nível intermédio.

3.3.3 Sensores de Nível por Ultrassons

A Figura 3.3 mostra um sensor de ultrassons, do tipo nível contínuo, que são os mais indicados para situações onde se pretende obter leituras sem que o sensor esteja em contacto com o líquido. Exemplos de líquidos onde a leitura sem contacto é uma necessidade são os corrosivos ou líquidos a altas temperaturas. Fatores como a condutividade não afetam este tipo de sensores.

O funcionamento é baseado no tempo de propagação que o ultrassom emitido demora a voltar ao refletir na superfície nivelada para onde se encontra dirigido.

Exemplos de áreas/problemas onde são utilizados:

- Deteção de obstáculos;
- Medição de distâncias.

3.3.4 Sensores de Nível por Condutividade

Como é apresentado na Figura 3.4, este tipo de sensor avalia diferenças de condutividade medindo a resistência elétrica ao longo da sua vareta para determinar o nível de fluido. Este tipo de sensor fornece leituras fiáveis e baixa manutenção, mesmo quando utilizado com líquidos sujos ou viscosos.

Exemplos de áreas/problemas onde são utilizados:

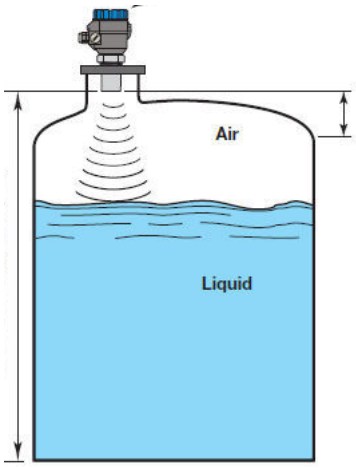


Figura 3.3: Sensor de ultrassom (Direct Industry, 2017).

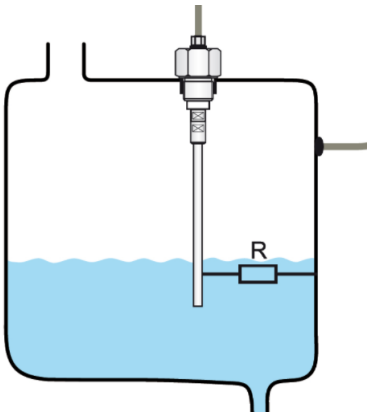


Figura 3.4: Sensor de nível por condutividade (PVL, 2017).

- Medição de nível de líquidos.

3.3.5 Sensores de Carga (Load Cells)

Um sensor de carga transforma a pressão/força sobre ele sofrida num sinal elétrico.

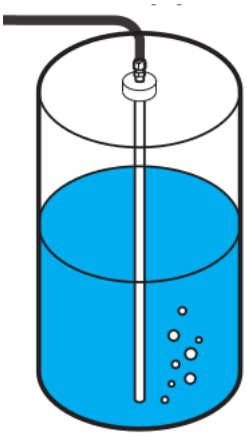


Figura 3.5: Sensor de nível por pressão (KING-GAGE, 2013).

Existem três tipos principais de sensores de carga para aferir a pressão traduzindo-a numa medida passível de ser lida, passamos a descrevê-los:

Sensores de Carga Pneumáticos

A pressão de ar aplicada à extremidade de um diafragma é lida por um manómetro ao escapar por um bico localizado na parte inferior do sensor pneumático, ver Figura 3.5.

Sensores de Carga Hidráulicos

Este tipo de sensores, apresentado na Figura 3.6, utilizam um sistema de pistão e cilindro para transmitir a alteração de pressão. O movimento do pistão reflete alterações, num diafragma, que são lidas pelo sensor.

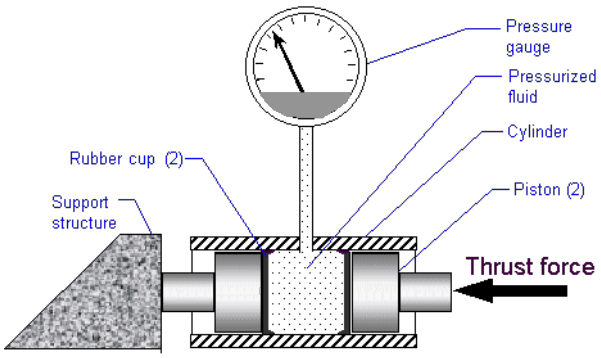


Figura 3.6: Sensor de carga hidráulico (*Getting Started with Load Cells*).

Sensores de Carga por Tensão

Este tipo de sensores, representado na Figura 3.7, tem a capacidade de fazer variar um sinal elétrico com base na deformação de uma zona, ou mais, onde é aplicada tensão (uma força mecânica).

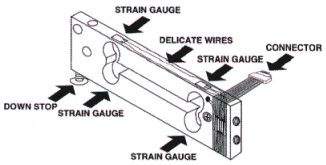


Figura 3.7: Sensor de carga por tensão (*Getting Started with Load Cells*).

Nos sensores de carga por tensão em forma de barra, as células de medição são dispostas numa formação em "Z" para que o efeito de torque se aplique à barra em quatro pontos, dois dos quais medem a compressão e os outros dois a tensão sofrida pela barra. Com esta disposição dos pontos sensores obtém-se uma ponte completa, garantindo-se medições bastante precisas mesmo com ligeiras alterações de pressão.

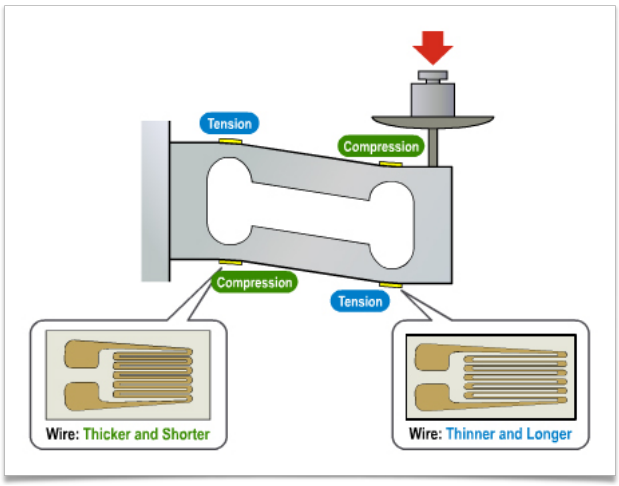


Figura 3.8: Sensor de carga em barra (*Getting Started with Load Cells*).

A Figura 3.8 apresenta em detalhe a aproximação e afastamento dos fios usados nos sensores de carga por tensão que origina a variação da resistência elétrica medida quando são aplicadas forças.

Este tipo de sensores de carga pode variar em material, tamanho e configuração mecânica, o que pode levar a diferentes cargas máximas e diferentes resoluções.

Os sensores de carga hidráulicos e pneumáticos são utilizados essencialmente em ambientes industriais, sendo a sua qualidade e custo elevados. Já os sensores de carga por tensão são possíveis de encontrar aplicados em diversas áreas, inclusive no uso doméstico, em balanças digitais de cozinha ou casa de banho.

A massificação e consequente baixa de custo no fabrico deste tipo de sensores de carga torna viável a sua utilização neste projeto, desta forma decidiu-se aprofundar o seu estudo.

Um medidor de tensão é um dispositivo que mede alterações na resistência elétrica em resposta e proporcionalmente a uma pressão a ele aplicada.

Este tipo de dispositivos sensores são normalmente compostos por um fio muito fino disposto num padrão em forma de grelha. Ao ser aplicada tensão, ocorre a deformação do material sensor, existindo uma aproximação ou afastamento deste fio que provoca a variação da resistência elétrica criada à passagem da corrente.

Cada medidor de tensão tem uma sensibilidade de escala (resolução) diferente que se exprime quantitativamente como fator de tensão (GF, gauge factor). O fator de tensão é definido pelo fator entre a alteração da resistência elétrica e a alteração da largura (espaçamento entre os fios) do sensor. Para materiais metálicos o fator de tensão ronda as 2 unidades.

As medições da tensão sofrem variações muito pequenas o que faz com que seja necessária a utilização de módulos amplificadores de sinal. Só com a ajuda destes é que se torna possível trabalhar com os valores lidos por este tipo de sensores.

O HX711 é um dos Circuito Integrado (IC) habitualmente utilizados na construção destes módulos, na Figura 3.9 pode-se observar um módulo amplificador de sinal da SparkFun onde foi usado este IC.

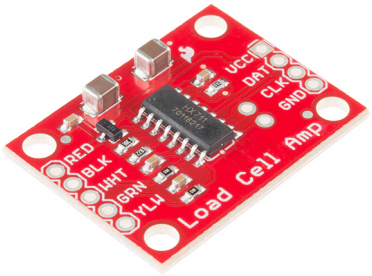


Figura 3.9: Módulo Amplificador de sinal HX711 da SparkFun (*Getting Started with Load Cells*).

Uma forma de tornar as pequenas variações de resistência elétrica mensuráveis é utilizando um circuito de losango ou ponte de Wheatstone. Este circuito, da Figura 3.10, é constituído pela conjugação de quatro resistências das quais se sabe o valor de equilíbrio.

Para uma tensão (Vin) de entrada constante, se os valores de R1/R2 forem iguais a R3/R4, a tensão de saída (Vout) será de zero Volt. Estas resistências podem ser substituídas pelas células de carga, e desta forma é possível avaliar a variação na tensão (Vout) de saída, fazendo a devida conversão para o valor da força de tensão (mecânica) aplicada.

Para que exista esta relação entre a tensão elétrica à saída do circuito e uma dada tensão física há que calibrar os sensores da carga, determinando qual o valor da tensão elétrica de saída quando os sensores se encontram em repouso. Esta calibração é necessária para que as forças, por exemplo o próprio peso, aplicadas pelos restantes peças que constituem o equipamento construído não influenciem os valores lidos sem carga adicional.

3.3.6 Outro Tipo de Sensores

Outro tipo de sensores a mencionar são os sensores de nível por infravermelhos, no entanto estes não foram alvo de maior análise já que podem apresentar falhas nas leituras devido à

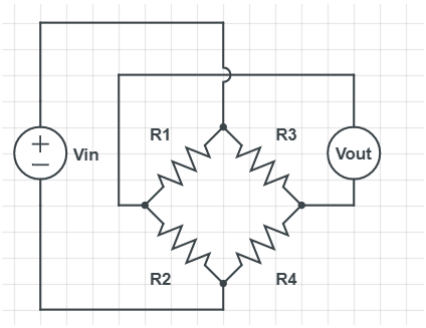


Figura 3.10: Circuito elétrico de uma ponte de Wheatstone desenhado em circuitlab.com.

alta sensibilidade a interferências rádio elétricas externas.

3.4 Serviço de Planeamento Otimizado de Rotas

O planeamento otimizado de rotas é um problema sobre o qual já se encontram estudos desde o Século XIX, formulados pelos matemáticos W.R. Hamilton e Thomas Kirkman.

Este problema, designado de *Travelling Salesman Problem* (TSP), é considerado de elevada complexidade, sendo classificado como um problema NP-hard na área das ciências da computação (*Theoretical computer science*).

As três abordagens a problemas NP-hard são:

- Algoritmos exatos, que apenas são rápidos para problemas de pequenas dimensões;
- Algoritmos heurísticos, que apresentam normalmente boas soluções mas que não garantem a solução ótima;
- Abordagem híbrida, onde são utilizadas as duas anteriores aplicadas a subproblemas.

A solução matemática atualmente aplicada baseia-se no cálculo da distância de todos os percursos possíveis, comparando-os para selecionar o mais curto. No entanto, mesmo utilizando os mais rápidos computadores do mundo, é difícil obter em tempo útil a melhor rota, quando se pretende visitar mais do que meras dezenas de destinos por rota.

O interesse na descoberta de um algoritmo que resolva este problema de forma eficiente é tal que é um dos seis *Millennium Prize Problems*, pelo qual é oferecido um milhão de dólares pelo (The Clay Mathematics Institute, 2017) de Cambridge.

O TSP pretende encontrar o caminho mais curto visitando todos os pontos escolhidos, terminando o percurso no ponto de partida. Uma das estratégias de resolução, seguida pelo *Greedy Algorithm*, é escolher sempre o seguinte ponto mais próximo. No entanto, a aplicação deste algoritmo dá normalmente origem a percursos que se cruzam, levando à necessidade de aplicar outras técnicas de otimização do percurso.

Uma generalização do TSP é o *Vehicle Routing Problem* (VRP) (Braekers, Kris; Ramaekers, Katrien; Nieuwenhuyse, 2016)onde são aplicadas técnicas heurísticas de otimização combinada, como algoritmos genéticos, *simulated annealing*, pesquisa tabu, formação dinâmica de rios ou otimização da colónia de formigas.

O investigador de Inteligência Artificial Marco Dorigo descreveu em 1993 um método de gerar heuristicamente "boas soluções"para o TSP usando uma simulação de uma colónia de formigas chamada *Ant Colony System* (ACS). Ele modelou o comportamento observado em formigas reais para encontrar caminhos curtos entre as fontes de alimento e seu ninho (Dorigo e Gambardella, 1997).

O modelo ACS é uma das técnicas preferidas para obter percurso satisfatórios, sendo um dos mais usados.

Abordagem normalmente adotada é constituída por 2 etapas:

- Otimização de agrupamento;
- Otimização de percursos.

Um exemplo dessa abordagem em duas etapas está descrito no artigo *Located Multiple Depots and Vehicles Routing with Capacity Problem* (Monirian, Vaziri e Vaziri, 2017).

Uma nova abordagem ao problema utilizando uma só etapa encontra-se descrita no artigo *Multi-Depot Vehicle Routing Problem: A One-Stage Approach* (Lim e Wang, 2005). Em contraste com a abordagem em duas etapas, a nova abordagem integra o agrupamento com a definição do percurso identificando dois tipos de percursos: percurso de rascunho e percurso de detalhes. Os resultados experimentais mostram que este novo algoritmo, de uma só etapa, supera o método das duas etapas.

3.4.1 Soluções Free and Open-Source Software (FOSS) para VRP

Existem diversas bibliotecas de software que, através da aplicação das técnicas matemáticas já abordadas, tentam obter bons resultados para o VRP. Seguindo a filosofia adotada pelo Smart Depot, pretende-se disponibilizar um serviço de planeamento otimizado das rotas baseado em bibliotecas ou mesmo serviços FOSS.

jsprit O jsprit é uma biblioteca open source, escrita em Java e com o licenciamento *GNU Lesser General Public License* (LGPL), para resolver o TSP e o VRP. Anuncia-se como flexível, simples de utilizar e com baixos requisitos de computação (Jsprit, 2017);

OptaPlanner O OptaPlanner é open source, escrito em Java, disponibilizando um motor de planeamento de rotas embebido que usa otimização heurística com algoritmos como Pesquisa Tabu, *Simulated Annealing* e *Late Acceptance* com bons resultados. O desenvolvimento desta ferramenta é patrocinado pela Red Hat (Red Hat, 2017);

SYMPHONY VRP Esta aplicação, também open source, compromete-se em resolver o problema VRP com limitações heurísticas de capacidade e o TSP (Ted Ralphs, 2017).

Tabela 3.2: Bibliotecas FOSS de planeamento otimizado de rotas.

Projeto	Licença	URL
Jsprit	LGPL	jsprit.github.io
OptaPlanner	Apache License	optaplanner.org
SYMPHONY VRP	Common Public License	projects.coin-or.org/SYMPHONY

3.4.2 Soluções Comerciais para VRP

É importante referir que esta área de estudo é comercialmente muito atrativa para grandes organizações como a Google ou a Red Hat que disponibilizam serviços pagos para o planeamento de rotas.

Devido à complexidade do problema, estes serviços limitam o número de pontos de destino, como se pode ver na Tabela 3.2.

Apesar de não serem serviços completamente gratuitos, permitem a utilização de um número limitado diário de pedidos de optimização sem custos, como é o caso do Google API. O acesso gratuito a este serviço, apesar de limitado, foi um fator importante durante o desenvolvimento do SGC já que permitiu testar a eficiência do serviço e aferir também qual o nível de dificuldade de integração do mesmo. A utilização da componente paga deste serviço tornou-se uma opção válida para a fase de produção.

Tabela 3.3: Serviços comerciais de planeamento otimizado de rotas.

Projeto	**Limite de Pontos**	**URL**
Google Maps Directions API	23	developers.google.com/maps/
graphhopper	150	graphhopper.com

Capítulo 4

Avaliação das Soluções e Tecnologias Existentes

Neste capítulo são descritas as avaliações feitas as abordagens e as tecnologias existentes, identificadas no Capítulo 3.

4.1 Avaliação das Abordagens Existentes

Relativamente a soluções existentes com algumas semelhanças com o Smart Depot, verifica-se que uma delas, a Green box, ainda não chegou ao mercado comercial e aparenta estar estagnada desde 2012. A outra solução, a aplicação Android, não é propriamente uma solução do problema, mas mais outra forma do produtor do resíduo contactar a empresa de recolha. Esta última aborda o problema (apenas dos OU) disponibilizando uma aplicação móvel, que necessita de ligação à Internet, para fazer pouco mais do que já era feito com uma chamada telefónica. Esta solução continua a necessitar da intervenção humana, continuando a empresa de recolha dependente da iniciativa do produtor do resíduo, que pode nunca vir a acontecer, seja pela aplicação móvel ou através da chamada telefónica.

A abordagem do projeto académico Green box, apesar de concentrado numa função de recolha de OAU vocacionada para as canalizações, tem algumas semelhanças conceituais com o Smart Depot, já que também identificaram a necessidade em ter um sensor remoto que enviasse dados para uma unidade centralizada. O facto deste projeto, Green box, ter sido desenvolvido antes de 2012 tem um impacto muito significativo no que toca à tecnologia usada no sensor remoto. A área da computação ubíqua, especialmente no que toca ao hardware, apresenta diferenças muito significativas neste espaço temporal de cinco anos (2012-2017).

Por existir esta discrepância tecnológica de cinco anos e pelo facto de ser um projeto focado numa caixa de separação de óleos, o Green box não contribui como abordagem modelo para inspirar o *design* do Smart Depot.

4.2 Avaliação das Tecnologias Existentes

As tecnologias relevantes para o *design* do projeto dividem-se em dois grupos:

- Hardware - MCU com módulo de comunicações e do sensor de nível de fluido;
- Software - Planeamento otimizado de rotas.

4.2.1 Escolha do Hardware

O primeiro ponto a ter em conta na escolha do MCU relaciona-se com o seu custo, já que é, em princípio, o componente mais caro do SMR. Outro ponto que é pertinente na escolha do MCU é o seu consumo energético.

No que toca ao módulo de comunicações, a escolha prende-se mais ao ambiente onde o SMR irá operar. Os SMR que estiverem em locais com acesso à Internet via um *Access Point* (AP) necessitarão de um módulo WiFi, os que não estejam nessas condições terão de ser equipados com outra tecnologia de comunicação ajustada a esse ambiente.

Relativamente aos sensores de nível de fluido o Smart Depot poderá optar por sensores de diferentes tecnologias. Poderão ser usadas técnicas de leituras diretas ou indiretas (calculadas através de conversões matemáticas).

Para selecionar o sensor a ser usado, serão comparados quanto a sua exatidão, precisão e resolução.

A resolução é obtida pelas caraterísticas técnicas de cada sensor e a precisão, em princípio, poderá ser corrigida matematicamente, via uma fórmula de *offset* aplicada ao valor obtido, desde que exista linearidade ao longo de toda a gama na escala de valores do sensor.

Após a redução do tipo de sensores utilizando as grandezas anteriormente mencionadas, resta avaliar os restantes quanto à exatidão dos valores lidos por cada um deles. Para isto será necessário efetuar testes de hipóteses.

De referir que, para sensores do mesmo tipo, a comparação de grandezas como a exatidão ou resolução pode ser feita através da consulta das fichas técnicas dos sensores.

Metodologia de Avaliação dos Sensores

Utilizando as ferramentas da inferência estatística, serão feitos testes de hipóteses de forma a decidir que tipo de sensores utilizar nas soluções propostas pelo Smart Depot.

Nestes testes são propostas hipóteses e são usadas ferramentas estatísticas para tirar conclusões sobre essas hipóteses, minimizando os erros.

O que é pretendido é verificar se pode ser rejeitada a hipótese nula (H0) para ser possível inferir a hipótese H1.

É sempre com base em H0 que se pode concluir algo. Será usado o erro de primeira espécie que tenta rejeitar H0, partindo do princípio que H0 é verdadeiro.

Em seguida descrevem-se os passos necessários para aplicar os testes de hipóteses:

1. Começa-se por delimitar as hipóteses, H0 e H1, e o tipo de teste (bilateral, unilateral dir ou esq.);
2. Identifica-se qual o teste estatístico adequado (emparelhados ou independentes, paramétricos ou não paramétricos);
3. Calcula-se a "Estatística de Teste";
4. Compara-se a "Estatística de Teste"calculada com os limites das regiões de rejeição (região crítica).

Rejeita-se H0 sempre que a “Estatística de Teste” calculada estiver na região crítica. Se a "Estatística de Teste"calculada estiver na zona de confiança, não será rejeitada H0.

Serão testados 3 sensores de tecnologias diferentes.

- Sensor de Ultrassons;
- Sensor de Efeito Hall;
- Sensor de Carga por tensão.

Os sensores de nível de fluido farão 30 leituras a vários níveis de fluido, utilizando os mesmos valores de controlo em mililitros, utilizando saltos de 100 ml, já que a resolução da pipeta de controlo é também de 100 ml.

Tendo as amostras 30 leituras pode-se utilizar testes paramétricos, já que é possível assumir que seguem uma distribuição normal (4.1).

$$X \sim \mathcal{N}(\mu, \sigma^2) \tag{4.1}$$

Grandezas a Avaliar

Com o objetivo de selecionar o tipo de sensor de nível de fluido, serão efetuados testes com quantidades de líquido de controlo, utilizando uma resolução de 100 ml e 30 amostras por sensor testado.

Com base no valor lido pelo sensor e no valor de controlo será calculado o módulo da diferença dos dois valores. Assim, serão comparadas as médias deste cálculo para cada sensor, pretendendo-se saber qual tem a média mais próxima de zero e assim se aproxima mais do valor de controlo/real.

Desta forma será escolhido o sensor que mais evidenciar exatidão dos valores lidos, sendo as leituras o mais próximas possíveis do valor real.

Hipóteses a Testar

A hipótese nula (4.2) que se pretende rejeitar é a de que os três (ou mais) sensores têm valores de média iguais com um erro inferior a $\alpha \leq 5\%$.

$$h0 : \mu 1 = \mu 2 = \mu 3 \tag{4.2}$$

Caso se verifique que há diferenças entre as leituras dos sensores, serão comparados dois a dois para selecionar o que mais se aproxima das leituras dos valores de controlo.

$$h1 : \mu 1 < \mu 2 \tag{4.3}$$

$$h2 : \mu 1 < \mu 3 \tag{4.4}$$

$$h3 : \mu 2 < \mu 3 \tag{4.5}$$

Para testar os três (ou mais) grupos de sensores de nível de fluido utiliza-se o teste estatístico de *Friedman* (não paramétrico), no entanto, como as amostras utilizadas são de 30 leituras, pode-se utilizar o teste Análise de Variância (ANOVA) (paramétrico). Estes dois tipos de teste estatístico são usados porque os dados da amostra são emparelhados, ou seja, aplicados sobre as mesmas condições. Do grupo de testes paramétricos aplicados a amostras emparelhadas pode-se ainda escolher entre o teste ANOVA e o teste T-Student, sendo que o este último apenas permite comparar amostras duas a duas.

Resultados

Apresenta-se em seguida a Tabela 4.1 com os dados recolhidos no teste aos sensores. O sensor de Efeito Hall foi eliminado do teste por se tratar de um tipo de sensor de medição de nível de ponto o que o colocava em situação de desvantagem relativamente aos outros sensores de nível contínuo, de Ultrassons e de Carga por tensão.

De referir que as colunas "lido"referem-se à mediana de 5 leituras para cada valor de referência, de forma a eliminar outliers, que no caso do sensor de carga por tensão nunca foram detetadas.

Ao analisar os resultados pode-se rejeitar a hipótese nula (4.2) já que existem diferenças nas leituras obtidas pelos diferentes sensores. Identificou-se que o sensor de carga por tensão consegue obter leituras com maior exatidão que o sensor de Ultrassons e que não sofre de problemas com *outliers*.

4.2.2 Escolha do Software

Para a escolha do serviço ou biblioteca a utilizar no cálculo das rotas será necessário efetuar testes de integração com o sistema de informação que se pretende desenvolver. Só após o desenvolvimento do sistema principal, que permite resolver o problema aqui tratado, é que serão feitos esses testes. A extensão deste projeto poderá levar a que o serviço (secundário ao problema) do planeamento otimizado das rotas seja delegado para trabalho futuro.

Para avaliar estes serviços e bibliotecas, para além do nível de documentação disponível e dos requisitos mínimos de qualidade dos resultados, o fator que aparenta vir a ser diferenciador será o tempo de execução necessário para o cálculo otimizado das rotas.

O código desenvolvido para o *firmware* do SMR pode ser escrito em duas linguagens, Lua e C++. Optou-se usar a linguagem C++ por esta ser compilada, gerando diretamente a imagem de *firmware* a ser enviada para o MCU, permitindo ter uma imagem de menor tamanho e com execução mais rápida. Caso o código fosse escrito em Lua, seria criado um *script* que era chamado por um interpretador de Lua incluído num *firmware* genérico, que se pode obter *online* (Frightanic.com, 2015), onde também são incluídas as bibliotecas usadas por esse *script*.

Para a aplicação móvel de apoio à configuração do SMR optou-se por desenvolver uma aplicação Android nativa, utilizando a linguagem Java para Android, por esta permitir o acesso e controlo direto aos módulos GPS e Wi-Fi dos dispositivos. O projeto desta aplicação foi criado utilizando o *Integrated Development Environment* (IDE) Android Studio disponibilizado pela Google.

Tabela 4.1: Teste de sensores

Teste sensores	Ultrassons			Carga por tensão		
valor ref. (ml)	lido	% sucesso	desvio	lido	% sucesso	desvio
100	87	87,00	13	101	99,00	1
200	189	94,50	11	200	100,00	0
300	291	97,00	9	300	100,00	0
400	394	98,50	6	400	100,00	0
500	493	98,60	7	501	99,80	1
600	612	98,00	12	600	100,00	0
700	694	99,14	6	700	100,00	0
800	796	99,50	4	800	100,00	0
900	911	98,78	11	900	100,00	0
1000	1009	99,10	9	1001	99,90	1
1100	1101	99,91	1	1100	100,00	0
1200	1200	100,00	0	1200	100,00	0
1300	1311	99,15	11	1300	100,00	0
1400	1431	97,79	31	1402	99,86	2
1500	1539	97,40	39	1500	100,00	0
1600	1634	97,88	34	1600	100,00	0
1700	1787	94,88	87	1700	100,00	0
1800	1893	94,83	93	1800	100,00	0
1900	1977	95,95	77	1899	99,95	1
2000	2003	99,85	3	2000	100,00	0
2100	2091	99,57	9	2100	100,00	0
2200	2179	99,05	21	2200	100,00	0
2300	2307	99,70	7	2300	100,00	0
2400	2401	99,96	1	2400	100,00	0
2500	2468	98,72	32	2500	100,00	0
2600	2595	99,81	5	2602	99,92	2
2700	2689	99,59	11	2700	100,00	0
2800	2767	98,82	33	2800	100,00	0
2900	2825	97,41	75	2900	100,00	0
3000	2945	98,17	55	3000	100,00	0
3100	3099	99,97	1	3101	99,97	1
3200	3170	99,06	30	3200	100,00	0
3300	3182	96,42	118	3300	100,00	0
3400	3279	96,44	121	3397	99,91	3
3500	3360	96,00	140	3500	100,00	0
3600	3649	98,64	49	3600	100,00	0
3700	3539	95,65	161	3700	100,00	0
3800	3823	99,39	23	3800	100,00	0
3900	3894	99,85	6	3900	100,00	0
4000	3945	98,63	55	4001	99,98	1

No desenvolvimento dos *web services* da API *Representational State Transfer* (REST) do SGC escolheu-se usar a linguagem Python por ser possível programar nesta utilizando o paradigma *Object-Oriented* (OO) e também por existir um *micro framework*, Flask, que agiliza o desenvolvimento de aplicações web e *web services*. Outro fator que contribui-o para a escolha do Python no desenvolvimento do SGC foi a existência de grandes projetos *web* que usam esta linguagem, das quais se destacam os seguintes:

- Youtube;
- DropBox;
- Survey Monkey;
- Google;
- Reddit;
- Yahoo Maps.

Capítulo 5

Design

Este capítulo descreve o design da solução proposta, identificação de requisitos, detalhes da arquitetura e modelo de dados, finalizando com a comparação e explicação das escolhas efetuadas.

5.1 Design da Solução

Para atingir os objetivos deste projeto, pretende-se desenvolver SMR constituídos por um microcontrolador, módulo de comunicações e sensores de leitura do nível de fluido. Estes SMR serão a fonte de dados do sistema de informação centralizado, também a desenvolver, baseado em *Web Services* disponibilizados através de uma API REST.

Para testar e avaliar os SMR, serão construídos dois protótipos. Estes protótipos serão equipados com sensores de nível de fluidos distintos, com um sensor de potência da bateria, com módulos de comunicação por Wi-Fi e GSM.

Além da escolha dos componentes a usar no SMR e sua construção, os dois protótipos necessitam ser programados para desempenhar as tarefas pretendidas. Alguns exemplos de tarefas já previstas são:

- Medição do nível de óleo existente no depósito;
- Monitorização do estado das baterias que alimentam o sensor;
- Interface para configuração da ligação do SMR a um Access Point (AP) Wi-Fi ou cartão GSM;
- Envio dos dados monitorados para um *Web Service*.

Os *Web Services* a desenvolver serão o núcleo centralizador do processamento dos dados recolhidos e gerador da informação, a ser prestada a clientes e gestores de negócio. O desenvolvimento de um SGC irá disponibilizar serviços através de uma API REST.

Na figura 5.1 foi utilizado um diagrama de classes na notação *Unified Modeling Language* (UML) para representar o modelo de domínio do projeto Smart Depot. Neste modelo pode-se visualizar como se enquadrarão os SMR e o SGC do Smart Depot com os elementos que integram o processo de recolha de óleos.

Das relações representadas no diagrama de classes, no âmbito deste projeto, destacam-se as que envolvem as entidades SMR e SGC.

Podem ser construídos diferentes SMR mediante o tipo de sensor usado para monitorar o nível de fluido dos depósitos. Apesar de utilizarem diferentes tipos de sensores, todos eles

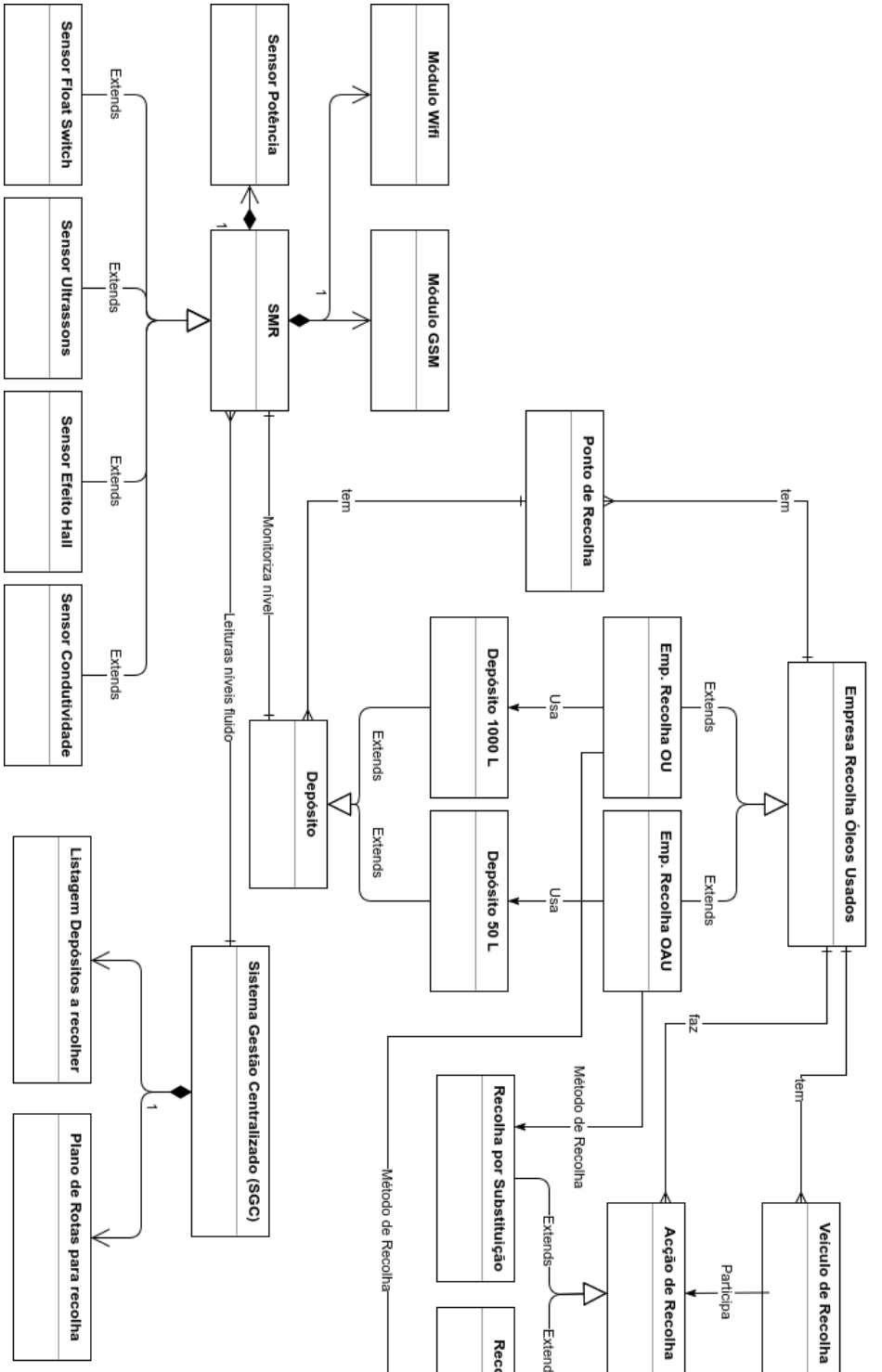


Figura 5.1: Modelo de Domínio em UML do projeto Smart Depot

partilham um conjunto de caraterísticas comuns para garantir a comunicação, por Wi-Fi ou GSM, com o SGC e para a leitura do estado da sua bateria interna. Os dados recolhidos pelo SMR são enviados para o SGC utilizando o API REST desenvolvido. Este API, para além da persistência dos dados recebidos, também permite que as empresas de recolha obtenham a listagem de depósitos a recolher e o plano de rotas.

5.2 Requisitos

Apresenta-se em seguida os requisitos, funcionais e não funcionais, do SMR e do SGC.

5.2.1 SMR

A solução idealizada apresenta vários desafios que terão de ser ultrapassados. O SMR terá de ser equipado com um sensor capaz de ler o nível de fluido presente no depósito. Descrevem-se em seguida os requisitos do SMR.

Requisitos Funcionais do SMR

1. Ler nível de fluido - O SMR necessita ser capaz de ler o nível de fluido existente no depósito de óleo usado para que seja possível enviar essa informação ao SGC;
2. Ler estado da bateria - O SMR necessita ser capaz de ler o estado da sua bateria para que seja possível enviar essa informação ao SGC;
3. Modo AP para configuração - O SMR necessita ter a capacidade de disponibilizar um AP, quando iniciado em modo de configuração, para permitir que um utilizador se consiga ligar a ele e lhe passe a configuração pretendida (entre outros parâmetros, um exemplo de configuração necessária são o *Service Set Identifier* (SSID) e a senha do AP que o SMR deve usar para aceder à Internet);
4. Identificar redes Wi-Fi próximas - O SMR necessita ser capaz de captar os AP Wi-Fi próximos para que seja possível disponibilizar essa informação ao utilizador que o irá configurar, a fim deste poder escolher uma rede Wi-Fi visível pelo SMR;
5. Envio dos dados lidos por Wi-Fi - O SMR necessita ser capaz de estabelecer uma conexão com um AP Wi-Fi com acesso à Internet para conseguir enviar os dados recolhidos ao SGC;
6. Envio dos dados lidos por GSM (SMS ou GPRS) - O SMR necessita ter uma alternativa à comunicação por Wi-Fi, utilizando por exemplo GSM, para conseguir comunicar com o SGC.

Requisitos Não Funcionais do SMR

1. Portabilidade - O SMR precisa ser portátil, sendo suficientemente leve e energeticamente autónomo, para ser possível equipá-lo em diferentes depósitos, localizados em diferentes locais e condições;

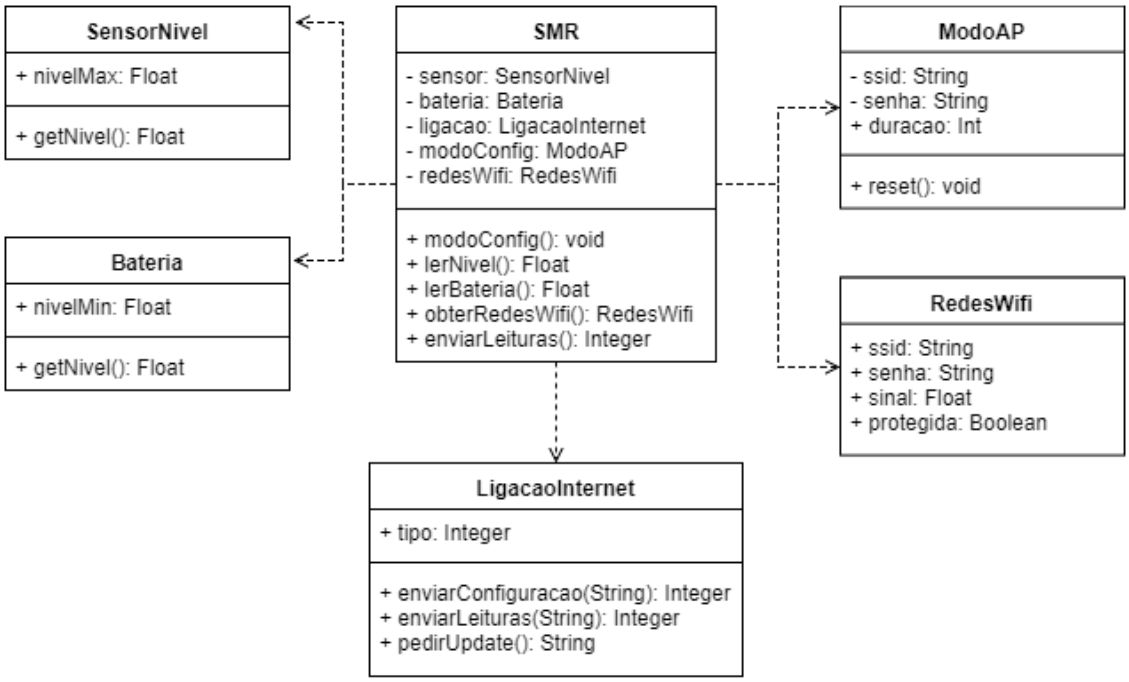


Figura 5.2: Diagrama de Classes do SMR em notação UML.

2. Fiabilidade - O SMR necessita ser capaz de ler o nível de fluido com exatidão com uma resolução mínima de 100 ml, escolhendo os sensores que melhor se ajustem e implementando mecanismos de eliminação de outliers, para obter leituras o mais próximo do real possível;
3. Mobilidade - O SMR precisa permitir a sua configuração de forma a poder-se associa-lo a diferentes coordenadas GPS, para ser possível utilizá-lo em diferentes locais de recolha.

Diagrama de Classes do SMR

Após se ter analisado os requisitos, funcionais e não funcionais, do SMR foi desenhado o diagrama de classes apresentado na figura 5.2, utilizando a notação UML. Trata-se de um diagrama de classes conceptual em que são elencadas as classes e indicadas as dependências entre si.

Diagrama de Sequência do SMR

Para representar de forma simples o comportamento descrito pelas user stories foram utilizados diagramas de sequência, em notação UML. Na figura 5.3 apresenta-se, como exemplo, um desses diagramas de sequência.

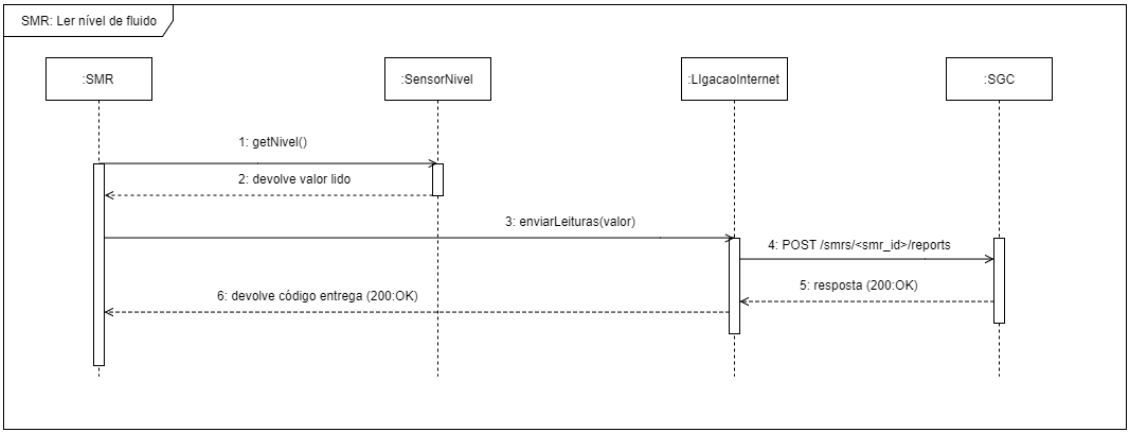


Figura 5.3: Diagrama de Sequência do SMR em notação UML.

5.2.2 SGC

O SGC desenvolvido no Smart Depot será baseado em *Web Services* que estarão disponíveis através de uma API REST. Descrevem-se em seguida os requisitos do SGC.

Requisitos Funcionais do SGC

1. Receber dados dos sensores - O SGC necessita ser capaz de receber mensagens *JavaScript Object Notation* (JSON) em protocolo HTTP, utilizado o métodos POST, para recolher os dados lidos e enviados pelo SMR;
2. Enviar notificações - O SGC necessita ser capaz de enviar mensagens de texto, utilizado um serviço de Email, para conseguir notificar os clientes de situações como; substituição de baterias ou necessidade de renovação da anuidade para acesso aos serviços;
3. Propor rotas - O SGC necessita ser capaz de calcular ou obter uma sequência otimizada de passagens por pontos de recolha, utilizando as suas coordenadas *Global Positioning System* (GPS), para fornecer rotas de recolha otimizadas aos seus clientes.

Requisitos Não Funcionais do SGC

1. Interoperabilidade - O SGC tem que garantir uma interface aplicacional e respetiva documentação de forma a permitir que diferentes clientes consigam comunicar com os seus serviços;
2. Segurança - O SGC tem que implementar mecanismos de segurança para proteger as comunicações e os dados guardados na sua base de dados;
3. Performance - O SGC tem que correm em servidores com capacidade de processamento suficiente para garantir que os seus serviços respondem aos pedidos dos clientes em tempo útil;
4. Disponibilidade - O SGC tem que correm em servidores com *High-Availability* (HA) para garantir o acesso permanente aos seus serviços;

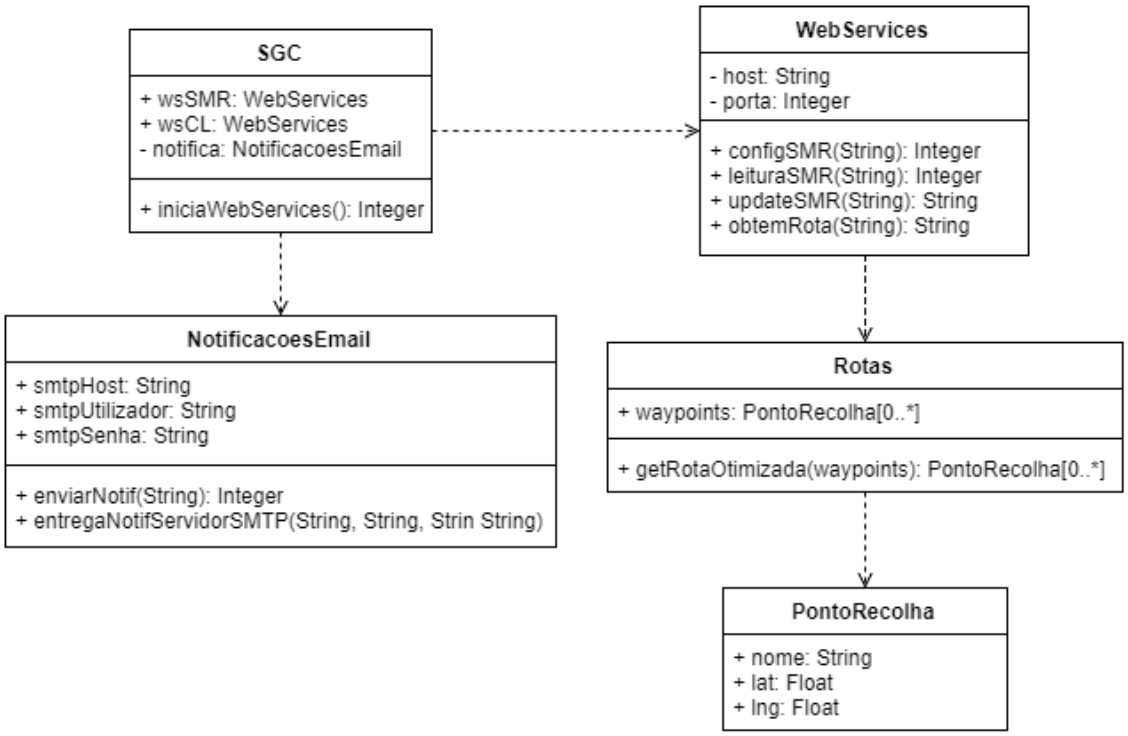


Figura 5.4: Diagrama de Classes do SGC em notação UML.

5. Éticos e legais (confidencialidade dos dados) - O SGC tem que implementar mecanismos de controlo de acesso aos dados e de anonimização por motivos de ética profissional e para respeitar as leis de confidencialidade em vigor.

Diagrama de Classes do SGC

Tal como foi feito quando da análise dos requisitos do SMR, também para o SGC foi desenhado o diagrama de classes apresentado na figura 5.4, utilizando a notação UML.

Diagrama de sequência do SGC

Na figura 5.5 apresenta-se um diagramas de sequência, em notação UML, utilizados na representação detalhada de um dos processos do SGC.

5.3 Design Arquitetural

O desenho da arquitetura deve seguir as boas práticas da engenharia e carece de reflexão atenta, já que caso sejam necessárias alterações futuras, estas terão um elevado impacto em todo o funcionamento do projeto.

A arquitetura utilizada no desenvolvimento do *software* para o SMR, habitualmente chamado de "firmware", terá uma organização modular, seguindo o *module pattern*. Este programa terá o seu código segregado em diferentes blocos que contêm o código necessário para tratar

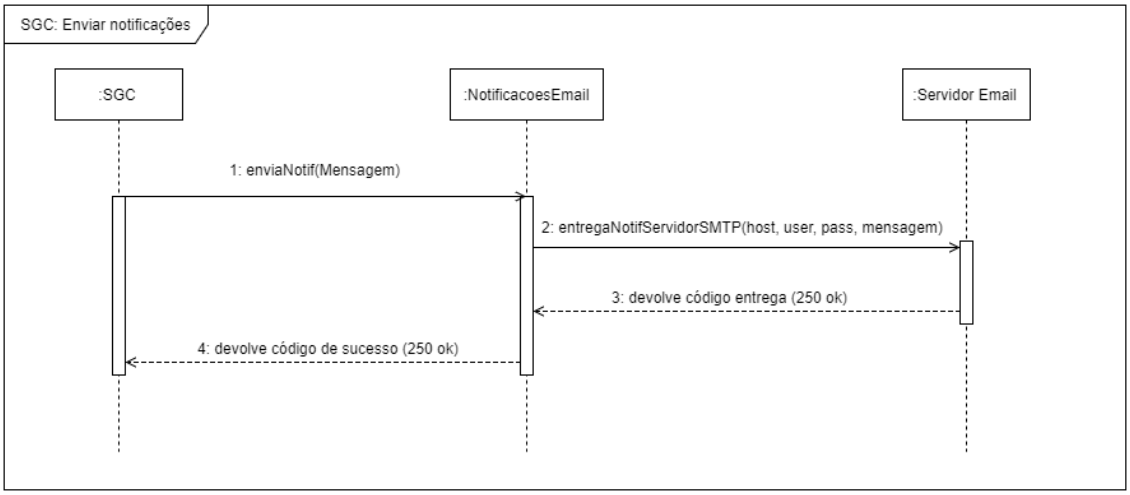


Figura 5.5: Diagrama de Sequência do SGC em notação UML.

cada um dos módulos. Destes módulos destacam-se o que trata da comunicação *Wi-Fi*, o da comunicação GSM, o módulo para o interface de configuração do SMR e o módulo de leitura de cada um dos tipos de sensores de nível de fluido usados. Na Figura 5.1 estão representados alguns destes módulos constituintes do *software* do SMR.

Para o desenvolvimento do SGC pretende-se utilizar uma topologia baseada numa API REST, apresentam-se duas propostas que se distinguem principalmente na utilização de um *Enterprise Service Bus* (ESB) e da ponderação de utilizar um sistema externo de autenticação *Central Authentication Service* (CAS) e outro de planeamento de rotas também externo ao SGC.

As figuras 5.6 e 5.7 utilizam a linguagem UML para representar de forma padronizada, através de diagramas de componentes, ambas as propostas.

Ambas as propostas apresentam vantagens e desvantagens que terão de ser alvo de reflexão para chegar à arquitetura a aplicar no projeto.

Apesar da necessidade de optar por uma das duas arquiteturas propostas, foi já feito um esboço dos processos de negócio necessários em ambas.

Como forma de descrever os passos já identificados de cada um dos processos de negócio e a sua interação com o cliente (empresas de recolha), foi desenhado um diagrama em *Business Process Model and Notation* (BPMN) que se apresenta na Figura 5.8.

Como é possível observar na Figura 5.8, pode-se identificar os três participantes do negócio:

- SMR;
- SGC;
- Programa do cliente.

O Programa do cliente foi representado neste diagrama de forma simplificada para facilitar o entendimento da sua interação com o SGC, já que este projeto não intervém no desenvolvimento interno dos processos deste participante.

No participante SMR estão descritas as atividades constituintes de dois eventos:

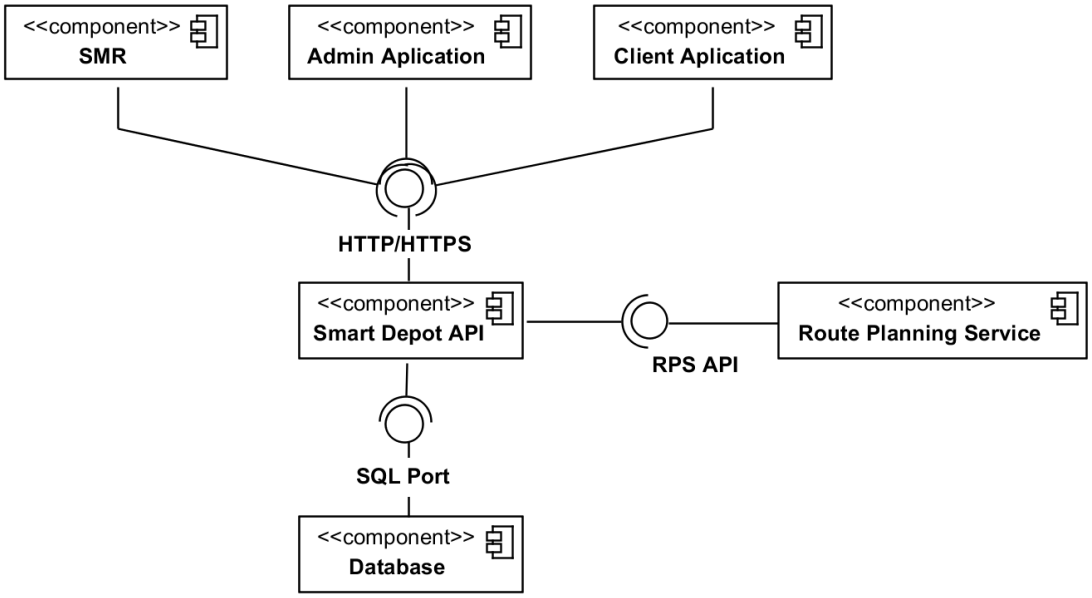


Figura 5.6: Diagrama de Componentes em UML do projeto Smart Depot

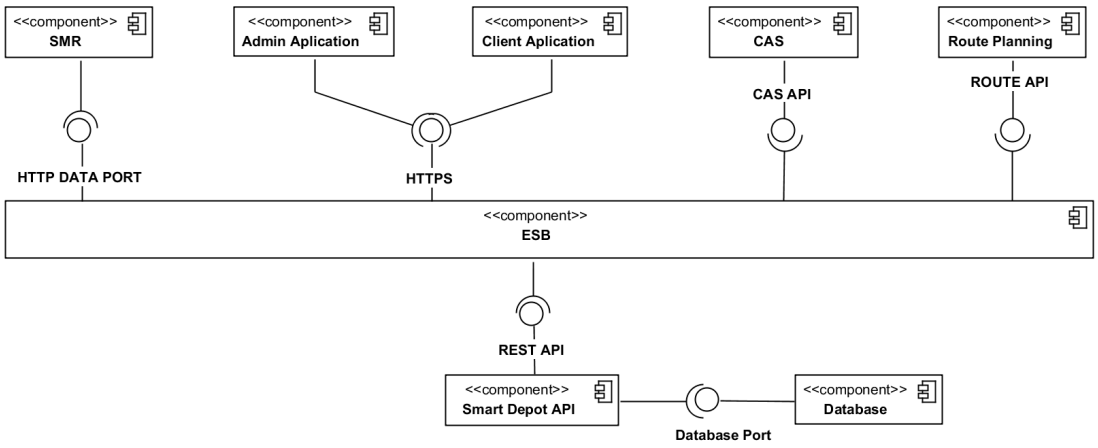


Figura 5.7: Diagrama de Componentes em UML do projeto Smart Depot com ESB

- Configurar a ligação ao AP;
- Ler e enviar dados ao SGC.

No participante SGC estão descritas as atividades constituintes de três eventos:

- Receber dados do SMR;
- Fornecer listagem dos pontos de recolha (prontos a ser visitados);
- Fornecer planeamento de rotas.

Nas atividades dos processos do SMR encontram-se detalhes como a validação da capacidade de ligação a um AP local, para determinar se é necessária a reconfiguração por parte do utilizador desta ligação.

Já nos processos do SGC pode-se observar atividades como a validação do SMR, o envio de um email à empresa de recolha, alertando para a necessidade de substituição das baterias do sensor ou a validação das credenciais da empresa de recolha e a verificação do estado do pagamento associada à utilização do serviço.

5.4 Escolha e Ajustes à Arquitetura

A escolha entre as duas propostas poderá depender da utilização de um sistema de autenticação *single sign-on* CAS externo. Para avaliar a utilização deste tipo de sistema devemos ter em conta, o custo do serviço, o esforço da implementação e ponderar sobre as questões de segurança e privacidade dos dados associadas.

Outro fator influenciador na escolha da arquitetura proposta é o planeamento otimizado das rotas ser feito ou não pelo SGC, utilizando uma das bibliotecas FOSS de cálculo do VRP ou consumida como serviço externo.

A solução idealizada apresenta vários desafios que terão de ser ultrapassados. O SMR terá de ser equipado com um sensor capaz de ler o nível de fluido presente no depósito. No entanto esse sensor terá que estar desvinculado fisicamente do depósito já que, após a análise do processo de recolha na empresa piloto, se verificou que o manuseamento, processo de transporte e limpeza do depósito seriam demasiado agressivos para utilizar um sensor incorporado no próprio depósito.

Tendo em conta as condicionantes do processo de recolha já enumeradas, foram eliminadas algumas das ideias de colocação do SMR e recuperada uma outra que se baseava na utilização de sensores de carga.

Com a utilização de sensores de carga seria possível construir um SMR semelhante a uma base de balança com capacidades de comunicação e notificação.

Com mais informações relacionadas com o processo de recolha em ambiente real, a escolha da arquitetura proposta na solução do problema sofreu alguns ajustes. Destacam-se os principais:

Módulo GSM no SMR Após se ter verificado que vários pontos de recolha (produtores do resíduo) acolhem mais do que um depósito e que, caso se equipasse cada um deles com um SMR com módulo GSM ter-se-ia que contratar, junto das empresas de telecomunicações, um cartão GSM por depósito. Assim, conclui-se que esta abordagem

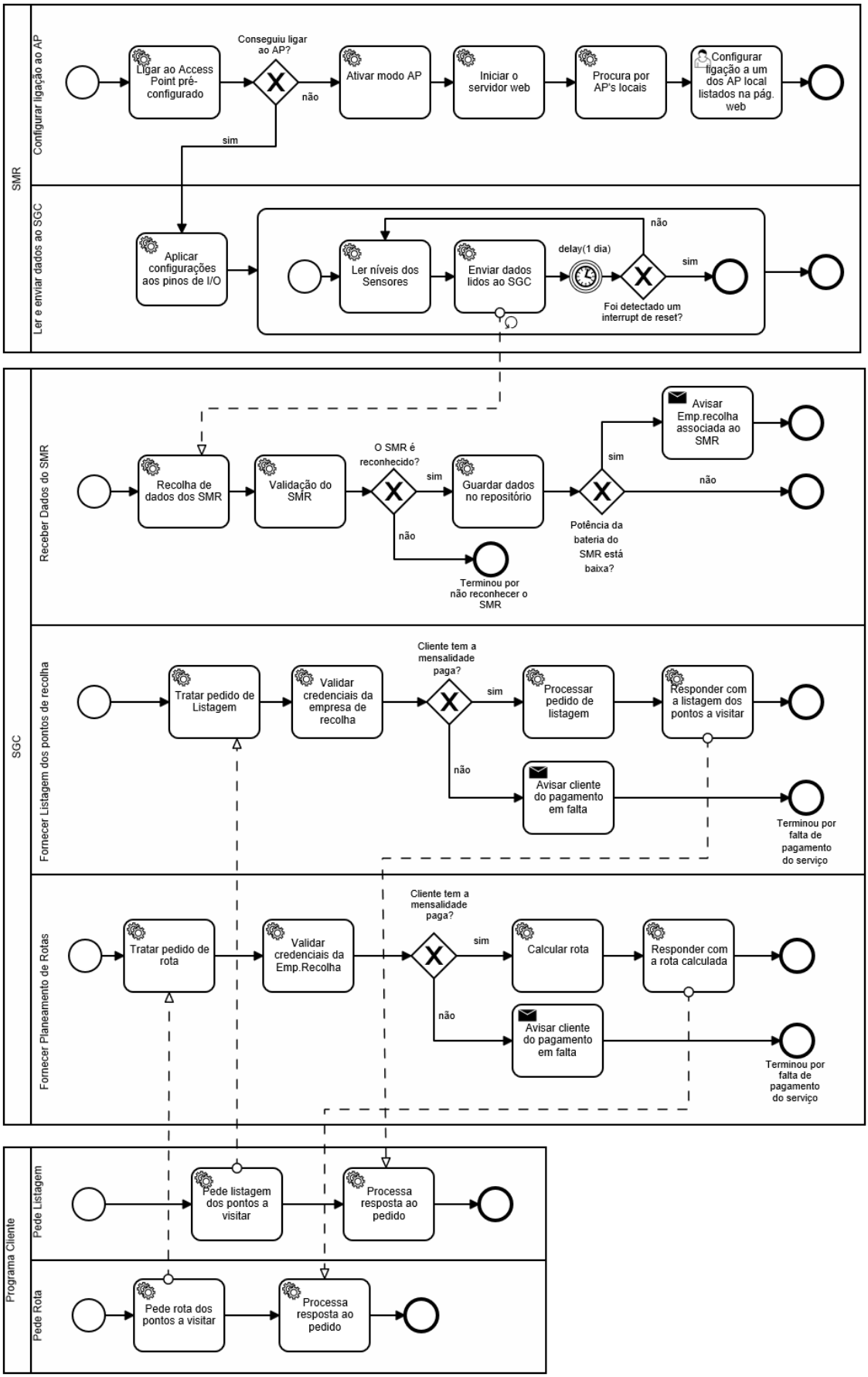


Figura 5.8: Diagrama dos processos de negócio do Smart Depot em BPMN.

não era economicamente viável. A solução, caso não exista um AP Wi-Fi disponível no ponto de recolha, implica que o cliente adquira apenas um cartão GSM e um modem *General Packet Radio Service* (GPRS) (3G ou 4G) passando assim a existir um AP Wi-Fi disponível para a utilização de todos os SMR aí instalados. Outra vantagem é não ser necessária a inclusão do módulo GSM no SMR, aumentando a sua complexidade e preço. Esta decisão fez com que se passasse a utilizar unicamente uma tecnologia de comunicação nos SMR, deixando de fazer sentido a execução dos testes que estavam previstos para selecionar o tipo de tecnologia de comunicação a utilizar.

Desvinculo do SMR do depósito Devido à forma como os depósitos necessitam ser manuseados, especialmente durante o seu processo de limpeza, onde ficam sujeitos a banhos com altas temperaturas e onde são utilizados agentes químicos de limpeza, foi necessário repensar a forma de obtenção das leituras de nível, tendo-se recuperado uma ideia inicial que baseava a sua inspiração numa base de balança digital. Isto fez com que fossem abandonados alguns tipos de sensores, como o por ultrassom, em detrimento de outros como os sensores de carga por medição de tensão.

Autenticação interna ao SGC Por questões de privacidade dos dados decidiu-se que a autenticação a utilizar seria controlada internamente pelo SGC.

Utilização da API Google para as rotas Após se ter verificado que a inclusão de bibliotecas FOSS de otimização de rotas ao SGC iria trazer uma elevada carga de processamento ao servidor do SGC, exigindo assim a aquisição de servidores mais caros, com mais memória e *Central Processing Unit* (CPU), decidiu-se ponderar a utilização de serviços externos das soluções comerciais. A constatação foi que a Google disponibiliza uma API, Google Maps Directions API, que mediante a entrega de uma coordenada GPS de origem, outra de destino e vários pontos (waypoints) de passagem obrigatória, ordenando-as de forma a otimizar a rota a ser percorrida.

De acrescentar que a Google Maps Directions API apresenta os seguintes limites de utilização:

1. 2.500 solicitações de rotas gratuitas por dia, calculadas como a soma das consultas do lado do cliente e do lado do servidor.
2. Até 23 pontos de referência em cada solicitação para consultas do lado do cliente ou do lado do servidor.
3. 50 solicitações por segundo, calculadas como a soma das consultas do lado do cliente e do lado do servidor.

5.5 Design da Base de Dados

Os dados recolhidos necessitam de ser guardados para consulta e tratamento. Tal será feito através da utilização de uma base de dados, que será escolhida de entre as FOSS, como foi feito com todas as outras tecnologias usadas neste projeto.

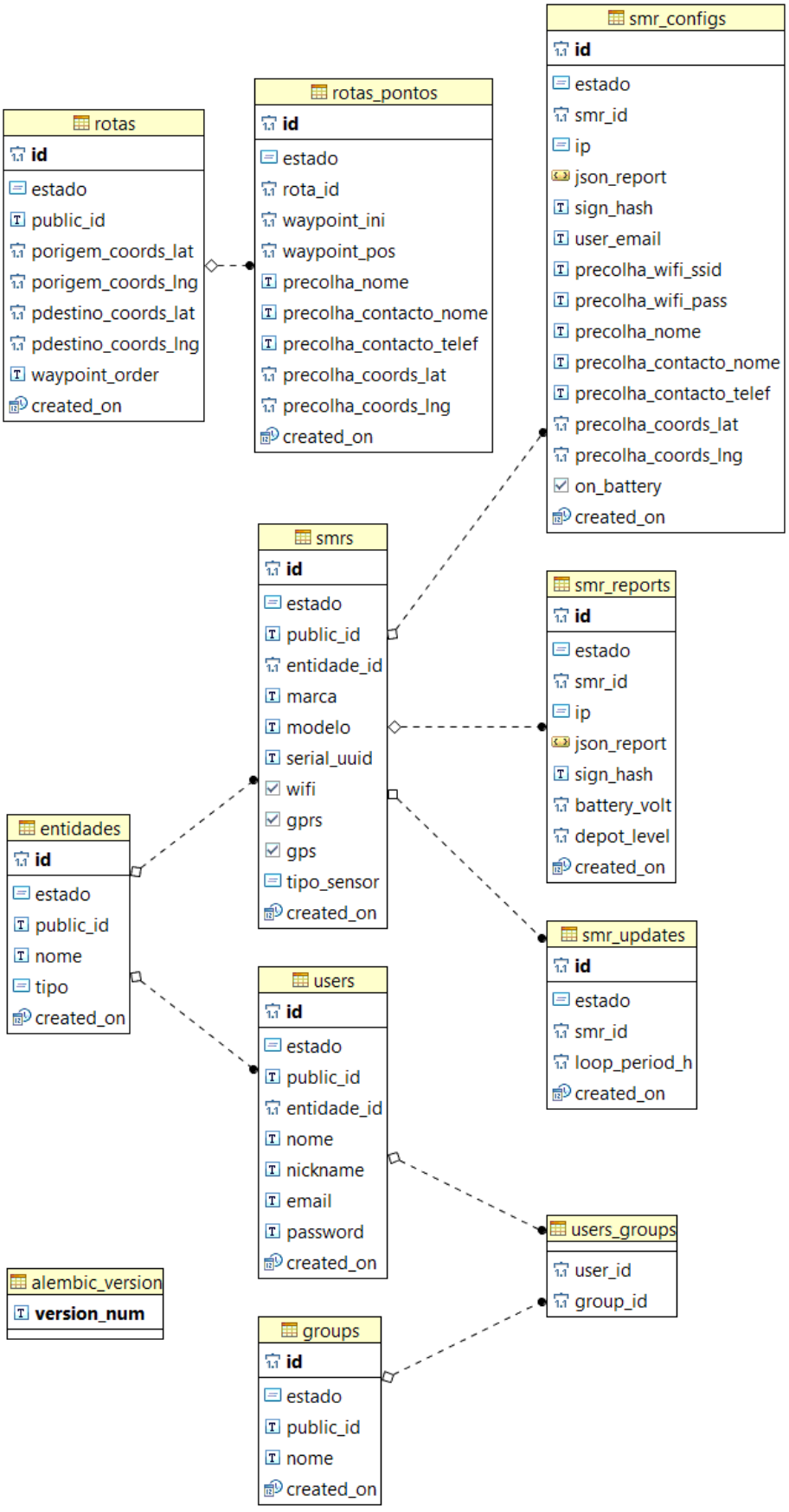
Para agilizar o acesso da linguagem de programação escolhida à base de dados, pretende-se utilizar um módulo que usa a técnica de *Object-Relational Mapping* (ORM). Esta técnica implementa padrões de persistência, como é o exemplo do *data mapper pattern* (Fowler e Martin, 2003). Este mapeamento entre a estrutura de dados e as classes, chamadas de

classes entidade na programação OO, faz com que as alterações sejam refletidas de ambos os lados, contribuindo para um rápido desenvolvimento.

A figura 5.9 apresenta o diagrama Entidade Relacionamento (ER) da base de dados utilizada pela API do SGC. Este modelo de dados permite dar suporte a várias funcionalidades:

- Gerir entidades, clientes ou administradores do SGC;
- Gerir utilizadores e grupos para autenticação e controlo de níveis de acesso;
- Gerir os equipamentos SMR, suas configurações, leituras e atualização remotas;
- Gerir a otimização de rotas e respetivos pontos de recolha.

Figura 5.9: Diagrama ER da base de dados SmartdepotDB.



Capítulo 6

Desenvolvimento e Implementação

Neste capítulo será documentado o trabalho elaborado durante o desenvolvimento e a implementação das três aplicações tratadas por este projeto; o SMR, o AMA e o SGC. Também serão aqui abordados detalhes relativos à gestão do projeto e sobre a preparação do servidor para o SGC, em especial, detalhes relacionados com a automatização da instalação das versões desenvolvidas e questões de segurança.

6.1 Gestão do Projeto

Para planear, agendar e documentar as tarefas a elaborar no decorrer do projeto foi utilizada uma ferramenta, da Zoho, de gestão de projetos online e gratuita.

Para além das tarefas criadas e datadas, foram também definidos *milestones*, apresentados na Figura 6.1, que definem pontos importantes que se pretende atingir no decorrer do projeto.

Na Figura 6.2 apresenta-se a página de entrada do gestor de projeto onde é possível observar um resumo do estado geral das tarefas e *milestones*, listagem de tarefas em aberto atribuídas ao utilizador autenticado e as alertas relativas a tarefas que estão a atingir a data de conclusão prevista.

Para utilizar esta ferramenta começou-se por definir as datas importantes para o projeto, data de início e entrega, assim como as datas para se atingirem os *milestones* intermédios. O passo seguinte passou por identificar o conjunto de tarefas a desenvolver para atingir os *milestones*. Esta organização do trabalho permitiu uma melhor estimativa do tempo a atribuir para a elaboração de cada tarefa, o que facilitou a definição de prioridades entre tarefas.

O registo da conclusão de tarefas e consequente atingir de *milestones* associados serviu, para além de organizar o projeto, como fator impulsionador do trabalho desenvolvido já que permitiu definir objetivos intermédios e uma melhor visão sobre o trabalho já concluído.

Com base nas datas de previsão e de conclusão das tarefas e milestones esta ferramenta gera diferentes tipos de relatórios e gráficos. O exemplo de um desses gráficos gerados automaticamente pela ferramenta com base na informação inserida é o diagrama de Gantt que se pode observar na Figura 6.3.

Para além do acesso web às várias funcionalidades de gestão do projeto também foi utilizada a versão móvel Android da Zoho que facilitou a notificação dos prazos, fecho de tarefas ou atribuição de novas tarefas a desenvolver.

Figura 6.1: *Milestones* (marcos intermédios) do projeto.

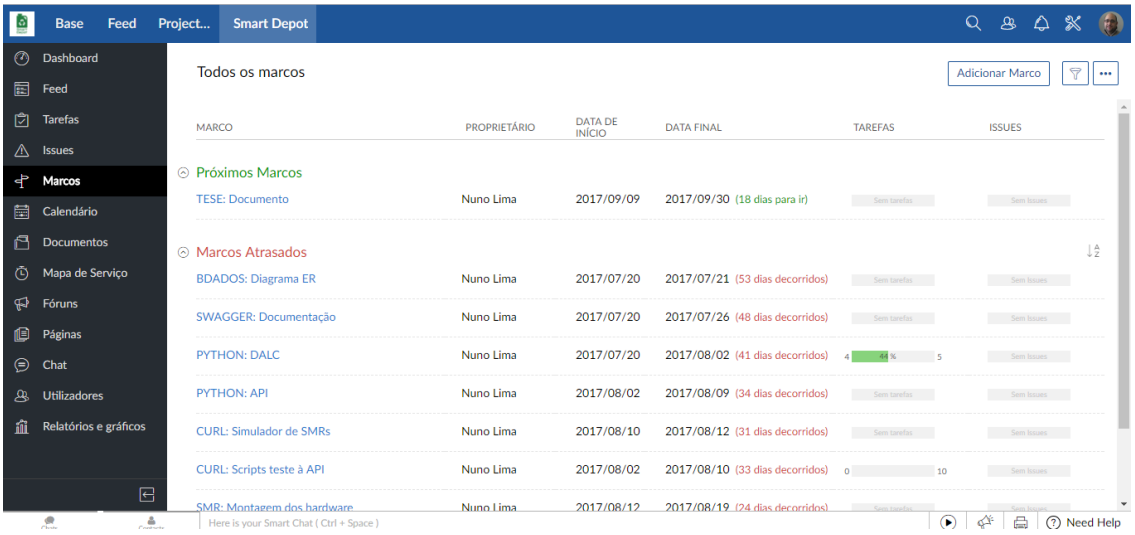


Figura 6.2: Programa gestor do projeto em zoho.com.

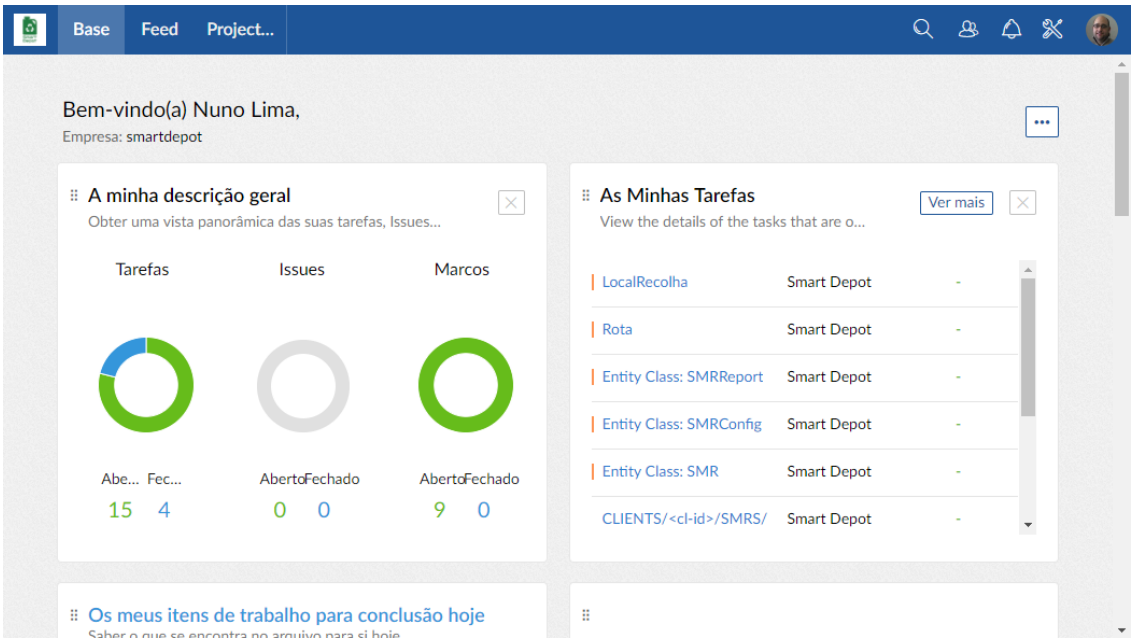
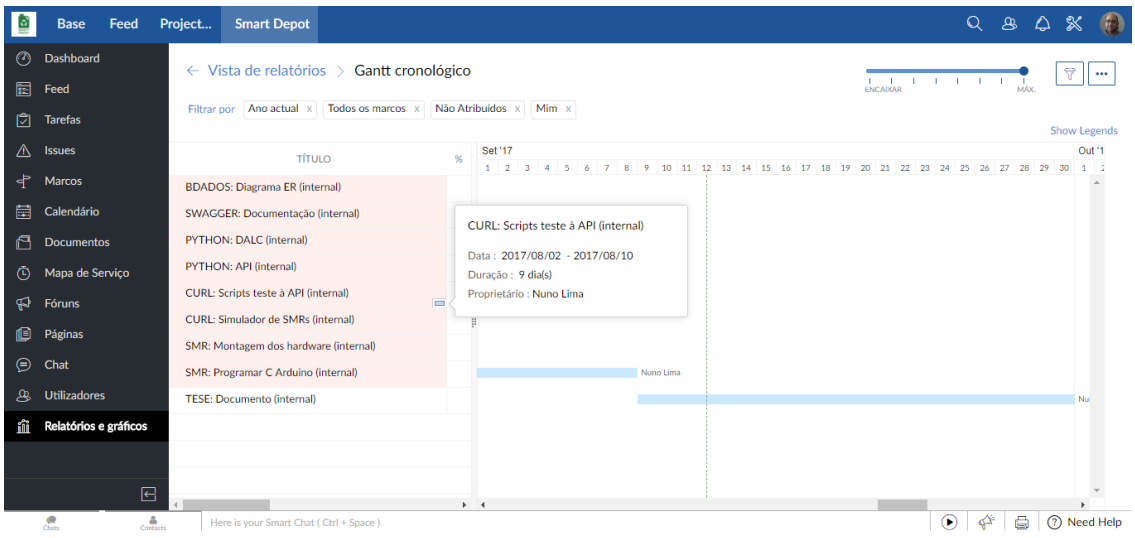


Figura 6.3: Diagrama de Gantt construído pelo gestor do projeto.



6.2 Protótipo do SMR

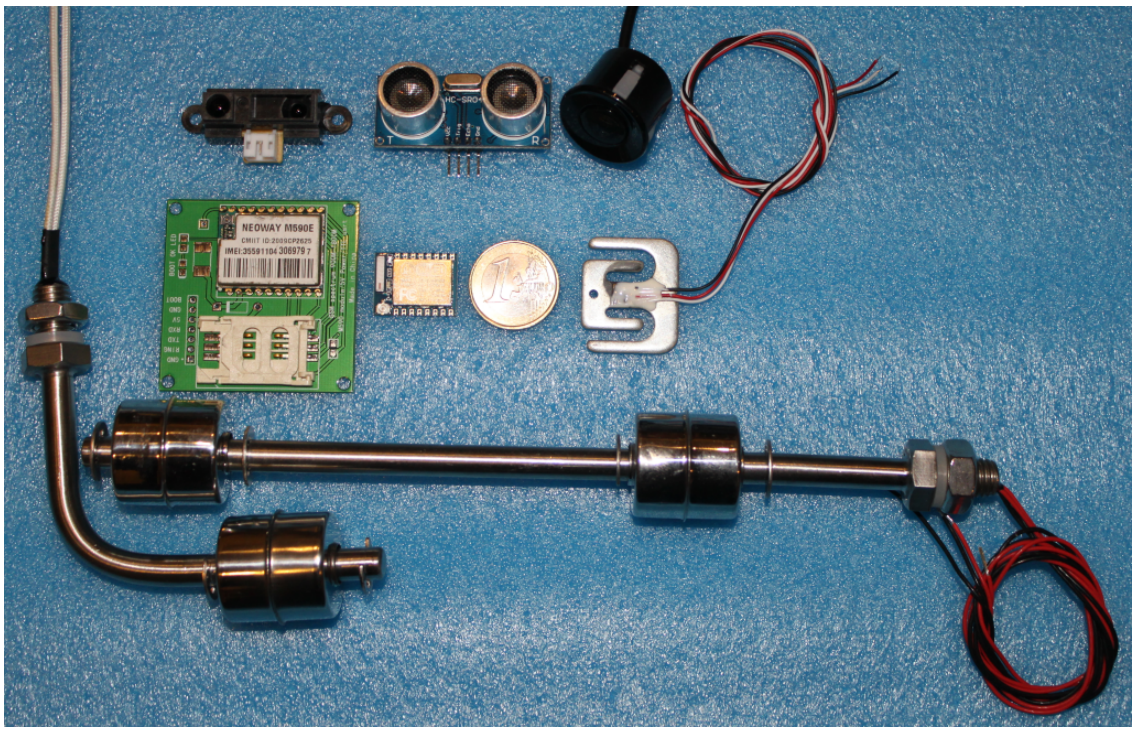
Na visita efetuada à empresa de recolha piloto foi possível analisar, em ambiente real, o processo de recolha do óleo usado que incluia a limpeza dos depósitos para posterior utilização. Todos os depósitos utilizados pela empresa tem as mesmas dimensões, formato, a capacidade de 50 litros e incluem tampa.

Verificou-se que o processo de limpeza dos depósitos recorria à utilização de químicos para a remoção das lamas depositadas no fundo dos mesmos e à utilização de água a elevadas temperaturas para a lavagem das tampas e dos próprios depósitos. Constatou-se que este processo de lavagem seria demasiado agressivo para permitir acoplar qualquer tipo de equipamento eletrónico ao depósito ou tampa. Este constrangimento funcional fez com que uma das principais soluções idealizadas, que incluiria um sensor de ultrassons na tampa do depósito, fosse descartada para ser recuperada uma outra solução que recorria à medição do peso do depósito. A ideia em utilizar uma balança para medir o nível de óleo existente em depósito tinha inicialmente perdido terreno para a utilização dos sensores de ultrassons, devido ao custo dos sensores de carga (*load cells*) ser superior. A repescagem da solução baseada nos sensores de carga deve-se a esta permitir a separação física entre o SMR e o depósito não interferindo com o processo de lavagem do depósito nem com a forma de transporte ou manipulação deste. O SMR passa a ser um equipamento independente do depósito e que pode ser utilizado com vários depósitos do mesmo tipo, sendo estes pousados em cima do SMR durante o período de recolha.

Com a introdução deste tipo de SMR o impacto no processo de recolha é minimizado já que os procedimentos existentes de manipulação, transporte e limpeza dos depósitos não são alterados, passando apenas a existir novos procedimentos para a manutenção do SMR. Outra vantagem é que o SMR tem uma periodicidade de manutenção muito menor que a aplicada aos depósitos.

A manutenção requerida pelo SMR é constituída pela limpeza regular da sua base superior, que serve para reter óleo vertido pelo depósito, pela substituição das baterias de alimentação e pela configuração dos dados de acesso ao AP Wi-Fi do ponto de recolha.

Figura 6.4: Componentes usados na construção do protótipo SMR.



6.2.1 Construção do Protótipo do SMR

Depois de se ter identificado que o SMR que melhor se adaptava ao processo de recolha era do tipo balança, utilizando sensores de carga por tensão, era necessário resolver o problema da comunicação com a Internet para os pontos de recolha onde não está disponível um AP Wi-Fi.

A proposta inicial era a construção de dois protótipos SMR com as mesmas funcionalidades de base, tendo um deles a capacidade adicional de comunicar com a Internet utilizando um módulo GSM (GPRS) incluído no próprio SMR.

Na Figura 6.4 pode-se observar alguns dos componentes utilizados para a construção dos protótipos.

Para construir um protótipo com capacidades de comunicação por GSM foram encontrados diversos módulos, alguns que incluíam até funcionalidades como Bluetooth e GPS. O módulo mais económico encontrado foi o M590E que se mostra no lado esquerdo da Figura 6.4.

Mesmo optando pelo módulo M590E o agravamento no custo de produção do SMR mais do que duplicaria e, para além deste agravamento, seria necessário equipar cada um deste tipo de SMR com um cartão GSM que traria mais um encargo mensal para as empresas de recolha.

Esta solução seria ainda menos eficiente nos pontos de recolha onde existem mais do que um depósito a monitorizar, obrigando a que fosse necessário adquirir um cartão GSM para cada SMR.

Concluiu-se que, para os pontos de recolha sem acesso à Internet, uma solução mais económica passa pela aquisição de apenas um cartão GSM e um equipamento *Hotspot* 3G ou 4G criando um AP Wi-Fi que permite que vários SMR (normais) acedam à Internet partilhando a mesma ligação. Outra vantagem desta solução é que não é acrescentada complexidade ao *firmware* do SMR para dar suporte a outro tipo de sistema de comunicação, nem torna necessário o redesenho da *Printed Circuit Board* (PCB) para possibilitar a conexão do módulo GSM adicional, evitando ainda o já mencionado aumento do custo de produção de uma variante do produto original.

Apesar de não existir a necessidade de construção do protótipo com capacidades de comunicação por GSM foram construídos dois protótipos SMR normais para fins de despiste de avarias e testes de desempenho de componentes.

De notar que ainda se chegou a efetuar alguns testes de comunicação por GSM utilizando comandos AT num dos protótipos.

Um dos componentes alvo de testes intensivos foi o módulo amplificador de sinal dos sensores de carga. Foram adquiridas duas versões com características semelhantes deste módulo, mas de dois fabricantes distintos, uma delas com *shield* de RF.

Após a construção e desenvolvimento do código para os dois protótipos foram recolhidas várias leituras das mesmas cargas com ambos os protótipos a fim de ser possível comparar os diferentes módulos para optar pelo mais fiável e estável. De referir que o custo destes módulos amplificadores de sinal é semelhante, por isso o custo não pesou como fator a ter em conta na escolha do melhor módulo.

Como apresentado na Figura 6.5, a estrutura de suporte do SMR é constituída por duas bases circulares (pratos) sendo o superior destacável a fim de facilitar a sua limpeza e o inferior onde é montada toda a parte eletrónica do SMR. A base inferior é constituída também por uns pés (quatro) que elevam a mesma do solo. Estes pés foram modelados em 3D e posteriormente impressos numa impressora 3D.

A construção do protótipo tem por objetivo conseguir um SMR autónomo energéticamente e com capacidades de comunicar com um conjunto de *web services* disponíveis na Internet. Para além destes requisitos o SMR tem que ser capaz de realizar a leitura do estado da sua bateria e nível de fluido em um depósito com capacidade para 50 Litros. A fim de se integrar facilmente no processo de recolha já implantado e de forma a evitar o seu comprometimento físico e funcional, o SMR deve estar desacoplado do depósito de recolha de óleo usado.

Relativamente à componente estrutural do SMR, indo de encontro com os objetivos, pretende-se construir um SMR semelhante a uma balança, com uma base que acolhe toda a componente eletrónica e outra base removível que servirá como proteção da primeira, contra o impacto e sujidade. Com este tipo de SMR será possível efetuar várias recolhas dos depósitos utilizando sempre o mesmo SMR. Apesar de ser com menor periodicidade do que as recolhas, o SMR irá requerer manutenção da sua bateria (caso não seja possível ligar-lo a um transformador na rede elétrica) e necessitará também receber uma configuração inicial sempre que instalado num novo local de recolha. A necessidade em desenvolver uma AMA tornou-se clara quando analisado o processo de configuração do SMR. Já que o preenchimento do Token de validação associado a cada cliente e das coordenadas (latitude e longitude) de cada ponto de recolha seriam campos de configuração sujeitos a erros de digitação por parte do instalador do SMR.

Figura 6.5: Protótipo SMR.

Durante o processo de análise ao desenvolvimento do *firmware* para o SMR foi identificada a necessidade, por questões de consumo energético, de fazer com que o SMR passasse a maior parte do tempo em modo de descanso (*Deep Sleep*) obrigando a que sempre que existisse um reinício fosse avaliado qual o estado que originou esse reinício a fim de determinar o que fazer em seguida.

No código do *firmware* do SMR foram definidas constantes globais, através da passagem direta ao compilador de *defines*, que se apresenta no Bloco 6.1.

Ao implementar o mecanismo de controlo ao consumo energético verificou-se que o MCU utilizado tinha uma limitação quanto ao tempo máximo (consecutivo) de descanso, de aproximadamente 71 minutos (4 294 967 295 μs). Esta limitação técnica, trouxe maior complexidade ao ciclo de reinícios tratado pelo *firmware* do SMR. Para facilitar a visualização dos vários passos de execução percorridos em cada reinício desenhou-se, já depois da fase de design, o fluxograma apresentado nas Figuras 6.6 e 6.7 que acrescentaria novas atividades ao diagrama BPMN apresentado na Figura 5.8 do Capítulo 5.

Ligação Elétrica das Células de Carga

Como já abordado no capítulo relativo ao estado da arte, o circuito elétrico escolhido para tirar partido dos sensores de carga por medição de tensão foi o circuito de losango ou ponte de Wheatstone. Cada célula utilizada tem a capacidade máxima de carga de 50 Kg e

```
// ############### DEFINES ###############

// ---------------> DEBUG
#define DEBUG_ON

// ---------------> ADC_VCC
#define VCC_ADJ 1.05886

// ---------------> SMR BOOT REASON
#define SMR_BOOT_REASON__DEFAULT_RST       0  // normal startup by power
    on
#define SMR_BOOT_REASON__WDT_RST           1  // hardware watch dog reset
#define SMR_BOOT_REASON__EXCEPTION_RST     2  // exception reset, GPIO
    status won't change
#define SMR_BOOT_REASON__SOFT_WDT_RST      3  // software watch dog reset
    , GPIO status won't change
#define SMR_BOOT_REASON__SOFT_RESTART      4  // software restart ,
    system_restart , GPIO status won't change
#define SMR_BOOT_REASON__DEEP_SLEEP_AWAKE 5  // wake up from deep-sleep
#define SMR_BOOT_REASON__EXT_SYS_RST       6  // external system reset

// ---------------> SMR SOFTRESET TYPE
#define SMR_SOFTRESET_TYPE__NOT_DEFINED        0
#define SMR_SOFTRESET_TYPE__CONFIG_SET         1
#define SMR_SOFTRESET_TYPE__CONFIG_TIMEOUT     2
#define SMR_SOFTRESET_TYPE__CONFIG_SEND_OK     3
#define SMR_SOFTRESET_TYPE__CONFIG_SEND_ERROR 4
#define SMR_SOFTRESET_TYPE__WAKEUP2WORK        5

// ---------------> 1H
#define ONE_DEEP_SLEEP_HOUR 3600*1000000
```

Listing 6.1: Constantes globais usadas no *firmware* para controlo dos modos de reinício do SMR.

Figura 6.6: Fluxograma de funcionamento do SMR.

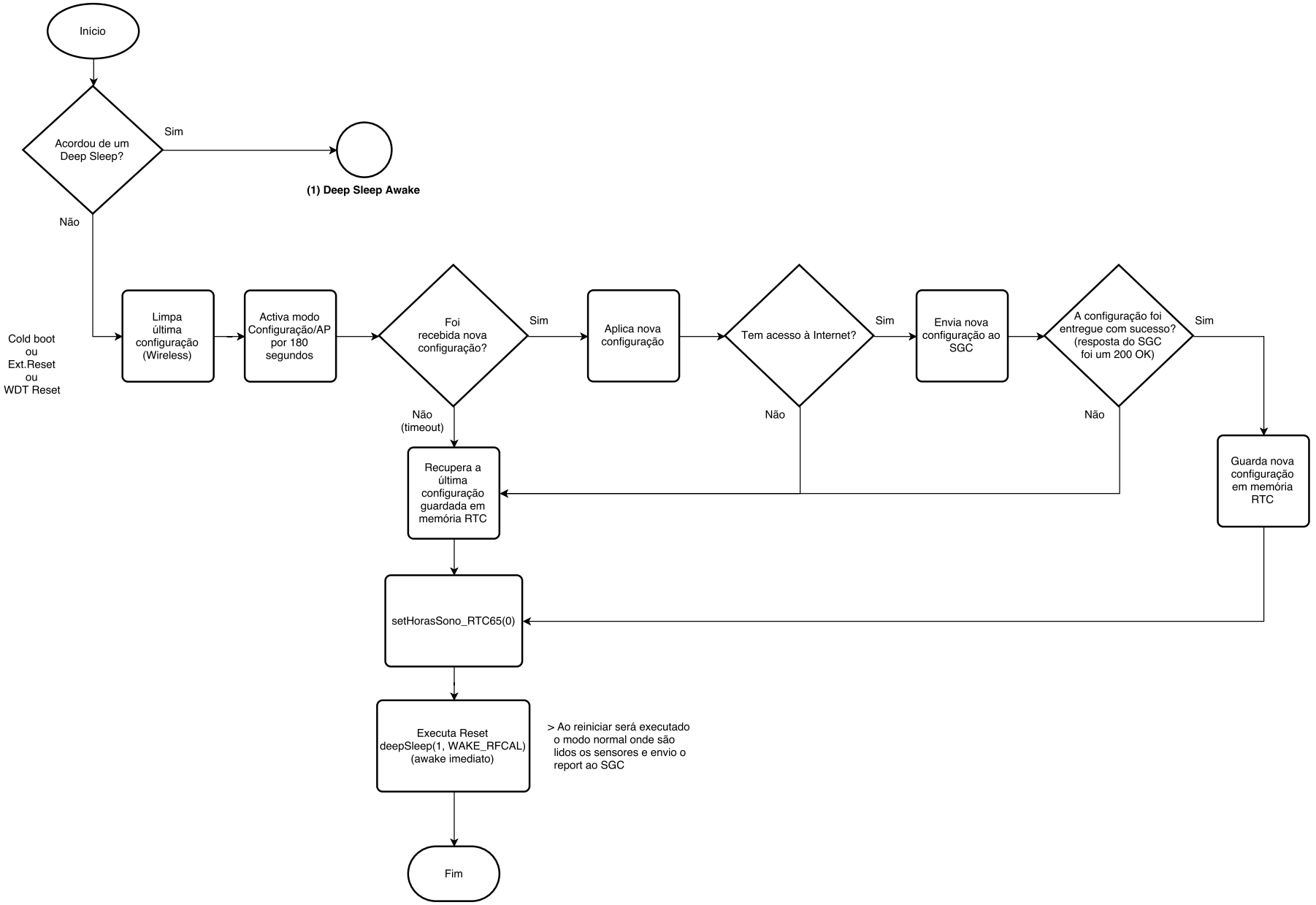
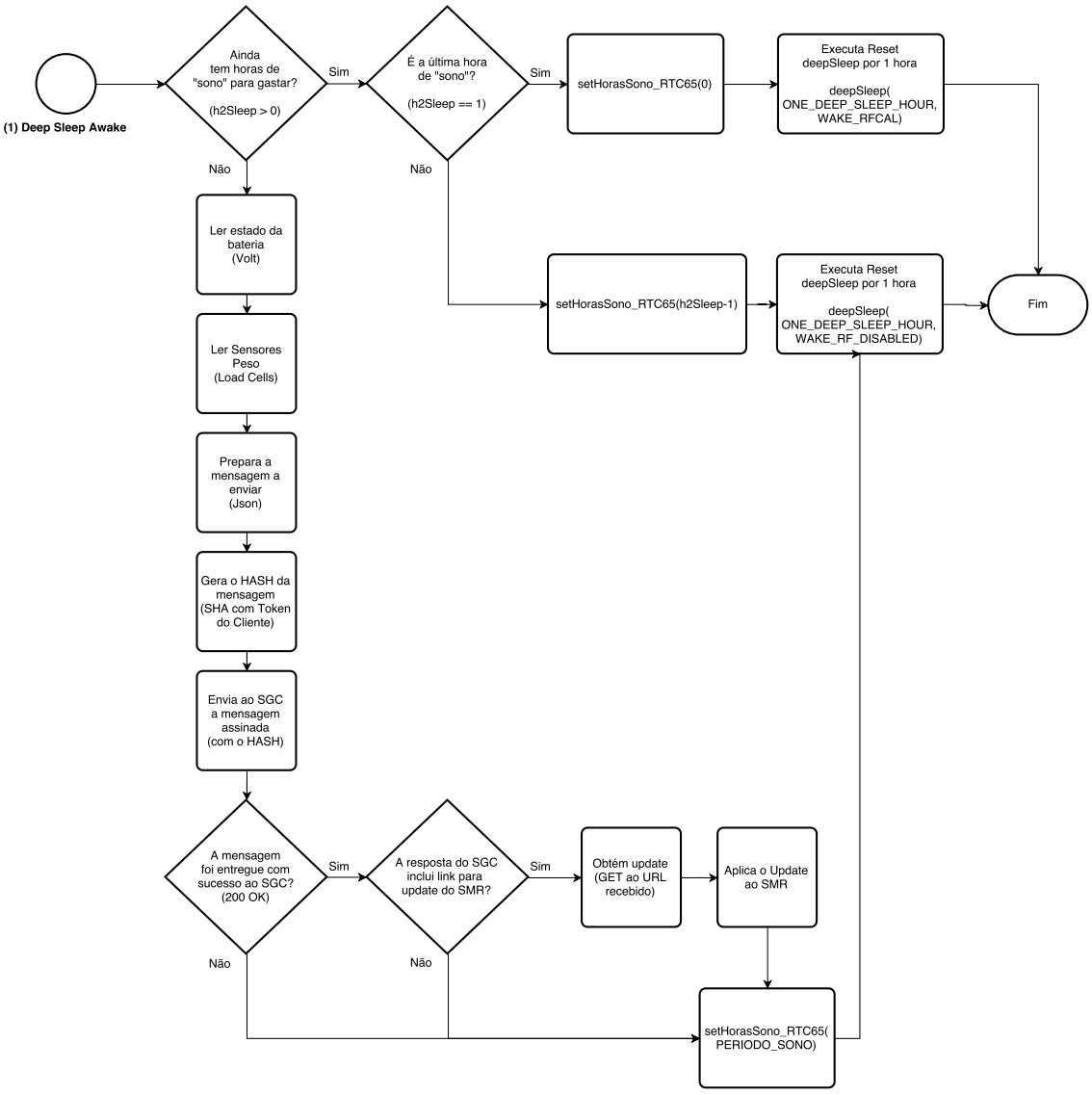


Figura 6.7: Fluxograma de funcionamento do SMR (cont.).



```
// ############## FUNCTIONS ##############

// ---------------> SCALE FUNCS

void configScale() {
  scale.begin(A1, A0);
  scale.set_scale(22.5f);
  scale.set_offset(190000);
}

float getScaleRead() {
  delay(1);
  scale.power_up();
  return scale.get_units(5);
}

void setScaleStandby() {
    scale.power_down();      // put the ADC in sleep mode
}
```

Listing 6.2: Funções para configuração e leitura dos sensores de carga do SMR.

uma resolução de 100 gamas. Foram utilizadas quarto células de carga prefazendo a ponte completa um total de carga máximo de 200 Kg.

No código do *firmware* do SMR foram criadas funções específicas para configurar e ler os sensores de carga, apresenta-se no Bloco 6.2 o código dessas funções.

No Bloco 6.3 pode-se observar algumas das estruturas e variáveis globais usadas no *firmware* do SMR.

Memória RTC

Ao estudar sobre os mecanismos de memória persistente disponibilizadas pelo MCU ESP8266 verificou-se que apesar deste permitir a escrita em memória EEPROM que esta não era a melhor forma de guardar as configurações do SMR já que estas poderiam ser feitas repetidas vezes. Esta limitação da EEPROM prende-se com o número de escritas ser limitado, originando leituras erradas ao se ultrapassado esse limite. A solução passou por tirar partido da memória RTC que apesar de não ter o acesso facilitado em termos de código, não apresenta limitações quanto ao número de escritas.

Para facilitar o acesso aos valores colocados nesta memória foram criadas funções específicas para cada estrutura de dados ou variável. No Bloco 6.4 mostra-se o código das funções criadas que permitem guardar e ler os dados em memória RTC.

Modos de Economia de Energia do MCU ESP8266

O SMR requer uma fonte de energia para funcionar. A solução proposta no protótipo desenvolvido permite a utilização de duas possíveis fontes de energia, através de um transformador de corrente de 220V AC para 5V DC ou utilizando uma bateria de lítio de duas células em série perfazendo uma tensão de 7,4V DC. O segundo protótipo construído foi equipado

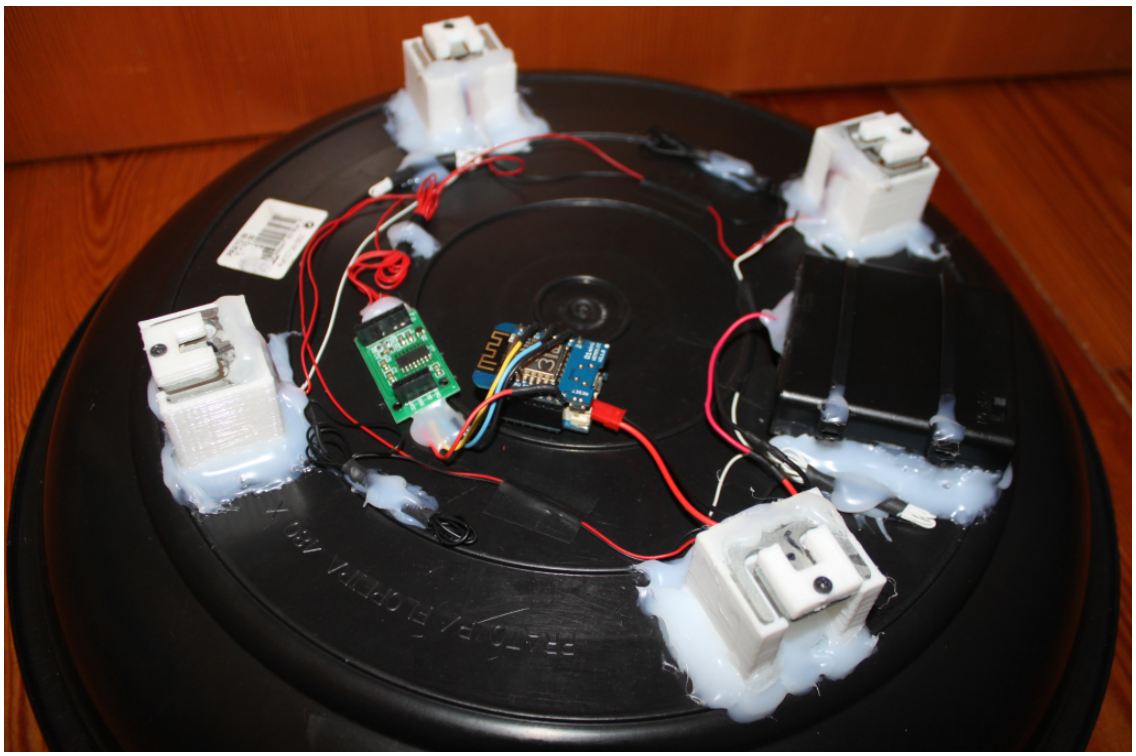
```
// ############## GLOBAL VARS ##############

typedef struct {
  char AP_SSID[32]; // tamanho max. SSID = 32 char
  char AP_PASS[63]; // tamanho max. passphase = 63 char
  int nrHorasAteProxReport;
  int other;
} rtcStore __attribute__((aligned(4)));

int tipoSoftReset_RTC64;
int horasSono_RTC65;
rtcStore SMRData_RTC66;

// ---------------> WiFiManager
WiFiManager wifiManager;

// ---------------> SMR
const String SMR_UUID = "SMR2017080916340001";

// ---------------> WifiManager
char *ssid      = "";               // Set you WiFi SSID
char *password  = "";               // Set you WiFi password

// ---------------> Client Data
char cl_email[50] = "support@smartdepot.duckdns.org";
char cl_token[200] = "TOKEN_SAMPLE_173CHARS";

// ---------------> resetInfo
rst_info *myResetInfo = ESP.getResetInfoPtr();
int resetReason = myResetInfo->reason;

// ---------------> Sensor Carga
HX711e scale;
```

Listing 6.3: Variáveis globais e estruturas de dados usados no *firmware* do SMR.

```
// --------------> MEM RTC FUNCS

void setSoftResetType_RTC64(int t) {
  tipoSoftReset_RTC64 = t;
  system_rtc_mem_write(64, &tipoSoftReset_RTC64, sizeof(
    tipoSoftReset_RTC64));
}

int getSoftResetType_RTC64() {
  system_rtc_mem_read(64, &tipoSoftReset_RTC64, sizeof(
    tipoSoftReset_RTC64));
  return tipoSoftReset_RTC64;
}

void setHorasSono_RTC65(int h) {
  horasSono_RTC65 = h;
  system_rtc_mem_write(65, &horasSono_RTC65, sizeof(horasSono_RTC65));
}

int getHorasSono_RTC65() {
  system_rtc_mem_read(65, &horasSono_RTC65, sizeof(horasSono_RTC65));
  return horasSono_RTC65;
}

void setSMRData_RTC66(rtcStore smr) {
  SMRData_RTC66 = smr;
  system_rtc_mem_write(66, &SMRData_RTC66, sizeof(SMRData_RTC66));
}

rtcStore getSMRData_RTC66() {
  system_rtc_mem_read(66, &SMRData_RTC66, sizeof(SMRData_RTC66));
  return SMRData_RTC66;
}
```

Listing 6.4: Funções para acesso à memória RTC do SMR.

Figura 6.8: Vista inferior do SMR.



com uma caixa com capacidade para 4 baterias comuns tipo AA recarregáveis, que se pode observar no lado direito da Figura 6.8. As baterias AA recarregáveis fornecem uma tensão de 1,25V DC o que permite ter um total de 5V por caixa.

Foi necessário utilizar um regulador de tensão para limitar os 3,3V DC necessários ao bom funcionamento do MCU ESP8266, montado na placa de cor azul que se encontra no centro da Figura 6.8, já que a alimentação fornecida pela caixa instalada é de 5V. Tendo-se trabalhado em laboratório com um LD1117 por facilidade de manuseamento e optado pela utilização de um Ht7333-A em protótipo já que este último é um componente SMC, com um tamanho muito inferior, que permite a construção de uma placa de circuito impresso de menores dimensões.

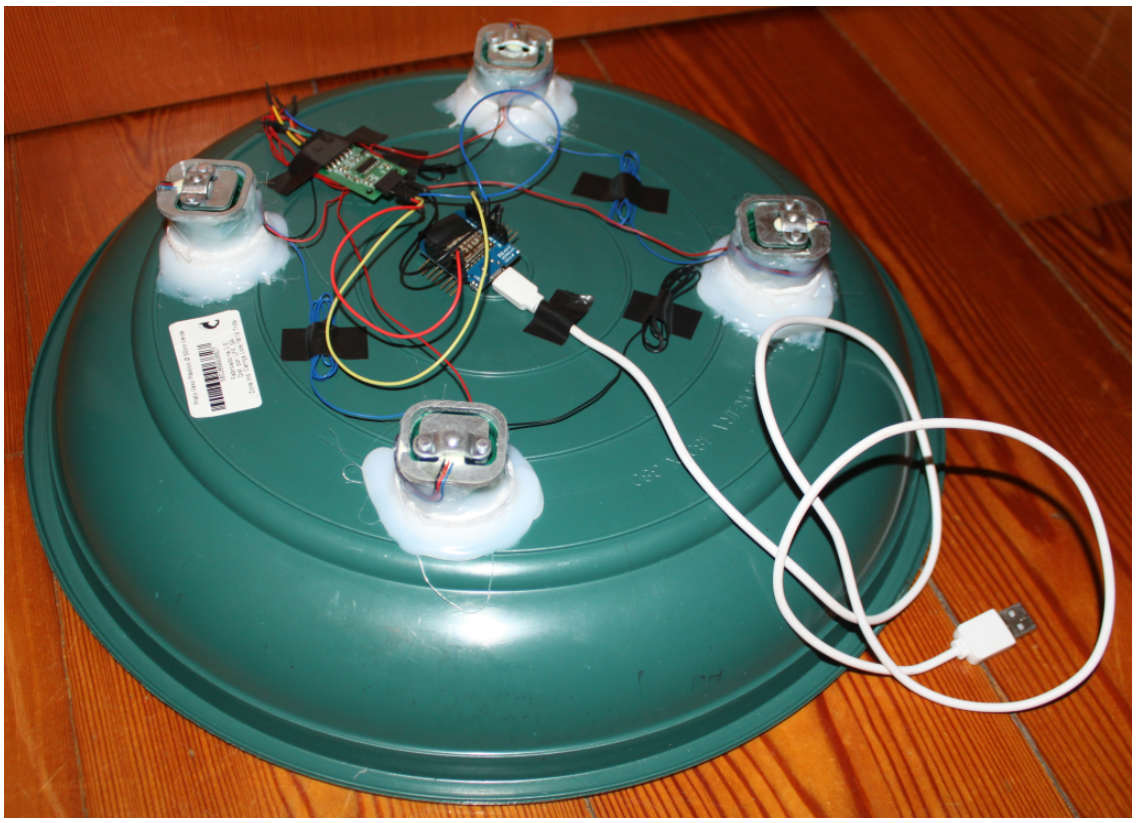
Quando da impossibilidade da utilização de um transformador ligado a uma tomada 220V para alimentar o SMR, que aparenta ser a situação mais comum, a solução será usar a alimentação por bateria. Isto apresenta um problema que o SMR tenta minimizar alternando entre dois estados de funcionamento, o normal e em standby.

O estado normal de funcionamento do SMR requer a utilização do módulo de System on a Chip (SoC) Wifi que é exigente em termos de consumo de energia.

Pré-Protótipo

Foi construída uma primeira versão do protótipo SMR, apresentado na Figura 6.9, que serviu para avaliar a estrutura e componentes a usar no protótipo de teste. Este primeiro protótipo serviu também como referência para a aferição do protótipo final, tendo permitido comparar

Figura 6.9: Pré-protótipo SMR.



leituras com componentes semelhantes e testes estruturais essenciais para tomar decisões de como avançar na construção do mesmo.

Modelação e Impressão 3D

Ao contrário do primeiro protótipo usado em laboratório que recebeu quatro pés feitos de tampas plásticas e cola quente para dar suporte a sua plataforma base, foi desenhado um modelo 3D para os pés do novo protótipo, apresentado na Figura 6.10. Posteriormente forem impressos quatro exemplares deste modelo numa impressora 3D. Apresenta-se na Figura 6.11 um foto tirada quando da impressão das três últimas peças.

Na Figura 6.12 pode-se ver os sensores de carga por tensão utilizador e os pés criados para os suportar.

6.3 Desenvolvimento da AMA

Para facilitar a configuração dos SMR colocados nos pontos de recolha foi desenvolvida uma aplicação móvel Android que permite ao funcionário da empresa de recolha identificar-se através da introdução do seu email e senha de acesso desde que tenha acesso à Internet.

Figura 6.10: Modelo 3D dos pés para o SMR.

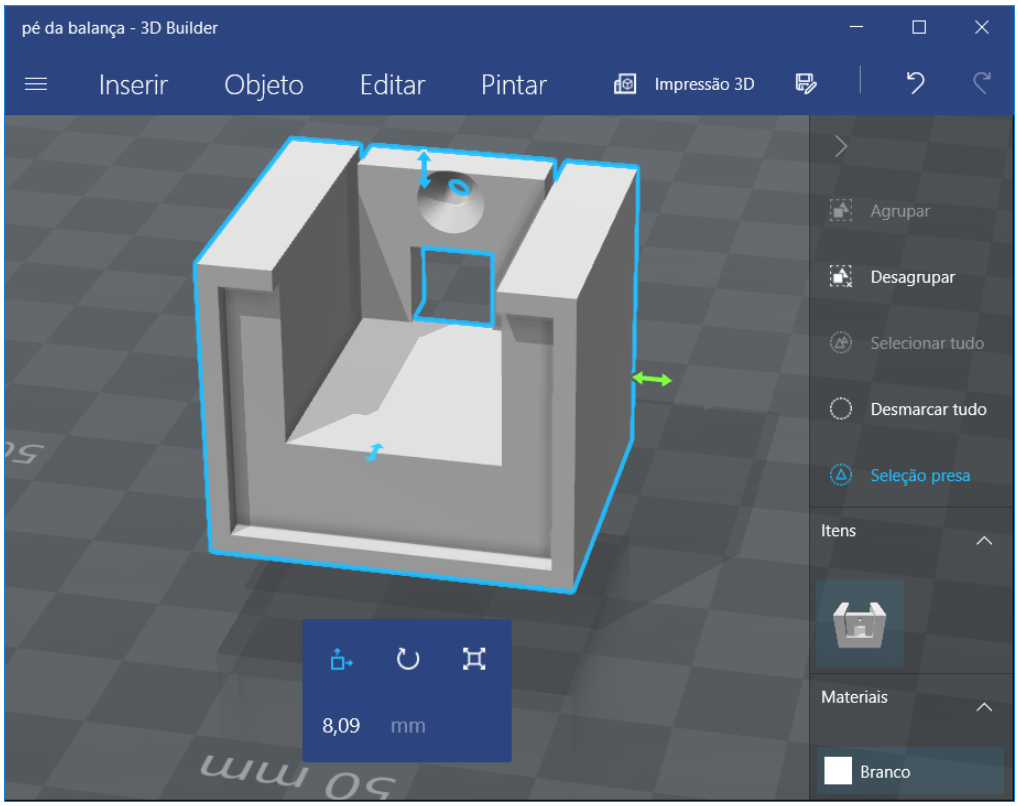


Figura 6.11: Impressão do modelo 3D dos pés para o SMR.

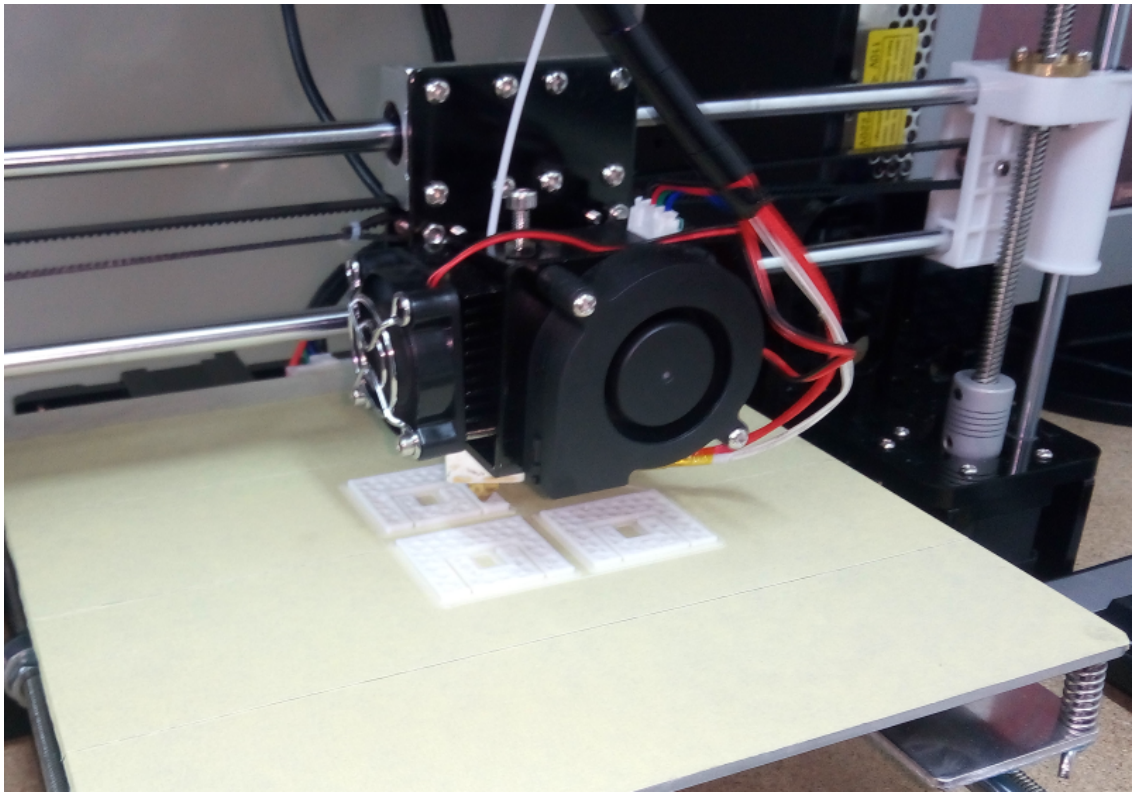
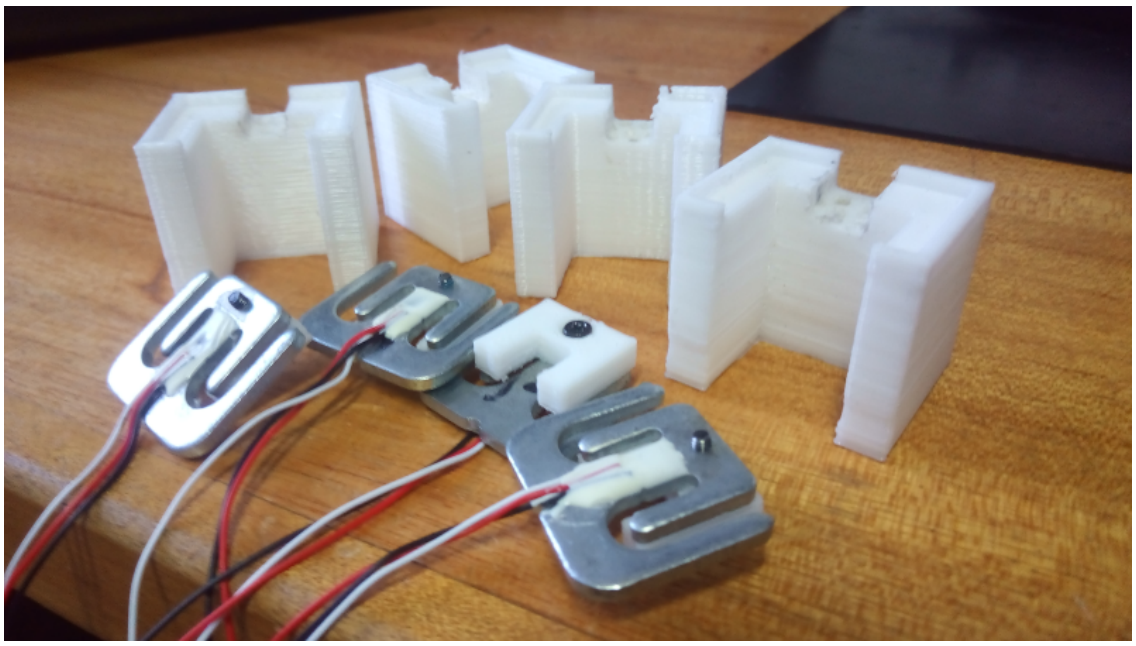


Figura 6.12: Sensores de carga por tensão e pés para o SMR.



Quando o SGC aceita a autenticação do utilizador da AMA gera um token único que é guardado na base de dados do SGC, associada à empresa cliente em causa, e é enviado como resposta à aplicação móvel que também o guarda em ficheiro. Este *token* servirá para ativar a aplicação dando-lhe acesso a todas as funcionalidades e será enviado para cada um dos SMR configurados pela AMA. O SMR irá utilizar este *token* na cifragem das mensagens que enviar para o SGC.

Uma vez ativada a aplicação não será necessária nova autenticação em futuras utilizações, já que o *token* fica guardado em ficheiro no dispositivo móvel deixando assim de ser necessário novo acesso à Internet.

Para efetuar a configuração de um SMR o utilizador da aplicação móvel necessita efetuar uma sequência de três passos controlados pela aplicação:

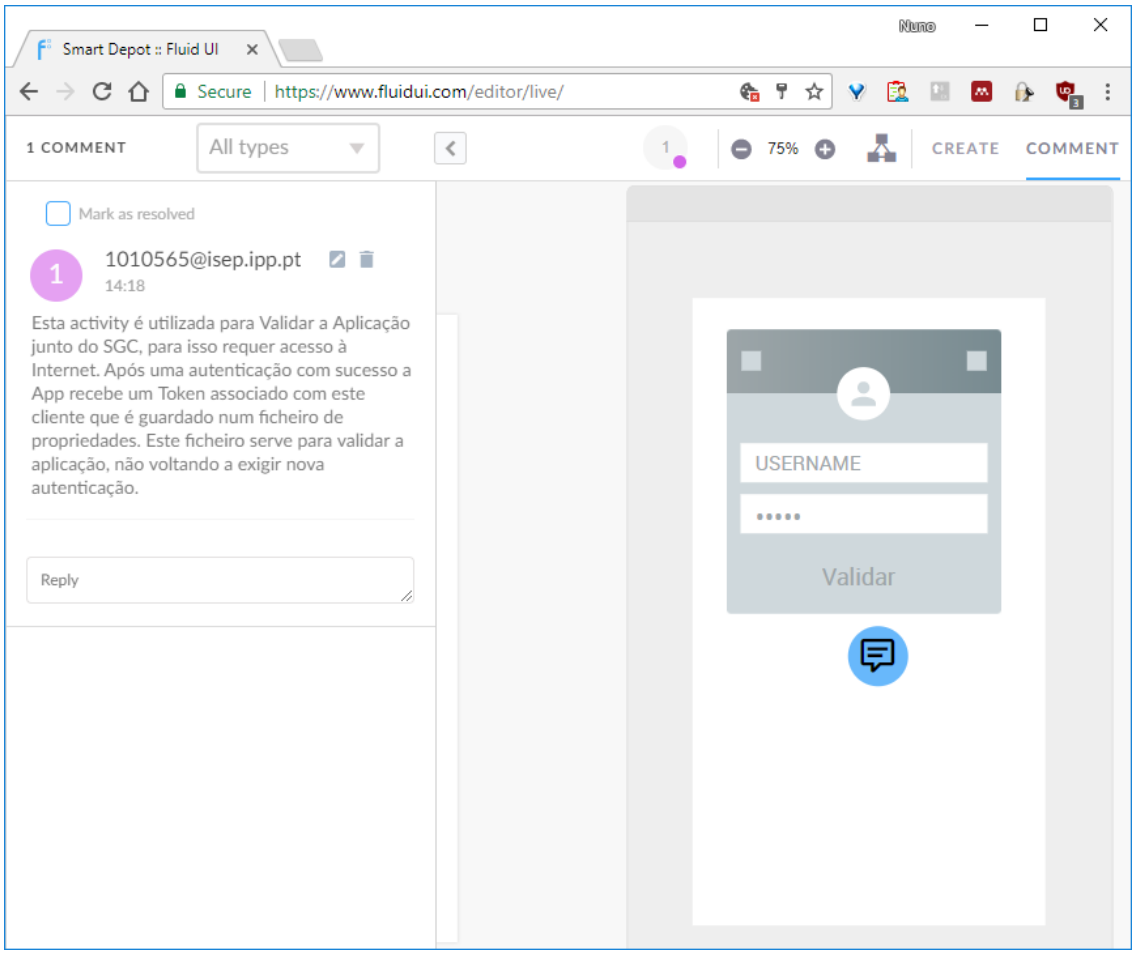
1. Obter as suas coordenadas GPS, sendo-lhe pedido para ativar esta opção no seu equipamento;
2. Estabelecer uma ligação ao AP Wi-Fi do SMR a fim da aplicação obter a lista de APs visíveis pelo SMR;
3. Escolher e enviar a configuração ao SMR.

O objetivo desta aplicação é auxiliar o funcionário da empresa de recolha na configuração dos SMR instalados em cada ponto de recolha, facilitando a aquisição das coordenadas GPS de cada local a fim de as enviar ao SMR e permitindo a obtenção e envio do Token de validação do cliente ao SMR, assim como outras configurações como, qual o SSID do AP Wi-Fi e senha a utilizar pelo SMR para se ligar à Internet ou informações relacionadas com o ponto de recolha como, nome do local, pessoa de contato e telefone de contato.

Quando se idealizava a AMA desenhou-se em papel um esboço de que janelas (*activities*) seriam necessárias. Este esboço foi então passado para o formato digitar com auxílio de

uma ferramenta de *Mockup online*. Foram associadas alguns comentários as várias janelas desenhadas para registar notas a ter em consideração durante o desenvolvimento. Apresentamos na Figura 6.13 um exemplo de uma das notas que foi anexa à *activity* de validação da AMA.

Figura 6.13: *Activity* da AMA com comentários.



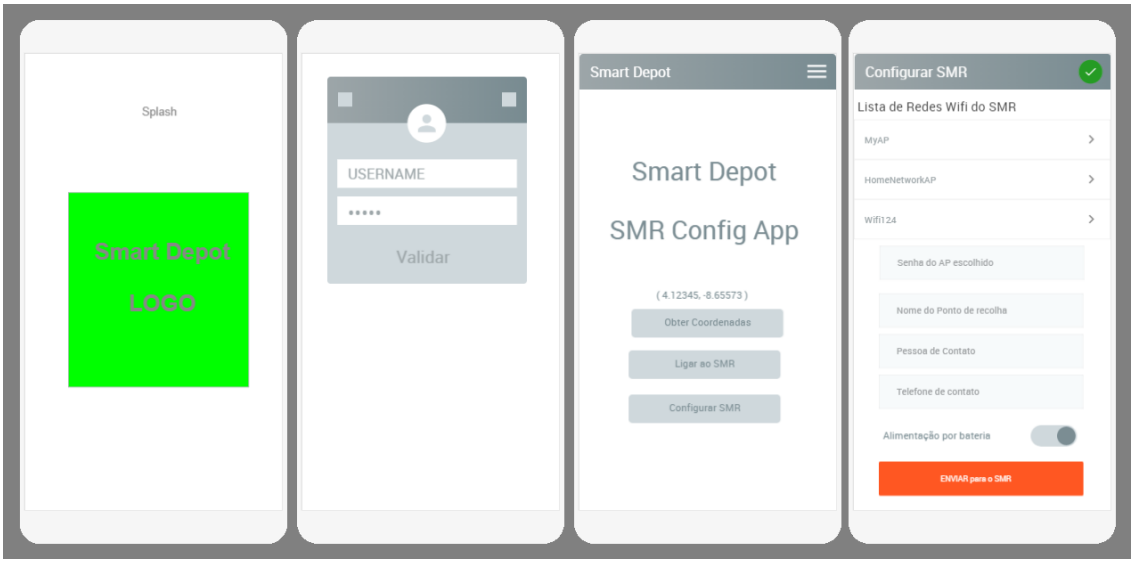
Foram identificadas quatro *activities* que se apresentam na Figura 6.14.

6.3.1 Ambiente de Desenvolvimento

Foi utilizado o IDE Android Studio da Google onde se criou um projeto para "Phone and Tablet"com o SDK mínimo limitado ao API 15 (Android 4.0.3). Apresenta-se na Figura 6.15 um *screenshot* do IDE utilizado onde se mostra do lado esquerdo a estrutura do projeto e na zona de edição dois ficheiros abertos, o da esquerda "ActEntrada.java"usado pela classe responsável pela janela de menu da aplicação e o ficheiro à direita é o da classe assincrona responsável por efetuar o pedido do Token ao SGC.

Para apoio ao desenvolvimento da AMA foi utilizada a documentação Android *online* da Google com realce para os seguintes pontos:

Figura 6.14: *Mockup* das *activities* da AMA.



- Requisitar permissões de acesso à Internet, ao controlo do módulo Wi-Fi e GPS, assim como de escrita de ficheiros;
- Criação, escrita e leitura de propriedades (pares chave, valor) em ficheiro;
- Envio de pedidos HTTP, métodos GET e POST, para serviços *web online* e fornecidos pelo SMR;
- Leitura das coordenadas GPS do equipamento móvel, latitude e longitude;
- Execução assíncrona dos pedidos HTTP a fim de não bloquear as *Activities* de interface com o utilizador;
- Passagem de parâmetros por *Bundle* entre *activities*, quando da chamada as Intents;
- Associação de *listeners* a botões normais e de menu e respetivos métodos de *handle*;
- Introdução de *Activity* de *Splash* para verificar a existência de validação da aplicação e consequente decisão de qual a seguinte *Activity* a ser apresentada.

O Bloco 6.5 representa uma mensagem JSON respondida pelo SMR, com a listagem das redes Wi-Fi captadas, quando o utilizador da AMA efetua o pedido de ligação ao SMR.

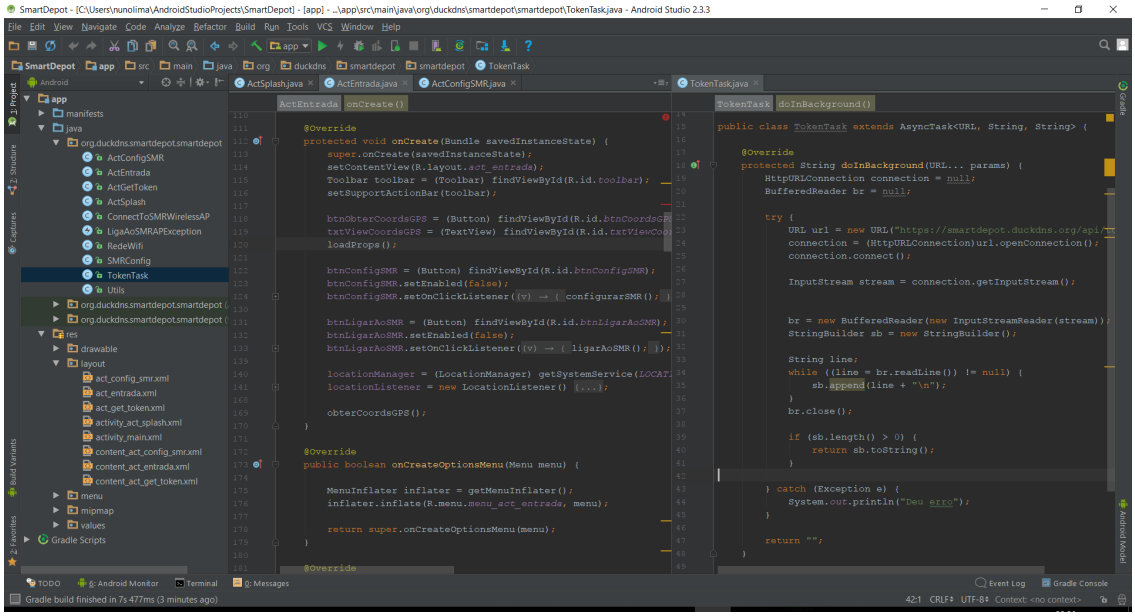
Na Figura 6.16 apresentam-se *screenshots* de todas as janelas da AMA obtidos durante o processo de configuração do protótipo SMR, desde a abertura da aplicação até ao envio da configuração ao SMR.

6.3.2 Qualidade e Testes ao AMA

Durante o desenvolvimento da AMA foram usados dois dispositivos Android, um *Smartphone* na versão 6.0 e um *Tablet* na versão 4.3, ambos com capacidades de acesso Wi-Fi e GPS, onde foram instaladas e testadas as diferentes versões desenvolvidas.

Para testar todas as funcionalidades da AMA foi necessário utilizar o protótipo SMR e ter acesso à API SGC.

Figura 6.15: Projeto Android Studio da AMA.



A AMA utilizou o protótipo SMR para obter a listagem dos AP Wi-Fi capturados por este e para aplicar as configurações escolhidas pelo utilizador, através do envio de uma mensagem JSON com o método POST do *Hyper Text Transfer Protocol* (HTTP).

O acesso ao SGC foi necessário para testar a validação da AMA, obtendo um *token JSON Web Token* (JWT), feita no *web service*
login através do envio do Email e Senha do utilizador. Outro *web service* do SGC necessário para testar a AMA foi o que recebe as configurações aplicadas aos SMR em formato JSON.

6.4 Desenvolvimento do SGC

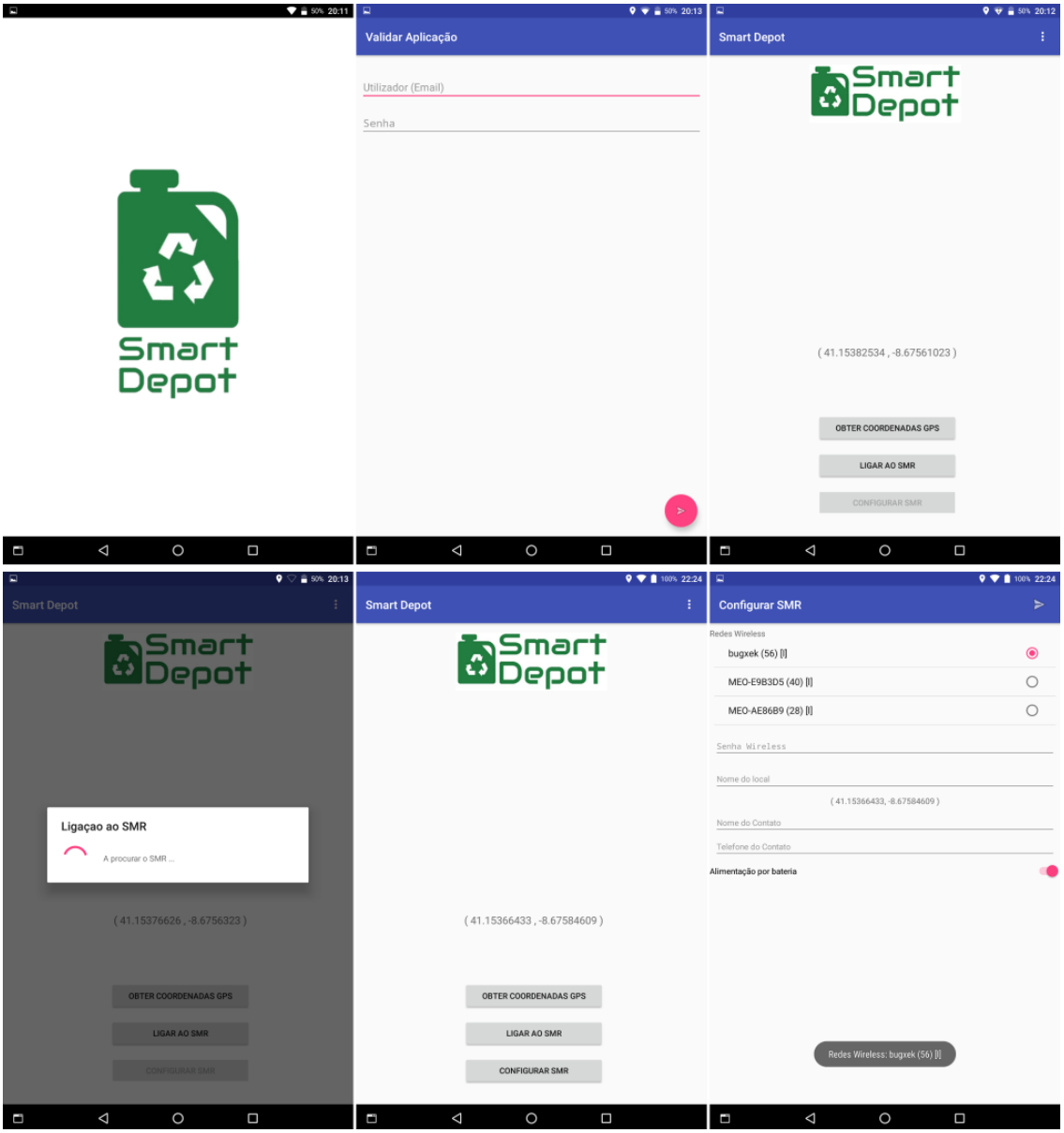
A API do SGC tem por objetivo fornecer um interface, através de web services REST, que permitam gerir (registar, editar e listar) as seguintes entidades:

- Clientes;
- Grupos de utilizadores;
- Utilizadores;
- SMR;
- Configurações dos SMR;
- Leituras dos SMR;
- Rotas.

6.4.1 Ambiente de Desenvolvimento

Foi utilizado o IDE PyCharm da JetBrains onde se criou um projeto Python na versão 3.x. Apresenta-se na Figura 6.17 um *screenshot* do IDE utilizado no desenvolvimento da API do SGC, onde se mostra do lado esquerdo a estrutura de diretórios e ficheiros do projeto.

Figura 6.16: *Screenshots* das *activities* da AMA.



```
{
  "networks": [
    {
      "ssid": "MEO-E9B3D5",
      "rssi": "74",
      "encryptionType": "1"
    },
    {
      "ssid": "NOS-9E20",
      "rssi": "60",
      "encryptionType": "1"
    },
    {
      "ssid": "bugxek",
      "rssi": "58",
      "encryptionType": "1"
    },
    {
      "ssid": "MEO-WiFi",
      "rssi": "32",
      "encryptionType": "0"
    }
  ]
}
```


Listing 6.5: Mensagem JSON com lista de redes Wi-Fi captadas pelo SMR

6.4.2 Documentação da API REST

Para documentar e organizar a análise dos serviços necessários a ser implementados pela API REST do SGC foi usada a especificação OpenAPI ((OpenAPI, 2017)), anteriormente conhecida por Swagger. Esta especificação usa Yaml ou JSON como sintaxe e permite descrever as rotas a implementar especificando quais os métodos HTTP disponíveis em cada rota. Também faz parte da especificação descrever o tipo e estrutura de dados recebidos e respondidos por cada método. Outras características documentadas pela especificação são o tipo de autenticação necessária, qual o protocolo (HTTP ou HTTPS), versão da implementação e inclusão de exemplos de dados usados nas requisições e respostas dos métodos HTTP.

Para escrever a documentação foi usado o editor *online* apresentado na Figura 6.18.

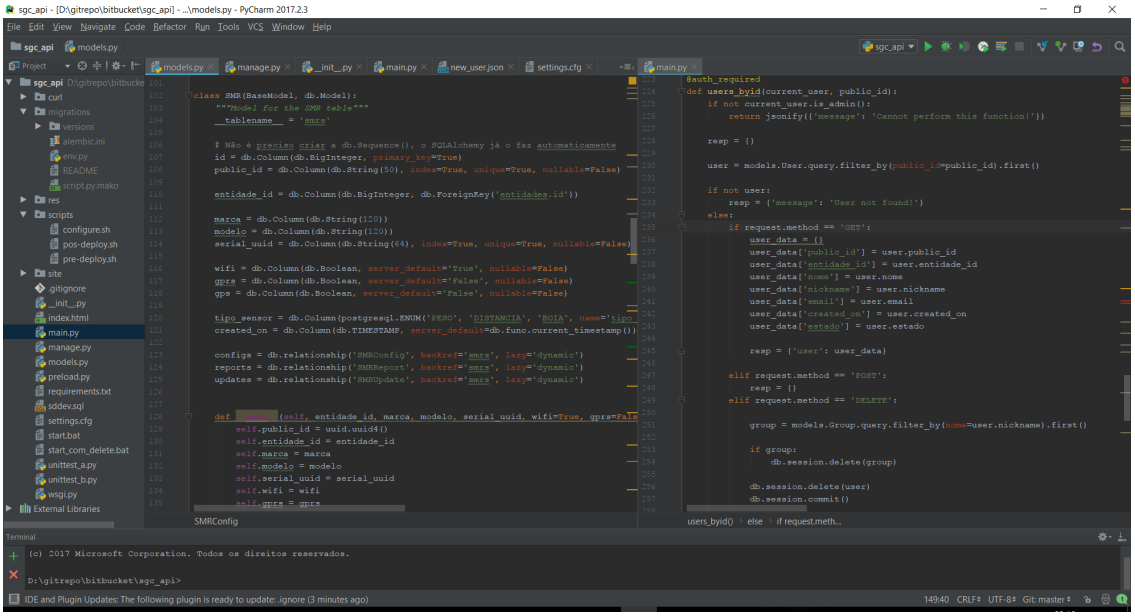
6.4.3 Comunicação Segura com Hyper Text Transfer Protocol Secure (HTTPS)

A utilização do protocolo HTTPS requer a aquisição de um certificado *Secure Sockets Layer* (SSL) junto de uma *Certificate Authority* (CA) e a respetiva configuração do mesmo no servidor web responsável pelo domínio.

Certificado SSL

A Let's Encrypt é uma CA onde é possível gerar certificados SSL de confiança de forma gratuita. Após a criação do certificado SSL para o domínio "smartdepot.duckdns.org" o

Figura 6.17: Projeto PyCharm do SGC.



mesmo foi configurado no Nginx do nosso servidor e foi também criado um *bash script*, chamado periodicamente via crontab para manter o certificado SSL ativo.

Configuração do Nginx

Como servidor web foi usado o Nginx por este se apresentar como uma alternativa de alta performance e escalabilidade em comparação com o outro serviço concorrente à liderança de mercado, o Apache. O Nginx em termos de configuração assemelha-se ao Apache, utilizando também virtual hosts que são configurados em ficheiros localizados no diretório "sites-available" de onde são feitos links simbólicos para "sites-enable". Outro fator que levou à escolha do Nginx foi a sua boa integração com o serviço de gateway uWSGI (Web Server Gateway Interface) utilizado para executar aplicações desenvolvidas na linguagem Python, como é o caso da API do SGC desenvolvida no âmbito deste projeto. Por questões de segurança, o serviço de HTTP a correr na porta default (80) foi configurado para forçar o *redirect* para a porta 443 onde foi configurado o serviço HTTPS com o certificado SSL adquirido.

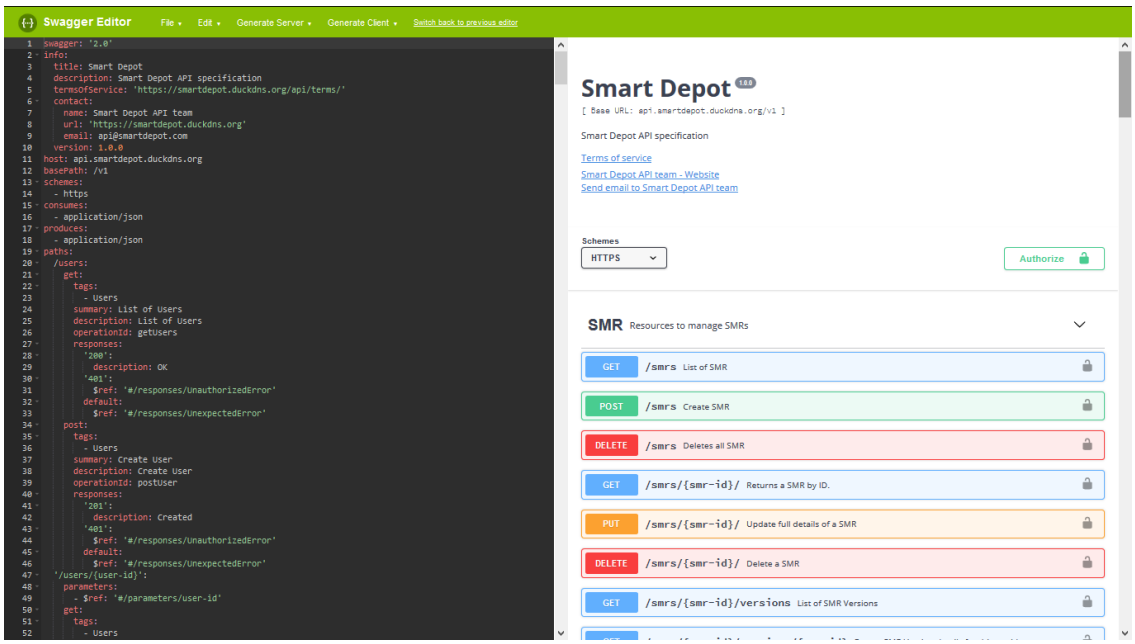
6.4.4 Autenticação no SGC

Nesta secção apresenta-se os detalhes sobre o processo de autenticação implementado no SGC.

Serviço /login com HTTP Basic Authentication (BA)

O web service que permite a autenticação dos utilizadores da API e obtenção do respetivo token usado na comunicação com os restantes serviços é garantida pelo método BA do

Figura 6.18: Editor Swagger *online* em (OpenAPI, 2017).



HTTP que recebe um username e uma password que são posteriormente comparadas com as credenciais guardadas na base de dados. Todas as senhas secretas guardadas em base de dados são cifradas com o algoritmo sha256 sendo utilizado o módulo Python werkzeug.security que facilita a geração e verificação destas.

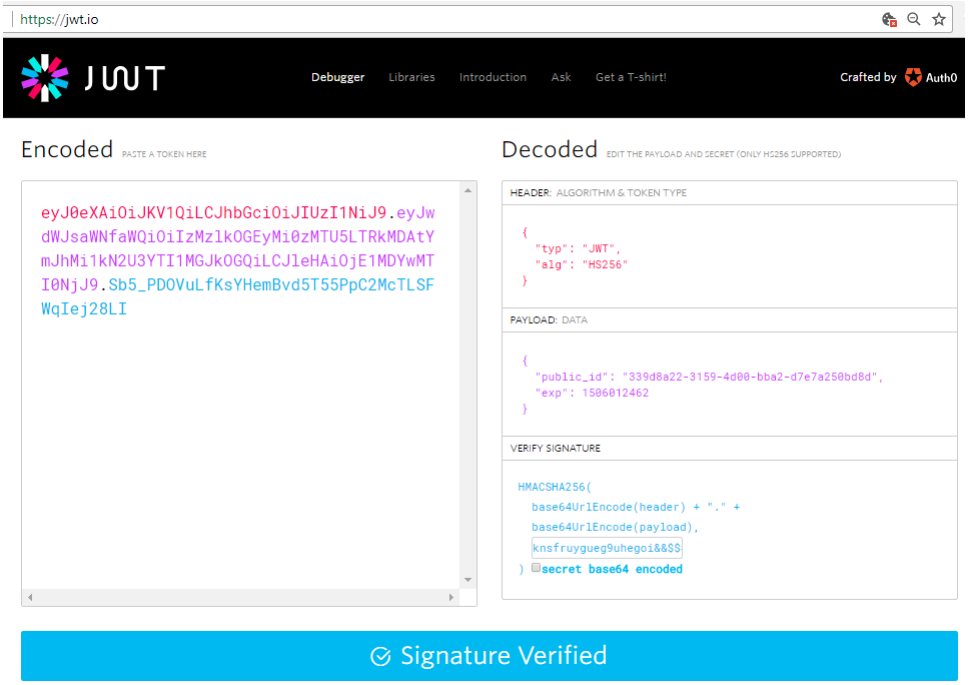
Identificador Público na URL

Todas as entidades utilizadas no modelo de dados do SGC utilizam um identificador único *Unique Identifier* (ID). No entanto este ID apenas é utilizado internamente, pelo Sistema de Gestão de Base de Dados (SGBD), pelo ORM SQLAlchemy e no código Python da API, já que para o exterior apenas passa um ID público gerado com o módulo UUID quando da criação do registo. Desta forma, é criada mais uma barreira de segurança contra ataques à API já que os ID usados para manipular os dados nunca são tornado públicos. Esta abordagem surgiu por ser necessário usar um ID na *Uniform Resource Locator* (URL) dos web services, por exemplo, /smrs/<ID>/report.

Token Gerado com JWT

Um token JWT não é apenas um HASH aleatório, trata-se de um conjunto de informação cifrada e estruturada em três partes separadas por pontos, "header.payload.signature". Na parte do *header* do JWT é usada para transportar a informação sobre o tipo de *token* e qual o algoritmo de *hashing* usado. A parte do *payload*, que é convertida em Base64 antes de todo o *token* ser cifrado, é onde se encontram todos os atributos do utilizador e outros metadados. A *signature* é a zona utilizada para validar se o remetente do JWT é quem diz ser ou se ocorreu alguma alteração durante a troca do *token*. Na Figura 6.19 mostra-se um exemplo do conteúdo de um JWT gerado pelo SGC decifrado, e com as partes de *payload* e

Figura 6.19: JWT *online debugger* em jwt.io.



signature ainda não convertidas para Base64. Para visualizar toda a estrutura de um *token* pode-se utilizar o JWT *Debugger* que se encontra disponível *online* (Auth0, 2017), onde foi obtida esta Figura 6.19.

A cada token gerado por um utilizador é guardado em base de dados para posteriormente ser utilizado no processo de assinatura e validação das comunicações trocadas entre os SMR e o SGC.

A utilização de um token JWT permitiu transportar informação cifrada no próprio token, em vez de apenas usar um hash aleatório sem significado. No Bloco 6.6 apresenta-se o código do serviço de //login que devolve um token JWT ao validar com sucesso as credenciais que lhe são passadas por HTTP BA.

Como se mostra no código do Bloco 6.6 o token transporta a informação do identificador público do cliente e a data sua data de expiração.

Entrega do Token ao SMR via AMA

Quando ocorre uma validação com sucesso o token devolvido é guardado na AMA e utilizado na configuração dos SMR. Este token é entregue, juntamente com os restantes parâmetros de configuração, aos SMR quando da configuração destes. É possível reutilizar o mesmo token na configuração de vários SMR.

Assinatura e Validação das Mensagens do SMR

O SMR guarda em zona de memória RTC, com auxílio das funções já apresentadas no Bloco 6.4, o *token* recebido na sua configuração e sempre que envia uma mensagem ao SGC

```
@app.route('/login', methods=['GET'])
def login():
    resp = {}
    auth = request.authorization

    if not auth or not auth.username or not auth.password:
        return make_response('Could not verify', 401, {'WWW-Authenticate
    ': 'Basic realm="Login required!"'})

    user = models.User.query.filter_by(nickname=auth.username).first()

    if not user:
        return make_response('Could not verify', 401, {'WWW-Authenticate
    ': 'Basic realm="Login required!"'})

    if check_password_hash(user.password, auth.password):
        token = jwt.encode(
            {'public_id': user.public_id, 'exp': datetime.datetime.
    utcnow() + datetime.timedelta(days=365)},
            app.config['SECRET_KEY'])

        return jsonify({'token': token.decode('UTF-8')})

    return make_response('Could not verify', 401, {'WWW-Authenticate': '
    Basic realm="Login required!"'})
```

Listing 6.6: Serviço de Login do SGC.

utiliza-o no processo de assinatura desta mensagem. Como a capacidade de processamento do MCU é limitada, seria morosa a cifra de todo o *payload* de cada mensagem. Devido a esta limitação, assinalou-se o *payload* em vez de o cifrar. No entanto, pode-se tirar partido em ter acesso ao *token*, que também existe do lado do recetor, decidimos o par mensagem e *token*, enviado por fim outro par mensagem e o *hash* da assinatura ao SGC.

O processo de validação efetuado pelo SGC ao receber a mensagem assinada começa por recuperar o *token* guardado na base de dados para este cliente, concatenando-o ao *payload* da mensagem recebida. A este par *payload* e *token* é aplicando o mesmo algoritmo (*Secure Hash Algorithms* (SHA)) usado pelo SMR quando da assinatura, terminando o processo de validação com a comparação da *hash* resultante com a *hash* recebida.

Ao obter uma correspondência positiva neste comparação temos a certeza que a mensagem foi enviada pelo SMR indicado e que a mensagem trocada não foi comprometida durante a comunicação desta.

6.4.5 ORM SQLAlchemy

O SQLAlchemy é um Object Relational Mapper (ORM) de base de dados fornecido como um módulo Python. Ao importar este módulo Python passamos a conseguir persistir dados e manipular os mesmo de forma facilitada através da manipulação de objetos. Estes objetos são instâncias de classes, definidas como classes entidade, que representam no universo OO as entidades associadas as tabelas da base de dados com as suas características e regras.

A metodologia utilizada pelo projeto Smart Depot para criar o modelo de dados foi a abordagem Code-first, que se carateriza por serem escritas as diferentes classes entidade em código OO, neste caso Python, que depois é utilizado por ferramentas do ORM para criar de forma automática as tabelas correspondentes na base de dados. Para este processo de criação automático ser possível, as entidades e tabelas de relacionamento tem de ser representadas no modelo, assim como todos os tipos de dados, limitações ou regras das suas propriedades. A utilização do ORM SQLAlchemy contribui com vários benefícios, entre eles:

- Código mais limpo: Evita-se ter de escrever comandos SQL em strings Python;
- Código mais seguro: Utilizar funcionalidades do SQLAlchemy ajudam o programador a não ocorrer em erros que original vulnerabilidades como a de SQL injection;
- Lógica simplificada: O SQLAlchemy permite uma abstração sobre a base de dados permitindo ao programador concentrar-se apenas em objetos Python, em vez de ter de pensar em ligações, tabelas, linhas e colunas, lidamos apenas com classes, instâncias e atributos.

A utilização do SQLAlchemy não inviabiliza a execução nativa de código *Structured Query Language* (SQL), podendo inclusive esta ser feita utilizando o próprio motor do SQLAlchemy.

A abordagem utilizada para criar a estrutura de dados de suporte ao SGC com o ORM foi a *code first* onde são escritas as classes entidade em código, neste caso Python, e depois são utilizadas as ferramentas disponibilizadas pelo próprio ORM que tratam de criar e mapear essas classes na base de dados, criando tabelas e campos que as relacionam.

Foi criada uma classe base, apresentadas no Bloco 6.7, para agregar todas as caraterísticas comuns das classes entidades.

O código que se mostra no Bloco 6.8 define a classe entidade para os SMR.

Outro exemplo de classes entidade usadas no modelo definido são as de suporte às rotas, apresentadas no Bloco 6.9.

Para além das classes entidade, outro código incluído no ficheiro "models.py"foi o apresentado no Bloco 6.10 que permite definir a estrutura que relaciona duas classes entidade, com uma relação muitos-para-muitos, e cria a respetiva tabela na base de dados.

6.4.6 Mecanismo de controlo de alterações da base de dados

O Alembic é um projeto que disponibiliza ferramentas que auxiliam o controlo da criação e alterações de estrutura ou dados de uma base de dados. Como o desenvolvimento da API do SGC em Python utilizou a *framework* Flask, foi também utilizado o módulo Flask-Migrate que é uma extensão do Alembic para SQLAlchemy. Com este módulo podemos versionar, atualizar e reverter alterações à estrutura ou dados da base de dados de forma documentada.

O processo começa pela criação do ambiente de migração, com a criação e execução de uma aplicação responsável pela migração da base de dados. Apresenta-se no Bloco 6.11 o código da aplicação criada para a gestão das migrações do SGC.

A criação do ambiente de migração é conseguida após ser executado o comando: **python manage.py db init**

```
from main import db
from sqlalchemy.dialects import postgresql
import datetime
import uuid

class BaseModel(db.Model):
    """Base data model for all objects"""
    __abstract__ = True

    estado = db.Column(postgresql.ENUM('ON', 'OFF', 'DEL', name='
    estado_enum'))

    def __init__(self, *args):
        super().__init__(*args)

    def __repr__(self):
        return '%s(%s)' % (self.__class__.__name__, {
            column: value
            for column, value in self._to_dict().items()
        })

    def json(self):
        return {
            column: value if not isinstance(value, datetime.date) else
    value.strftime('%Y-%m-%d')
            for column, value in self._to_dict().items()
        }
```

Listing 6.7: Classe base das classes entidade mapeadas pelo ORM.

A execução do comando anterior, que só é feita uma vez, cria um diretório chamado migrations com os ficheiros alembic.ini e env.py e outros sub-diretórios que irão receber futuras migrações.

Para criar a estrutura de suporte ao script de alterações de migração temos de executar: **python manage.py db migrate**

Depois de alterar a mensagem no início do ficheiro execute-se o comando: **python manage.py db upgrade**

Este último comando aplica na base de dados as alterações definidas pelos ficheiros de migração e atualiza a versão da migração na tabela **alembic_version** criada na base de dados para este efeito. Esta tabela pode ser vista na Figura 5.9 do modelo de dados ER no Capítulo 5.

6.4.7 Gestão de Versões

Para a gestão de versões de código foram surgindo ao longo dos anos diversos serviços, dos quais se destacam o CVS e o SVN. Linus Torvalds, o criador do *kernel* Linux, necessitava utilizar um destes sistemas (o BitKeeper) para organizar as alterações de código do *kernel* partilhadas por diversos programadores espalhados pelo mundo. Em 2005, Linus Torvalds decidiu criar o seu próprio sistema de controlo de versões, o Git, por não concordar com o tipo de licenciamento, estrutura e modo de funcionamento das alternativas existentes.

```
class SMR(BaseModel, db.Model):
    """Model for the SMR table"""
    __tablename__ = 'smrs'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
    entidade_id = db.Column(db.BigInteger, db.ForeignKey('entidades.id')
    )
    marca = db.Column(db.String(120))
    modelo = db.Column(db.String(120))
    serial_uuid = db.Column(db.String(64), index=True, unique=True,
    nullable=False)
    wifi = db.Column(db.Boolean, server_default='True', nullable=False)
    gprs = db.Column(db.Boolean, server_default='False', nullable=False)
    gps = db.Column(db.Boolean, server_default='False', nullable=False)
    tipo_sensor = db.Column(postgresql.ENUM('PESO', 'DISTANCIA', 'BOIA',
     name='tipo_sensor_enum'))
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())
    configs = db.relationship('SMRConfig', backref='smrs', lazy='dynamic
    ')
    reports = db.relationship('SMRReport', backref='smrs', lazy='dynamic
    ')
    updates = db.relationship('SMRUpdate', backref='smrs', lazy='dynamic
    ')

    def __init__(self, entidade_id, marca, modelo, serial_uuid, wifi=
    True, gprs=False, gps=False, tipo_sensor='PESO', estado='ON'):
        self.public_id = uuid.uuid4()
        self.entidade_id = entidade_id
        self.marca = marca
        self.modelo = modelo
        self.serial_uuid = serial_uuid
        self.wifi = wifi
        self.gprs = gprs
        self.gps = gps
        self.tipo_sensor = tipo_sensor
        self.estado = estado
```

Listing 6.8: Classes entidade relacionadas com o SMR.

```
class Rota(BaseModel, db.Model):
    """Model for the Routes table"""
    __tablename__ = 'rotas'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
    porigem_coords_lat = db.Column(db.Float)
    porigem_coords_lng = db.Column(db.Float)
    pdestino_coords_lat = db.Column(db.Float)
    pdestino_coords_lng = db.Column(db.Float)
    waypoint_order = db.Column(db.String(120))  # exemplo: [ 4, 1, 3, 2,
    0 ]
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())
    pontos = db.relationship('RotaPonto', backref='rotas', lazy='dynamic
    ')

    def __init__(self, porigem_coords_lat, porigem_coords_lng,
    pdestino_coords_lat, pdestino_coords_lng, estado='ON'):
        self.public_id = uuid.uuid4()
        self.porigem_coords_lat = porigem_coords_lat
        self.porigem_coords_lng = porigem_coords_lng
        self.pdestino_coords_lat = pdestino_coords_lat
        self.pdestino_coords_lng = pdestino_coords_lng
        self.estado = estado

    def setWaypoint_order(self, waypoint_order):
        self.waypoint_order = waypoint_order

class RotaPonto(BaseModel, db.Model):
    """Model for the Routes places table"""
    __tablename__ = 'rotas_pontos'

    id = db.Column(db.BigInteger, primary_key=True)
    rota_id = db.Column(db.BigInteger, db.ForeignKey('rotas.id'))
    waypoint_ini = db.Column(db.Integer)
    waypoint_pos = db.Column(db.Integer)
    precolha_nome = db.Column(db.String(120), index=True)
    precolha_contacto_nome = db.Column(db.String(120))
    precolha_contacto_telef = db.Column(db.String(20))
    precolha_coords_lat = db.Column(db.Float)
    precolha_coords_lng = db.Column(db.Float)
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    def __init__(self, rota_id, waypoint_ini, precolha_nome,
    precolha_contacto_nome, precolha_contacto_telef, precolha_coords_lat,
     precolha_coords_lng, estado='ON'):
        self.rota_id = rota_id
        self.waypoint_ini = waypoint_ini
        self.precolha_nome = precolha_nome
        self.precolha_contacto_nome = precolha_contacto_nome
        self.precolha_contacto_telef = precolha_contacto_telef
        self.precolha_coords_lat = precolha_coords_lat
        self.precolha_coords_lng = precolha_coords_lng
        self.estado = estado

    def setWaypoint_pos(self, waypoint_pos):
        self.waypoint_pos = waypoint_pos
```

Listing 6.9: Classes entidade relacionadas com as Rotas.

```
# many-to-many
db.Table('users_groups',
          db.Column('user_id', db.BigInteger, db.ForeignKey('users.id')),
          db.Column('group_id', db.BigInteger, db.ForeignKey('groups.id')
    )
          )
```

Listing 6.10: Código para relacionamento de classes entidade.

```
from flask_script import Manager
from flask_migrate import Migrate, MigrateCommand
from main import app, db

manager = Manager(app)
migrate = Migrate(app, db)

manager.add_command('db', MigrateCommand)

if __name__ == '__main__':
    manager.run()
```

Listing 6.11: Aplicação Python de manutenção da migração da base de dados.

O Git é atualmente a referência neste tipo de serviços e foi o gestor de versionamento escolhido por este projeto.

Git

Caso se pretenda um repositório com controlo de versões Git público, para partilhar com vários utilizadores, os serviços Bitbucket e GitHub são uma solução simples, mas com algumas limitações no que toca à privacidade dos projetos. Estes serviços, na sua opção gratuita, limitam a utilização dos repositórios a um número máximo de utilizadores ou apenas alojam os projetos de forma pública. Essas limitações fazem com que se tenha de optar pelos serviços pagos. Caso se pretenda controlar o versionamento de projetos sem essas limitações pode-se configurar um repositório Git em um servidor Linux sobre nosso controlo. Desta forma é possível configurar o número de utilizadores que pretendemos e controlar o seu nível de acesso aos diferentes projetos alojados. Outra vantagem é que este serviço pode ficar apenas acessível a partir da rede interna utilizando uma partilha NFS, acesso por SSH ou externamente via um serviço de VPN.

Para configurar o acesso a um repositório Git é necessário criar um grupo de utilizadores e adicionar todos os utilizadores, que se pretenda dar acesso, a esse grupo.

```
# Criar um grupo para os utilizadores do repo. git
sudo groupadd gitrepo

# adicionar o(s) users ao novo grupo
sudo adduser nunolima gitrepo
sudo adduser sofia gitrepo
```

Para criar um repositório que pode ser montado remotamente é necessário usar os parâmetros –bare e –shared que define as permissões de acesso dos utilizadores, grupos e outros (no mínimo 660). O nome do diretório, por convenção, termina em ".git".

```
git init --bare --shared=0660 gitrepo-proj01.git
```

Para um cliente utilizar um repositório terá que criar uma cópia local do projeto executando um dos seguintes comandos:

```
# para acesso local
git clone file:////home/nunolima/dev/gitrepo-proj01.git nome_dir_do_proj

# para acesso remoto
git clone nunolima@<SERVER_IP>:/home/nunolima/dev/gitrepo-proj01.git
    nome_dir_do_proj
```

Em caso de falha no sincronismo do clone deve-se executar o comando:

```
git branch --unset-upstream
```

Caso o clone do projeto seja bem sucedido pode-se efetuar o primeiro commit com os seguintes comandos:

```
cd nome_dir_do_proj
vim README
git status
git add .
git commit -m "Criado o ficheiro README do projeto"
```

Após se ter feito todos os *commits* (locais) pretendidos deve-se atualizar o projeto no repositório servidor.

```
git remote
git push origin master
```

O primeiro comando lista os nomes dos servidores remotos conhecidos pelo git local e o segundo comando atualiza o repositório no servidor onde "*origin*"é o nome usado por omissão para o *branch* principal sendo o *master* o *branch* local.

O comando anterior pode falhar caso se tenha alterado as mesmas zonas de código que outro programador, para corrigir esse problema é possível atualizar o nosso repositório local corrigindo os conflitos encontrados e procedendo a um *merge* com a versão no servidor.

É possível atualizar o projeto local tentando aplicar um *merge* automático com a última versão existente no servidor com o comando "*push*". Se não for possível efetuar o *merge* automático deve-se utilizar o comando *fetch* para atualizar a versão local e posteriormente resolver os conflitos de código.

```
import unittest
from os import environ

class TestStringMethods(unittest.TestCase):

def test_env_settings(self):
self.assertFalse(environ.get('SGC_API_SETTINGS') is None)

def test_env_database_url(self):
self.assertTrue(environ.get('DATABASE_URL') is not None)

if __name__ == '__main__':
unittest.main()
```

Listing 6.12: Exemplo de utilização de unittest

```
git pull origin master
git fetch origin updates-do-servidor
```

Os comandos de consola (bash) aqui apresentados são alguns dos mais utilizados durante o processo de versionamento de código, independentemente do repositório estar num servidor privado ou nos servidores online do Bitbucket ou GitHub.

Bitbucket

Por questões de simplicidade e por podermos utilizar uma conta académica, que permite manter o código dos projetos em repositório privado, foi escolhido o Bitbucket como servidor remoto de repositórios Git. Este repositório foi usado para versionar o desenvolvimento do SGC. Para permitir uma comunicação mais simples, sem a constante introdução de login e password a cada "git push", e para tornar mais segura essa comunicação, entre o posto de trabalho do programador e o Bitbucket, foi configurada o acesso ao BitBucket por chave cifrada SSH.

6.4.8 Testes Unitários e de Cobertura de Código

Para automatizar o processo de testes aplicados ao código desenvolvido para a API do SGC recorremos à utilização do módulo unittest do Python que permite agilizar a aplicação de testes unitários.

O exemplo de código de testes unitários aqui apresentado, avalia a existência de variáveis de ambiente necessárias ao funcionamento da API REST desenvolvida.

Em caso de obtermos sucesso em todos os testes é devolvido o resultado que se mostra na Figura 6.20. A Figura 6.21 apresenta o resultado em caso de um dos testes falhar, no caso o teste que verifica se a variável de ambiente DATABASE_URL está definida no sistema.

Outro módulo Python analisado foi o Coverage.py que permite saber a percentagem de código coberta pelos testes unitários. Isto é feito através do controlo de quais partes do código são executadas em comparação com todo o código disponível.

Figura 6.20: Passagem positiva de todos os testes unitários.

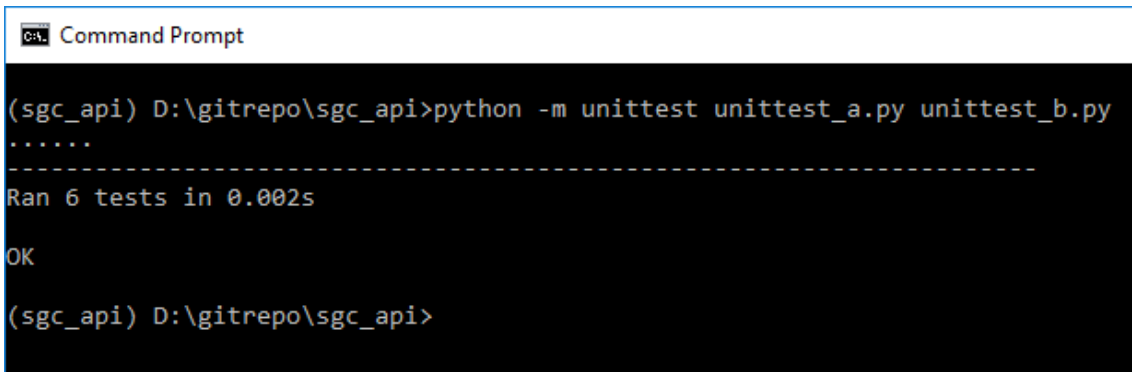
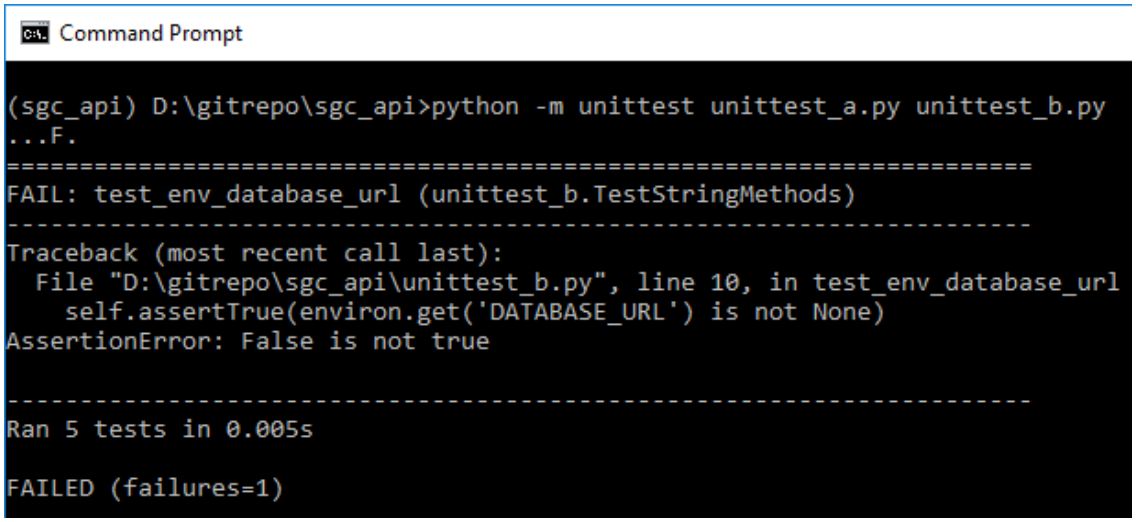


Figura 6.21: Exemplo de falha dos testes unitários.



O resultado dos testes de cobertura permite avaliar a eficácia dos testes unitários aplicados, sendo desejável obter valores de cobertura de código superiores a 75%.

6.5 Instalação e Configuração do Servidor

Para dar suporte à estrutura aplicacional do SGC foi adquirido um computador com CPU Intel i5 M 460, 8GB de memória e disco *Solid-State Drive* (SSD) de 128GB, onde foi instalado o Sistema Operativo (SO) Linux Ubuntu Server 16.04. Esta versão do SO foi escolhida por se tratar de uma *Long Term Support* (LTS), que garante um período de suporte de 5 anos, sendo lançada nova versão LTS de 2 em 2 anos.

6.5.1 OpenVPN

Para possibilitar acesso a partir do exterior à LAN foi usado um RaspberryPi onde foi instalado o serviço OpenVPN. Por questões de segurança pretendemos que este serviço não ficasse

instalado no servidor aplicacional. Como o router do nosso ISP não ter funcionalidades de VPN, foi necessário encontrar um equipamento económico, em termos de aquisição e de consumo energético.

Para estabelecer uma rede VPN os protocolos L2TP/IPsec e PPTP são os mais utilizados por ser parte integrante da maior parte dos sistema operativos mais utilizados e também por ser rápido e de simples configuração. No entanto, foram identificadas várias falhas de segurança nestes protocolos. Uma alternativa mais segura, apesar de ser mais complexa a sua configuração, é o OpenVPN. Apesar dos sistemas cliente suportados pelo OpenVPN ser mais reduzido, isto não se revelou um problema por existirem clientes para todos os sistemas usados neste projeto para estabelecer ligações à rede de servidores usada. O servidor OpenVPN foi inicialmente instalado num equipamento RaspberryPi, utilizando o projeto PiVPN que facilita a configuração deste serviço, com o intuito de efetuar testes às comunicações. Após estes testes terem sido bem sucedidos, este serviço foi instalado no servidor de produção usado pelo Smart Depot que corre um Linux Ubuntu Server 16.04 LTS. A infraestrutura de comunicações para o acesso à Internet usada pelos servidores usa numa linha de fibra óptica com taxas de upload e download de 100/100 Mbps.

Como já mencionado antes o cliente OpenVPN não é nativo da maioria dos sistemas operativos mais utilizados. Foram usados clientes OpenVPN para Windows 64-bit e para Android, tendo sido ambos sido adicionados ao sistema operativo. Também foram configurados e utilizados acessos a partir de máquinas Linux onde não foi necessária a instalação do cliente OpenVPN por este SO já suportar este tipo de ligações de forma nativa. Para além do software cliente, a configuração de uma ligação OpenVPN requer a partilha de um ficheiro com extensão .ovpn que contêm informação do servidor OpenVPN, como o seu IP, protocolos usados e chaves criptográficas.

6.5.2 Secure Shell (SSH)

De forma a ser possível efetuar uma ligação remota segura ao servidor, foi instalado o serviço OpenSSH e configuradas algumas restrições de segurança:

- Apenas usar o protocolo SSH na sua versão 2;
- Limitar o acesso a determinados utilizadores;
- Não permitir o acesso (direto) com o utilizador root;
- Alterar a porta normal do serviço (22) para outra, p.ex. 2222;
- Restringir a origem dos acessos a hosts na mesma LAN, na firewall (iptables);
- Habilitar o acesso apenas via chaves cifradas (keychain based authentication);
- Bloquear os endereços IP usados em tentativas falhadas de ligação (DenyHosts).

Como forma de segurança, para além das configurações específicas de cada serviço instalado, são feitas atualizações regulares e consultados os logs.

6.5.3 HTTPS com SSL gratuito

A Let's Encrypt é uma CA onde era possível gerar certificados SSL de confiança de forma gratuita. Após a criação do certificado SSL para o domínio "smartdepot.duckdns.org" o mesmo foi configurado no Nginx do nosso servidor e foi também criado um *bash script*, chamado periodicamente via crontab, que mantém o certificado SSL ativo.

6.5.4 Nginx e Web Server Gateway Interface (WSGI)

Como servidor web foi usado o Nginx por este se apresentar como uma alternativa de alta performance e escalabilidade em comparação com o outro serviço concorrente à liderança de mercado, o Apache. O Nginx em termos de configuração assemelha-se ao Apache, utilizando também *virtual hosts* que são configurados em ficheiros localizados no diretório "sites-available" de onde são feitos links simbólicos para "sites-enable". Outro fator que levou à escolha do Nginx foi a sua boa integração com o serviço de *gateway* WSGI utilizado para executar aplicações desenvolvidas na linguagem Python.

6.5.5 DuckDNS

O serviço de dynamic DNS da DuckDNS foi utilizado para criar e associar o nome de domínio "smartdepot.duckdns.org" ao IP público dinâmico atribuído pelo nosso ISP. Este serviço é gratuito e permite manter atualizado o IP associado a este nome de domínio. Outra opção seria registar um novo domínio, .COM, .ORG ou .PT, junto de um *Domain Name Registrar*, efetuando o pagamento de uma anuidade associada.

6.5.6 PostgreSQL

O SGC a desenvolver necessita persistir a informação e para isso foi instalado o SGBD PostgreSQL por permitir a escrita de procedimentos, funções e *triggers* internos escritas em PL/pgSQL.

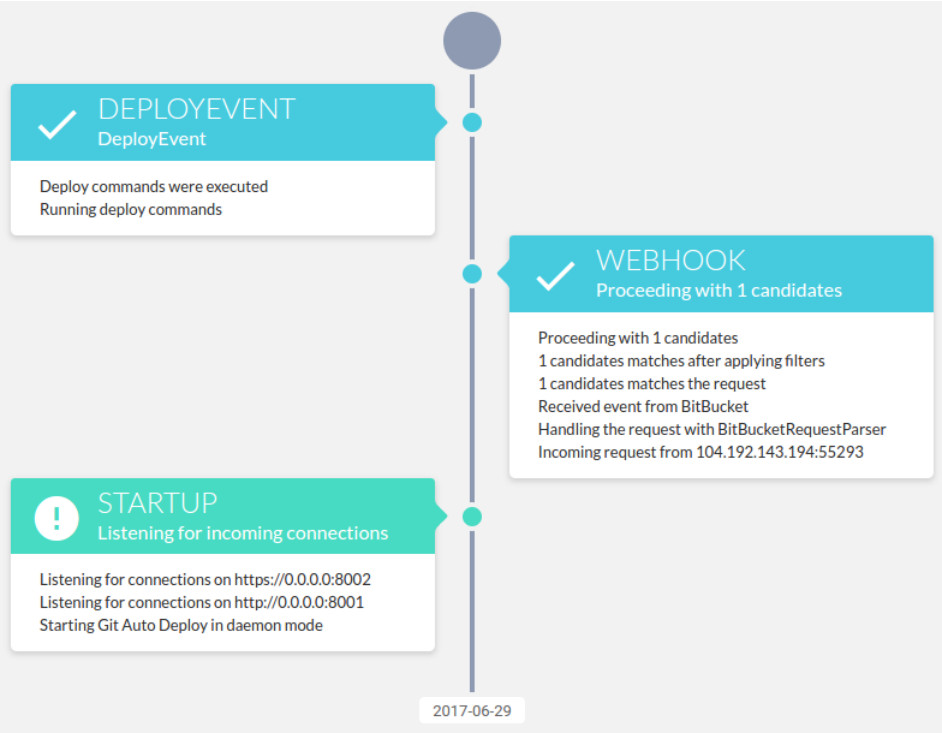
Levar para dentro da base de dados alguma da lógica de negócio, desenvolvendo procedimentos internos, tem como desvantagem o acoplamento (a esta base de dados), no entanto traz uma grande vantagem de performance quando o volume de dados é grande, como esperamos vir a ser quando milhares de SMR estiverem a enviar os seus relatórios de leituras que precisam ser tratados para recolher a informação.

6.5.7 Git Auto Deploy (GAD)

Para ser possível aplicar uma abordagem de "Continuous Deployment" durante o desenvolvimento do SGC foi instalado no servidor o serviço GAD que aguarda um *request* enviado pelo servidor Git quando de um *push* do programador. O servidor Git usado pelo Smart Depot para o versionar o SGC foi o Bitbucket sendo configurado neste o WebHook que dispara o *request* quando é detetado um *push* no repositório.

O serviço GAD foi configurado para executar *scripts* de *bash* antes e depois da instalação da versão mais recente. Algumas das tarefas desses *scripts* de *bash* são parar serviços

Figura 6.22: GAD.



como Nginx e WSGI, remover diretórios e ficheiros do ambiente virtual Python (virtualenv), redefinir variáveis de sistema e reiniciar serviços. Na Figura 6.22 pode-se observar a informação disponibilizada, pela página de administração do serviço GAD, após ser reportada a existência de uma nova versão no repositório.

O serviço GAD quando recebe o *request*, aplica os *scripts* de pré e pós *fetch* que foram escritos para possibilitar a atualização correta do SGC que executa em produção e os necessários reinícios ou *reload* de configurações de serviços como Nginx e WSGI.

A *Continuous Delivery* é o conjunto de procedimentos aplicados ao código desenvolvido que automatizam a instalação do mesmo em ambientes de teste semelhantes aos de produção para que sejam aplicados testes unitários e de cobertura ao código desenvolvido.

Para definir e aplicar testes ao código desenvolvido em Python foi escolhida a ferramenta de testes unitários unittest, também conhecida como pyUnit. Este *framework* de testes unitários funciona com base na escrita de classes e métodos de testes que despoletam avisos utilizando *asserts* quando os métodos testados falham.

Como ferramenta de medição da cobertura do código testado o Python dispõe do módulo Coverage.py (Batchelder, 2009) que monitoriza a aplicação identificando que partes deste são executadas e depois analisa o código para identificar zonas que poderiam ter sido executadas e não foram. Com esta ferramenta pode-se medir a eficácia dos testes desenvolvidos, identificando que partes do código estão ou não a ser testadas.

O passo seguinte é o *Continuous Deployment* que automatiza o processo de instalação das versões de código que passaram os testes nos servidores de produção. Normalmente fazem também parte do processo de *Continuous Deployment* a execução de *scripts* que param e reiniciam serviços como servidores de base de dados ou web, alteração de permissões

de execução e acesso a ficheiros, de cópia de segurança ou reconfiguração de critérios de segurança e alarmística.

Capítulo 7

Testes e Resultados

Os procedimentos de teste aplicados ao protótipo SMR e respetivos resultados são descritos neste capítulo.

7.1 Testes

Este projeto desenvolveu um dispositivo SMR automático capaz de substituir a intervenção humana na leitura do nível de fluido presente em depósitos de recolha de óleos usados.

Existir um equipamento capaz de efetuar a leitura do nível de um depósito de óleo usado de forma automática, com exatidão e fornecer essa informação remotamente com uma periodicidade diária (ou regulável) faz com que as empresas de recolha deste tipo de resíduos deixem de depender das leituras visuais e notificações feitas pelo produtor de resíduo.

O protótipo SMR construído no âmbito deste projeto foi avaliado quanto à exatidão (*accuracy*) das suas leituras, dada a necessidade de aferir o grau de confiança dos valores recolhidos.

Tendo o objetivo de recolher o valor o mais exato possível cada leitura reportada é o valor médio de cinco leituras consecutivas. Poderia também ser utilizado o calculo da mediana de um conjunto de valores lidos, ou mesmo de médias, para minimizar a ocorrência de *outliers* contudo o tipo de sensores usados, de carga por tensão, não apresenta este problema.

Durante os processos de aferição dos sensores de carga por tensão e de teste das leituras de nível de fluido obtidas pelos protótipos, o protótipo de controlo e o de teste, foram utilizados valores de referência em ml (mililitros) medidos com o jarro volumétrico graduado apresentado na Figura 7.1.

7.1.1 Grandezas a Avaliar

A grandeza que este teste pretendeu testar é a exatidão das leituras do nível de fluido, sendo feita a comparação entre os valores lidos pelo SMR e os lidos visualmente junto ao depósito de óleo pela pessoa responsável pelo ponto de recolha.

Figura 7.1: Jarro volumétrico graduado em ml.

7.1.2 Hipóteses a Testar

A hipótese nula (7.1), que se pretende rejeitar, é que as médias dos resultados são iguais, ou seja, os dois tipos de leitura obtêm os mesmos resultados para um erro inferior a $\alpha \leq 5\%$.

$$h0 : \mu 1 = \mu 2 \tag{7.1}$$

$$h1 : \mu 1 < \mu 2 \tag{7.2}$$

Caso se confirme que há diferenças entre os dois grupos, será possível rejeitar a hipótese nula, assumindo H1 (7.2) como verdadeira.

7.1.3 Metodologia de Avaliação

Foi escolhido um ponto de recolha, onde foi instalado um SMR a recolher leituras do nível de fluido num depósito com capacidade de 20L, em polietileno translúcido onde foi marcada uma escala graduada, com resolução de 500 ml.

Pediu-se que à pessoa responsável pelo ponto de recolha efetuasse o registo da quantidade depositada, sempre que fosse acrescentado óleo ao depósito.

O teste teve a duração de 30 dias, tendo decorrido entre 16 de Agosto e 14 de Setembro. Durante o período de teste foram entregues no ponto de recolha 8 embalagens com diferentes volumes de óleo usado, estes foram o grupo de controlo. Todos os volumes entregues foram medidos utilizando um jarro volumétrico graduado em mililitros.

Tendo sido obtidos os resultados apresentados na Tabela 7.1.

Tabela 7.1: Teste SMR

Dias	Óleo (controlo ml) entregue	Óleo (controlo ml) recolhido	Leituras SMR	% sucesso	Leituras visuais	% sucesso
1		0,000	0,000	100,0	0	100,0
2		0,000	0,000	100,0	0	100,0
3		0,000	0,000	100,0	0	100,0
4		0,000	0,000	100,0	0	100,0
5		0,000	0,000	100,0	0	100,0
6	2,900	2,900	2,900	100,0	3	96,6
7		2,900	2,902	99,9	3	96,6
8		2,900	2,903	99,9	3	96,6
9	2,500	5,400	5,404	99,9	5	86,2
10		5,400	5,400	100,0	5	86,2
11		5,400	5,401	100,0	5	86,2
12		5,400	5,402	100,0	5	86,2
13		5,400	5,400	100,0	5	86,2
14	2,400	7,800	7,808	99,9	8	93,1
15		7,800	7,805	99,9	8	93,1
16		7,800	7,806	99,9	8	93,1
17	1,200	9,000	8,998	100,0	8	65,5
18		9,000	9,001	100,0	8	65,5
19		9,000	8,999	100,0	8	65,5
20		9,000	9,000	100,0	8	65,5
21	0,500	9,500	9,498	100,0	8	48,3
22		9,500	9,499	100,0	8	48,3
23		9,500	9,500	100,0	8	48,3
24	2,800	12,300	12,301	100,0	12	89,7
25		12,300	12,300	100,0	12	89,7
26		12,300	12,297	100,0	12	89,7
27	2,800	15,100	15,109	99,9	15	96,6
28		15,100	15,103	100,0	15	96,6
29		15,100	15,105	100,0	15	96,6
30	1,800	16,900	16,894	100,0	17	96,6

Os valores lidos por ambas as amostras foram usados para calcular o desvio, módulo da diferença entre o nível de controlo e o valor lido por cada um dos grupos. Utilizando o valor de controlo, o desvio e a regra de três simples, calculamos a percentagem de sucesso (7.3) para cada leitura. Uma percentagem de sucesso de cem por cento corresponde a uma leitura exata, igual à do valor de controlo.

$$x = (ValorControlo - |ValorControlo - ValorLido|) * 100/ValorControlo \quad (7.3)$$

A média dos 30 valores de cada grupo foi usada para determinar qual mais se aproximava do valor de real, com uma significância de $\alpha \leq 5\%$.

7.1.4 Teste de Hipóteses

Para testar os dados lidos pelo protótipo em comparação com os valores observados pelo funcionário no local, poderia ser utilizado o teste estatístico de *Wilcoxon signed-rank test*, mas como as amostras são de 30 leituras cada, pode-se assumir que seguem uma distribuição normal, o que significa que devemos usar um teste paramétrico.

TEndo sido as leituras feitas sobre os mesmos valores de controlo, consideram-se emparelhadas.

Para testar duas amostras emparelhadas e que seguem uma distribuição normal (paramétricas) o teste recomendado é o *Student's t-test*.

7.2 Resultados

Os valores apresentados na Tabela 7.1 foram obtidos durante o teste feito ao protótipo SMR que foi configurado para recolher leituras a cada vinte e quatro horas. Essas leituras foram comparadas com as feitas visualmente pela pessoa responsável pelo ponto de recolha.

Os resultados demonstram que existe uma maior exatidão nas leituras recolhidas pelo SMR em comparação com as feitas visualmente. Outra situação que também se verificou foi a falha no registo de duas das entregas por parte da pessoa responsável pelas leituras visuais. O sensor de carga por tensão utilizado pelo SMR foi configurado para devolver o valor médio de cinco leituras de forma a se obter o valor mais próximo do real.

Ao comparar os registadas das duas amostras do teste verifica-se que, nas entregas dos dias 17 e 21, a pessoa responsável pela leitura visual não fez qualquer registo, por considerar que não existiam alterações ou por esquecimento.

Constata-se que o desvio das leituras visuais, relativamente aos valores de controlo, varia entre 100 ml e 400 ml, se considerarmos ter ocorrido esquecimento no registo de duas das entregas. Nas leituras diariamente feitas pelo SMR exitem desvios que variam entre 1 e 9 ml em comparação com os volumes de controlo.

Conclui-se que, com a utilização do SMR, foram identificadas todas as entregas e que os valores lidos são mais exatos.

Os resultados obtidos foram de encontro às expectativas já que o SMR é capaz de recolher leituras exatas, para uma resolução de 10 ml, enquanto que a escala visual usada no depósito tinha uma resolução de 500 ml.

Capítulo 8

Conclusão

Neste capítulo são apresentados os objetivos atingidos e o qual o trabalho futuro necessário para dar continuidade a este projeto.

8.1 Objetivos Alcançados

Foram atingidos todos os objetivos propostos por este projeto, tendo sido ainda acrescentado um novo objetivo no decorrer do mesmo.

Como inicialmente proposto, foi construído um equipamento, ao qual se deu o nome de Sistema de Monitorização Remota (SMR), com a capacidade de medir o nível de fluido presente num depósito de 50 Litros e de reportar remotamente essa informação para um *web service online* na Internet com uma periodicidade pré-configurada;

Foi também desenvolvida a plataforma online, chamada de Sistema de Gestão Centralizado (SGC), para a gestão de contas de clientes, utilizadores e grupos, com a capacidade para receber os dados reportados pelos SMR, para responder a pedidos de estado dos depósitos (nível de fluido) e dos sensores (estado da bateria e configuração) e para efetuar a otimização de rotas de recolha. O conjunto de *web services* disponibilizado pela API REST desenvolvida no SGC foi devidamente documentado, utilizando a especificação OpenAPI, onde são descritos todos os métodos disponíveis para cada rota criada, os códigos de resposta possíveis e qual a estrutura de dados esperada e obtida em cada um deles, incluindo também exemplos JSON.

Para além dos objetivos inicialmente propostos já enumerados, identificou-se a necessidade em desenvolver uma Aplicação Móvel Android (AMA) que desse suporte à configuração dos SMR feita pelos funcionários das empresas de recolha. Esta necessidade foi identificada por não ser prática a introdução manual do *Token*, das coordenadas (latitude e longitude) dos pontos de recolha e dos nomes das redes Wi-Fi a ser enviadas na configuração de cada SMR.

O protótipo SMR, desenvolvido no âmbito deste projeto, oferece às empresas de recolha uma solução inteligente de monitorização remota do nível dos seus depósitos de óleo usado, resolvendo desta forma um dos problemas centrais destas empresas.

O SGC soluciona outro dos problemas, colocados as empresas de recolha de óleos usados, ao fornecer a lista ordenada dos pontos de recolha a ser visitados em cada rota.

8.1.1 Outras Aplicações da Solução Smart Depot

Para além da reciclagem, existem muitas outras áreas de mercado onde o SMR pode ser aplicado com extrema utilidade, contribuindo para o aumento de vendas, evitando rutura de stock, alertando atempadamente para a necessidade de intervenções de prevenção, fornecimento ou de assistência. Seguem-se alguns exemplos:

- Esvaziamento de fossas séticas;
- Reabastecimento de água potável nas empresas;
- Reabastecimento de oxigénio ao domicílio;
- Reabastecimento de gás butano e propano ao domicílio;
- Reposição do stock de tintas e diluentes;
- Reposição do stock de detergentes e líquidos de limpeza;
- Reposição do stock de bebidas e óleos alimentares na restauração;
- Reposição do stock de combustíveis e óleos lubrificantes em oficinas e distribuidores.

8.1.2 Conhecimento Adquirido

Este trabalho permitiu estudar alguns dos problemas que envolvem a reciclagem de resíduos, mais concretamente dos óleos usados. Para o estudo deste mercado muito contribuiu a visita às instalações e reunião com os responsáveis da empresa de recolha de óleos alimentares usados Filtapor, que permitiu a observação do processo de recolha em ambiente real.

Outras fontes de informação importantes foram os estudos encontrados sobre a evolução e adesão à reciclagem e toda a documentação legal, nacional e europeia, aplicada a esta área, desde a distribuição de responsabilidades de reciclagem a produtores de óleos novos até à responsabilização dos municípios pela recolha de forma autónoma ou através da contratação de empresas especializadas.

Para além dos conhecimentos obtido no âmbito da reciclagem, foram ainda estudados conceitos e metodologias de estudo de mercado e produto, de viabilidade de negócio e do estado da arte de várias tecnologias aplicáveis ao projeto. Através deste estudo tornou-se possível a seleção das tecnologias mais adequadas a cada situação, quer com base nos requisitos, nas suas caraterísticas, ou mesmo, aplicando testes de hipóteses ou AHP. Relativamente aos componentes aplicáveis ao SMR foram alvo de estudo diferentes tecnologias usadas pelos sensores de nível de fluido e pelos módulos microcontroladores de baixo custo com capacidades de comunicação sem fios. No âmbito do SGC foram analisadas bibliotecas FOSS e serviços comerciais capazes de calcular boas soluções de rotas solucionando problemas de VRP.

Para permitir a escolha da melhor arquitetura a usar no projeto, foram elaboradas duas abordagens distintas, uma que integrava no SGC todos os serviços necessários e outra que tirava partido de serviços externos ao projeto. Optou-se pela segunda abordagem, ao tirar partido da API da Google para a otimização das rotas, mas sem inclusão do ESB proposto, já que se tinha o controlo de todo o restante código, do SMR e do SGC, interveniente na solução.

No aspeto tecnológico foram adquiridos conhecimentos técnicos de funcionamento de sensores e de MCU, tendo-se ainda trabalhado na área eletrónica, com a análise de circuitos e esquemas elétricos necessários para a construção dos protótipos SMR.

Relativamente a objetivos atingidos no âmbito do desenvolvimento de software foram desenvolvidas três aplicações, o *firmware* desenvolvido em C++ do SMR, a AMA desenvolvida em Java para Android e uma API REST em linguagem Python para o SGC.

Foram também implementados com sucesso, mecanismos de gestão de versões em Git, apoio ao desenho e análise de software com vários diagramas UML e Mockups, implementação de um sistema de *Continuous Deployment* para a API do SGC com o serviço Git-Auto-Deploy e Webhooks da Bitbucket, assim como a instalação e configuração de um servidor Linux com serviços vários entre eles de comunicação por canais seguros, SSH e OpenVPN, e servidores Web e de base de dados, Nginx e PostgreSQL respetivamente.

De destacar que para auxiliar a gestão do tempo, de tarefas e *millestones* recorreu-se a um software *online* de gestão de projetos da Soho.

8.2 Trabalho Futuro

Nesta secção são apresentados alguns dos pontos que necessitam de desenvolvimento futuro de forma a ser possível o lançamento do projeto Smart Depot no mercado, e com ele, ajudar a agilizar o processo de recolha de óleos usados, contribuindo assim, para a redução da pegada ambiental.

8.2.1 Financiamento do Projeto

Será importante procurar apoios financeiros, ou através de fontes governamentais Portuguesas ou Europeias, projetos de apoio ao desenvolvimento nas áreas ambientais ou mesmo iniciativas privadas. Só através de uma forte injeção de capital será possível lançar o projeto no mercado.

O capital será especialmente investido na promoção e marketing, junto das empresas de recolha de óleos usados, no desenho e construção dos moldes necessários para a produção em massa, na ordem dos 40 a 50 Mil unidades e também para a compra e montagem dos componentes eletrónicos.

8.2.2 Aplicação Web de Administração

Como trabalho futuro, propõe-se o desenvolvimento de uma aplicação web para administração do SGC que facilite a sua gestão através de uma interface simples e prática.

8.2.3 Enriquecimento da API

Será importante o contínuo enriquecimento da API com a adição de novos serviços que acrescentem as funcionalidades procuradas na área, através da identificação de oportunidades ou introduzindo ideias de serviços inovadores.

Bibliografia

Sousa, Marcelo Valle De (2010). «Análise de». Em: pp. 12–14.

Ministério do Ambiente (1997). *Diário da República n.º 208/1997 de 9 de Setembro do Ministério do Ambiente*. Portugal. url: www.dre.pt.

Ministérios das Cidades, Ordenamento Do Território E Ambiente (2003). «Decreto-Lei n.º 153/2003, de 11 de julho». Em: *Diário da República 1ª Série-A* 158, pp. 3957–3965.

Território, Menistério Do Ambiente E Do Ordenamento Do (2011). «Decreto-Lei nº 73/2011 de 17 de Junho». Em: *Diário da República*, pp. 1–50.

Global, Uso et al. (2013). «Decreto-Lei n.º 75/2013. D.R. I Série. 107 - 04 de junho». Em: pp. 3218–3219.

Koen, Peter et al. (2001). «Providing Clarity and a Common Language To the 'Fuzzy Front End.'» Em: *Research Technology Management* 44.2, pp. 46–55. issn: 08956308. doi: Article. arXiv: /ehis.ebscohost.com/ [http:]. url: http://www.tandfonline.com/action/journalInformation?journalCode=urtm20http://dx.doi.org/10.1080/08956308.2001.11671418.

Lindner, Michael. *Oil Condition Monitoring Using Electrical Conductivity*. url: http://www.machinerylubrication.com/Read/29407/oil-condition-monitoring (acedido em 07/02/2017).

SOGILUB (2015). «SOGILUB - Relatório 2015». Em:

Recoil (2017). *Recoil Project*. url: https://www.recoilproject.eu/index.php/en/ (acedido em 25/02/2017).

Mcatalao (2013). «RECOIL - Best methods guiding principles». Em:

Alexander Osterwalder (2008). *The Business Model Canvas - Nonlinear Thinking*. url: http://nonlinearthinking.typepad.com/nonlinear{_}thinking/2008/07/the-business-model-canvas.html (acedido em 23/02/2017).

Saaty, Thomas L. (1990). «How to make a decision: The analytic hierarchy process». Em: *European Journal of Operational Research* 48.1, pp. 9–26. issn: 03772217. doi: 10.1016/0377-2217(90)90057-I.

UTAD (2012). *GreenBox*. url: http://greenbox.utad.pt/index.html (acedido em 16/02/2017).

Sogilub (2017). *Sogilub - Sogilub | Sogilub*. url: http://www.sogilub.pt/quem-somos/sogilub (acedido em 21/02/2017).

Link Labs (2016). *Selecting the right wireless technology for M2M*. url: https://www.link-labs.com/selecting-a-wireless-technology-for-industrial-iot (acedido em 22/02/2017).

Espressif Systems (2014). *ESP8266EX - Espressif Systems*. url: https://espressif.com/en/products/hardware/esp8266ex/overview (acedido em 21/02/2017).

Realtek Semiconductor Corp (2017). *Realtek*. url: http://www.realtek.com/ (acedido em 21/02/2017).

Eelecfans (2017). *Realtek RTL8710 vs Espressif ESP8266*. url: http://bbs.elecfans.com/jishu{_}715553{_}1{_}1.html (acedido em 21/02/2017).

Neoway Tech (2010). «Neo_M590 Hardware Design Manual». Em:

Ai Thinker (2017). *A6 - Ai Thinker*. url: http://en.ai-thinker.com/html/Products/GPRS{_}Module/A6/ (acedido em 21/02/2017).

SIMcom (2017). *SIMCom Wireless Solutions*. url: http://simcomm2m.com/En/ (acedido em 21/02/2017).

KING-GAGE (2013). «Tank Liquid Level». Em: url: http://www.king-gage.com/.

Electronics-tutorials (2017). *Hall Effect Sensor and How Magnets Make It Works*. url: http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html (acedido em 21/02/2017).

Direct Industry (2017). *Ultrasonic level sensor / for liquids / for tanks / non-contact - ECHOTEL® 335 - MAGNETROL*. url: http://www.directindustry.com/prod/magnetrol/product-6020-1633245.html (acedido em 21/02/2017).

PVL (2017). *Level sensors and water in oil sensors*. url: http://www.pvl.co.uk/level{_}sensors.html (acedido em 21/02/2017).

AL-MUTLAQ, SARAH. *Getting Started with Load Cells*. url: https://learn.sparkfun.com/tutorials/getting-started-with-load-cells (acedido em 22/09/2017).

Elsevier Science (Firm). *Theoretical computer science*. Elsevier Science.

The Clay Mathematics Institute (2017). *The Millennium Prize Problems | Clay Mathematics Institute*. url: http://www.claymath.org/millennium-problems/millennium-prize-problems (acedido em 16/02/2017).

Braekers, Kris; Ramaekers, Katrien; Nieuwenhuyse, Inneke Van (2016). «The vehicle routing problem: State of the art classification and review». Em: *Computers & Industrial Engineering* 99, pp. 300–313. issn: 0360-8352. doi: 10.1016/J.CIE.2015.12.007. url: http://www.sciencedirect.com/science/article/pii/S0360835215004775.

Dorigo, Marco e Luca Maria Gambardella (1997). «Ant colonies for the travelling salesman problem». Em: *Biosystems* 43.2, pp. 73–81. issn: 03032647. doi: 10.1016/S0303-2647(97)01708-5.

Monirian, Masoud Amel, Shabnam Mahmoudzadeh Vaziri e Asadollah Mahmoudzadeh Vaziri (2017). «Located Multiple Depots and Vehicles Routing with Capacity Problem». Em: *Proceedings of the Tenth International Conference on Management Science and Engineering Management*. Ed. por Jiuping Xu et al. Singapore: Springer Singapore, pp. 619–630. isbn: 978-981-10-1837-4. doi: 10.1007/978-981-10-1837-4_52. url: http://link.springer.com/10.1007/978-981-10-1837-4{_}52http://dx.doi.org/10.1007/978-981-10-1837-4{_}52.

Lim, A. e F. Wang (2005). «Multi-Depot Vehicle Routing Problem: A One-Stage Approach». Em: *IEEE Transactions on Automation Science and Engineering* 2.4, pp. 397–402. issn: 1545-5955. doi: 10.1109/TASE.2005.853472. url: http://ieeexplore.ieee.org/document/1514459/.

Jsprit (2017). *jsprit | java toolkit for rich VRPs and TSPs*. url: https://jsprit.github.io/ (acedido em 19/02/2017).

Red Hat (2017). *OptaPlanner - Constraint satisfaction solver (Java^{TM}, Open Source)*. url: https://www.optaplanner.org/{\#} (acedido em 19/02/2017).

Ted Ralphs (2017). *SYMPHONY VRP*. (Acedido em 19/02/2017).

Frightanic.com (2015). *NodeMCU custom builds*. url: https://nodemcu-build.com/ (acedido em 12/10/2017).

Fowler, Martin e Martin (2003). *Patterns of enterprise application architecture*. Addison-Wesley, p. 533. isbn: 0321127420.

OpenAPI (2017). *Swagger Editor*. url: https://editor.swagger.io/ (acedido em 16/10/2017).

Auth0 (2017). *JSON Web Tokens - jwt.io*. url: https://jwt.io/ (acedido em 16/10/2017).

Batchelder, Ned (2009). *Coverage.py — Coverage.py 4.4.1 documentation*. url: `https://coverage.readthedocs.io/en/coverage-4.4.1/` (acedido em 16/10/2017).

Apêndice A

Ficheiro models.py do SGC

Este apêndice apresenta o código Python desenvolvido para descrever a camada de modelo do SGC, usado pelo ORM SQLAlchemy no mapeamento do codigo OO com a base de dados.

```
from main import db
from sqlalchemy.dialects import postgresql
import datetime
import uuid

class BaseModel(db.Model):
    """Base data model for all objects"""
    __abstract__ = True

    estado = db.Column(postgresql.ENUM('ON', 'OFF', 'DEL', name='
    estado_enum'))

    def __init__(self, *args):
        super().__init__(*args)

    def __repr__(self):
        """Define a base way to print models"""
        return '%s(%s)' % (self.__class__.__name__, {
            column: value
            for column, value in self._to_dict().items()
        })

    def json(self):
        """
                Define a base way to jsonify models, dealing with
    datetime objects
        """
        return {
            column: value if not isinstance(value, datetime.date) else
    value.strftime('%Y-%m-%d')
            for column, value in self._to_dict().items()
        }


class Rota(BaseModel, db.Model):
    """Model for the Routes table"""
    __tablename__ = 'rotas'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
```

```
    porigem_coords_lat = db.Column(db.Float)
    porigem_coords_lng = db.Column(db.Float)
    pdestino_coords_lat = db.Column(db.Float)
    pdestino_coords_lng = db.Column(db.Float)
    waypoint_order = db.Column(db.String(120))  # exemplo: [ 4, 1, 3, 2,
    0 ]
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    pontos = db.relationship('RotaPonto', backref='rotas', lazy='dynamic
    ')


    def __init__(self, porigem_coords_lat, porigem_coords_lng,
    pdestino_coords_lat, pdestino_coords_lng, estado='ON'):
        self.public_id = uuid.uuid4()
        self.porigem_coords_lat = porigem_coords_lat
        self.porigem_coords_lng = porigem_coords_lng
        self.pdestino_coords_lat = pdestino_coords_lat
        self.pdestino_coords_lng = pdestino_coords_lng
        self.estado = estado


    def setWaypoint_order(self, waypoint_order):
        self.waypoint_order = waypoint_order


class RotaPonto(BaseModel, db.Model):
    """Model for the Routes places table"""
    __tablename__ = 'rotas_pontos'

    id = db.Column(db.BigInteger, primary_key=True)
    rota_id = db.Column(db.BigInteger, db.ForeignKey('rotas.id'))

    waypoint_ini = db.Column(db.Integer)
    waypoint_pos = db.Column(db.Integer)

    precolha_nome = db.Column(db.String(120), index=True)
    precolha_contacto_nome = db.Column(db.String(120))
    precolha_contacto_telef = db.Column(db.String(20))

    precolha_coords_lat = db.Column(db.Float)
    precolha_coords_lng = db.Column(db.Float)

    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())


    def __init__(self, rota_id, waypoint_ini, precolha_nome,
    precolha_contacto_nome, precolha_contacto_telef, precolha_coords_lat,
     precolha_coords_lng, estado='ON'):
        self.rota_id = rota_id
        self.waypoint_ini = waypoint_ini
        self.precolha_nome = precolha_nome
        self.precolha_contacto_nome = precolha_contacto_nome
        self.precolha_contacto_telef = precolha_contacto_telef
        self.precolha_coords_lat = precolha_coords_lat
        self.precolha_coords_lng = precolha_coords_lng
        self.estado = estado
```

```
    def setWaypoint_pos(self, waypoint_pos):
        self.waypoint_pos = waypoint_pos


class SMR(BaseModel, db.Model):
    """Model for the SMR table"""
    __tablename__ = 'smrs'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)

    entidade_id = db.Column(db.BigInteger, db.ForeignKey('entidades.id')
    )

    marca = db.Column(db.String(120))
    modelo = db.Column(db.String(120))
    serial_uuid = db.Column(db.String(64), index=True, unique=True,
    nullable=False)

    wifi = db.Column(db.Boolean, server_default='True', nullable=False)
    gprs = db.Column(db.Boolean, server_default='False', nullable=False)
    gps = db.Column(db.Boolean, server_default='False', nullable=False)

    tipo_sensor = db.Column(postgresql.ENUM('PESO', 'DISTANCIA', 'BOIA',
    name='tipo_sensor_enum'))
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    configs = db.relationship('SMRConfig', backref='smrs', lazy='dynamic
    ')
    reports = db.relationship('SMRReport', backref='smrs', lazy='dynamic
    ')
    updates = db.relationship('SMRUpdate', backref='smrs', lazy='dynamic
    ')


    def __init__(self, entidade_id, marca, modelo, serial_uuid, wifi=
    True, gprs=False, gps=False, tipo_sensor='PESO', estado='ON'):
        self.public_id = uuid.uuid4()
        self.entidade_id = entidade_id
        self.marca = marca
        self.modelo = modelo
        self.serial_uuid = serial_uuid
        self.wifi = wifi
        self.gprs = gprs
        self.gps = gps
        self.tipo_sensor = tipo_sensor
        self.estado = estado


class SMRConfig(BaseModel, db.Model):
    """Model for the SMR Reports table"""
    __tablename__ = 'smr_configs'

    id = db.Column(db.BigInteger, primary_key=True)
    smr_id = db.Column(db.BigInteger, db.ForeignKey('smrs.id'))
```

```
    ip = db.Column(postgresql.INET)
    json_report = db.Column(postgresql.JSON)
    sign_hash = db.Column(db.String(250))

    user_email = db.Column(db.String(120), index=True)

    precolha_wifi_ssid = db.Column(db.String(120))
    precolha_wifi_pass = db.Column(db.String(120))

    precolha_nome = db.Column(db.String(120), index=True)
    precolha_contacto_nome = db.Column(db.String(120))
    precolha_contacto_telef = db.Column(db.String(20))

    precolha_coords_lat = db.Column(db.Float)
    precolha_coords_lng = db.Column(db.Float)

    on_battery = db.Column(db.Boolean())

    ### docs: https://techarena51.com/blog/flask-sqlalchemy-postgresql-
    tutorial/
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())


    def __init__(self, smr_id, ip, json_report, sign_hash, user_email,
    precolha_wifi_ssid, precolha_wifi_pass,
                 precolha_nome, precolha_contacto_nome,
    precolha_contacto_telef, precolha_coords_lat,
                 precolha_coords_lng, on_battery, estado='ON'):
        self.smr_id = smr_id
        self.ip = ip
        self.json_report = json_report
        self.sign_hash = sign_hash
        self.user_email = user_email
        self.precolha_wifi_ssid = precolha_wifi_ssid
        self.precolha_wifi_pass = precolha_wifi_pass
        self.precolha_nome = precolha_nome
        self.precolha_contacto_nome = precolha_contacto_nome
        self.precolha_contacto_telef = precolha_contacto_telef
        self.precolha_coords_lat = precolha_coords_lat
        self.precolha_coords_lng = precolha_coords_lng
        self.on_battery = on_battery
        self.estado = estado


class SMRReport(BaseModel, db.Model):
    """Model for the SMR Reports table"""
    __tablename__ = 'smr_reports'

    id = db.Column(db.BigInteger, primary_key=True)
    smr_id = db.Column(db.BigInteger, db.ForeignKey('smrs.id'))

    ip = db.Column(postgresql.INET)
    json_report = db.Column(postgresql.JSON)
    sign_hash = db.Column(db.String(250))

    battery_volt = db.Column(db.Float)
    depot_level = db.Column(db.Float)
```

```
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())


    def __init__(self, smr_id, ip, json_report, sign_hash, battery_volt
    =3.3, depot_level=0.0, estado='ON'):
        self.smr_id = smr_id
        self.ip = ip
        self.json_report = json_report
        self.sign_hash = sign_hash
        self.battery_volt = battery_volt
        self.depot_level = depot_level
        self.estado = estado


class SMRUpdate(BaseModel, db.Model):
    """Model for the SMR Updates table"""
    __tablename__ = 'smr_updates'

    id = db.Column(db.BigInteger, primary_key=True)
    smr_id = db.Column(db.BigInteger, db.ForeignKey('smrs.id'))
    loop_period_h = db.Column(db.Integer)
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())


    def __init__(self, smr_id, loop_period_h=24, estado='ON'):
        self.smr_id = smr_id
        self.loop_period_h = loop_period_h
        self.estado = estado


class Entidade(BaseModel, db.Model):
    """Model for the Entidades table"""
    __tablename__ = 'entidades'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
    nome = db.Column(db.String(120), index=True, unique=True)
    tipo = db.Column(postgresql.ENUM('ADMIN', 'CLIENT', name='
    entidade_tipo_enum'))
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    users = db.relationship('User', backref='entidades', lazy='dynamic')
    smrs = db.relationship('SMR', backref='entidades', lazy='dynamic')


    def __init__(self, nome, tipo='CLIENT', estado='ON'):
        self.public_id = uuid.uuid4()
        self.nome = nome
        self.tipo = tipo
        self.estado = estado


class User(BaseModel, db.Model):
    """Model for the Users table"""
```

```
    __tablename__ = 'users'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
    entidade_id = db.Column(db.BigInteger, db.ForeignKey('entidades.id')
    )
    nome = db.Column(db.String(120), index=True)
    nickname = db.Column(db.String(64), index=True, unique=True)
    email = db.Column(db.String(120), index=True, unique=True)
    password = db.Column(db.String(120), unique=False, nullable=False)
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    groups = db.relationship('Group', secondary='users_groups', backref=
    'users', lazy='dynamic')


    def __init__(self, entidade_id, nome, nickname, email, password,
    estado='ON'):
        self.public_id = uuid.uuid4()
        self.entidade_id = entidade_id
        self.nome = nome
        self.nickname = nickname
        self.email = email
        self.password = password
        self.estado = estado

        self.groups.append(Group(nome=nickname))


    def is_admin(self):
        entidade = Entidade.query.filter_by(id=self.entidade_id).first()

        if not entidade or entidade.tipo != 'ADMIN':
            return False
        else:
            return True


class Group(BaseModel, db.Model):
    """Model for the Groups table"""
    __tablename__ = 'groups'

    id = db.Column(db.BigInteger, primary_key=True)
    public_id = db.Column(db.String(50), index=True, unique=True,
    nullable=False)
    nome = db.Column(db.String(120), index=True, unique=True)
    created_on = db.Column(db.TIMESTAMP, server_default=db.func.
    current_timestamp())

    def __init__(self, nome, estado='ON'):
        self.public_id = uuid.uuid4()
        self.nome = nome
        self.estado = estado


# many-to-many
db.Table('users_groups',
```

```
        db.Column('user_id', db.BigInteger, db.ForeignKey('users.id')),
        db.Column('group_id', db.BigInteger, db.ForeignKey('groups.id')
    )
        )
```

assets/models.py

Apêndice B

Documentação da API REST do SGC

Este apêndice apresenta a documentação da API REST do SGC, gerada na especificação OpenAPI.

Smart Depot

Smart Depot API specification

Version 1.0.0

Contact information
Smart Depot API team

api@smartdepot.com (mailto:api@smartdepot.com)

https://smartdepot.duckdns.org (https://smartdepot.duckdns.org)

Terms of service
https://smartdepot.duckdns.org/api/terms/ (https://smartdepot.duckdns.org/api/terms/)

Security

basicAuth (HTTP Basic Authentication) Authenticate

smrKeyHeader (API Key) Authenticate

Name	X-SMR-Key
In	header

Paths

`/login`

`/users`

`GET /users`

`POST /users`

`/users/{user-id}`

`GET /users/{user-id}`

PUT /users/{user-id}

Users

Summary

=====

Description

=====

Parameters

Name	Located in	Description	Required	Schema
user-id	path	User identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
204	No Content	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 204 message: "No Content"
401	Unauthorized	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

PATCH /users/{user-id}

Users

Summary

=====

Description

=====

Parameters

Name	Located in	Description	Required	Schema
user-id	path	User identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
204	No Content	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 204 message: "No Content"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

DELETE /users/{user-id}

Users

Summary

=====

Description

=====

Parameters

Name	Located in	Description	Required	Schema
user-id	path	User identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
204	No Content	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 204 message: "No Content"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

/smrs

`GET /smrs`

SMR

Summary

List of SMR

Description

List of SMR

Responses

Code	Description	Schema	Examples
200	OK	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

POST /smrs

SMR

Summary

Create SMR

Description

Create SMR

Parameters

Name	Located in	Description	Required	Schema
smr	body	SMR	No	⇄ ▾ **SMR {** *SMR Info* smr-header: ▸ **smr-header { }** id: integer * brand: string * model: string * firmware: string * depot-capacity: integer read-rate: integer }

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

DELETE /smrs

SMR

Summary

Deletes all SMR

Description

Deletes all SMR

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

/routes

POST /routes

Routes

Summary

Optimize Route

Description

Optimize Route

Parameters

Name	Located in	Description	Required	Schema
best-route	body	ROUTES	No	⇄ ▾ **ROUTE {** *Optimize Route* smr-id: integer }

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

/smrs/{smr-id}/

`GET /smrs/{smr-id}/`

SMR

Summary

Returns a SMR by ID.

Description

Returns the details of a SMR by ID.

Parameters

Name	Located in	Description	Required	Schema
smr-id	path	SMR identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

PUT /smrs/{smr-id}/

SMR

Summary

Update full details of a SMR

Description

Update full details of a SMR identified by ID.

Parameters

Name	Located in	Description	Required	Schema
SMR	body		No	⇄ ▾ **SMR {** *SMR Info* smr-header: ▸ **smr-header { }** id: integer * brand: string * model: string * firmware: string * depot-capacity: integer read-rate: integer }
smr-id	path	SMR identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

DELETE /smrs/{smr-id}/

SMR

Summary

Delete a SMR

Description

Delete a SMR

Parameters

Name	Located in	Description	Required	Schema
smr-id	path	SMR identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 200 message: "OK"
401	Unauthorized	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

/smrs/{smr-id}/versions

GET /smrs/{smr-id}/versions

SMR SMR-Versions

Summary

List of SMR Versions

Description

List of SMR Versions

Parameters

Name	Located in	Description	Required	Schema
last	query	Limite to last n elements (?last=1) (?last=3). If last parameter is not preset all elements are returned.	No	⇄ ▾ integer (int32)
smr-id	path	SMR identifier	Yes	⇄ ▾ integer (int64)

Responses

Code	Description	Schema	Examples
200	OK	⇄ ▾ **versions[** *List of Versions for this SMR* ▸ **SMR-Version { }**]	
401	Unauthorized	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 401 message: "Unauthorized"
403	Forbidden	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 403 message: "Forbidden"
404	Not Found	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"
default	Unexpected error	⇄ ▾ **code-resp {** *Response Code and Message* code: ▸ integer * msg: string }	application/json ▾ Object code: 404 message: "Not Found"

Try this operation

/smrs/{smr-id}/versions/{ver-id}

GET /smrs/{smr-id}/versions/{ver-id}

POST /smrs/{smr-id}/versions/{ver-id}

/smrs/{smr-id}/versions/{ver-id}/upgrade

GET /smrs/{smr-id}/versions/{ver-id}/upgrade

/smrs/{smr-id}/reports

GET /smrs/{smr-id}/reports

POST /smrs/{smr-id}/reports

/smrs/{smr-id}/reports/{report-id}

GET /smrs/{smr-id}/reports/{report-id}

/clients

GET /clients

POST /clients

/clients/{client-id}

GET /clients/{client-id}

PUT /clients/{client-id}

PATCH /clients/{client-id}

DELETE /clients/{client-id}

/groups

GET /groups

POST /groups

/groups/{group-id}

Models

smr-header

```
▾ smr-header {
  SMR Header Info
  action:    string *
  from:      string *
  to:        string *
  method:    string
  timestamp: string *
}
```

SMR

```
▾ SMR {
  SMR Info
  smr-header:      ▸smr-header { }
  id:              integer *
  brand:           string *
  model:           string *
  firmware:        string *
  depot-capacity: integer
  read-rate:       integer
}
```

SMR-Version

```
▾ SMR-Version {
  SMR Version Info
  smr-header: ▸smr-header { }
  id:          ▸ string *
  timestamp:   ▸ string *
  applied:     ▸ boolean
}
```

ROUTE

```
▾ ROUTE {
  Optimize Route
  smr-id: integer
}
```

code-resp

```
▾ code-resp {
    Response Code and Message
⇄   code: ▸ integer *
    msg:  string
  }
```