INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Portal de Contratação Utilities

MARIANA LOURENÇO AMADO PRETO
Outubro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Portal de Contratação Utilities

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

Mariana Lourenço Amado Preto

Orientador: Prof. Luís Miguel Pinho Nogueira
Supervisor Externo: Agostinho Cruz

Porto, 15 de Outubro de 2020

Dedicatória

Aos meus pais e aos meus irmãos por todo o apoio.

Resumo

Num mundo cada vez mais competitivo e não alheio a crises económicas, a margem de erro é mínima para as entidade empresariais. Deste modo, têm estas instituições que estar dotadas de mecanismos de resposta focadas nas linhas da inovação e criatividade, não podendo os seus líderes descurar a prossecução da estabilidade económico-financeira.

A tais circunstancialismos, alia-se a ideia da globalização o que tudo somado, leva à obrigatoriedade das empresas em adotarem os valores de eficácia, eficiência e competitividade nas áreas de angariação de novos clientes e da sua manutenção, devendo os seus métodos organizacionais ser sinónimo destes ideais chaves.

No que respeita aos serviços de utilidade pública, a competitividade na aquisição e expansão da clientela, não deixa de ser uma realidade inegável, pelo que as empresas têm o dever de investir na melhoria e desenvolvimento dos instrumentos tecnológicos de gestão, captação e cativação de novos consumidores, para que tais organizações possam conhecer um desenvolvimento sustentável e harmonioso.

Palavras-chave: Gás Natural, Pré-Contratos, BC, OLMC, Automatização, Informatização

Abstract

In an increasingly competitve world and not unrelated to economic crises, the margin for error is minimal for business entities. Therefore, these institutions have to be equipped with response mechanisms focused on innovation and creativity, and their leaders must not neglect the pursuit of economic and financial stability.

In fact, the impositions of the market, the demands of clients, the pressure from investors, the competition from other peers, generate ferocity in the fight for competitiveness and control of public utility markets.

In addition to these circumstances, the idea of globalisation, which all adds up, leads companies to adopt the values of effectiveness, efficiency and competitiveness in the areas of attracking new clients and maintaining them, and their organisational methods should be synonymous with these key ideals.

As far as public utility services are concerned, competitiveness in acquiring and expanding customers is an undeniable reality, and companies have a duty to invest in the improvement and development of technological instruments for managing, attracting and captivating new consumers, so that such organizations can experience a sustainable and harmonious development.

Agradecimentos

À minha família e aos meus amigos por todo o apoio dado durante este percurso.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

Application Pool	Propriedade do *Internet Information Services* (IIS) que permite isolar aplicações umas das outras, mesmo que estejam em execução no mesmo servidor. Dessa forma, se houver um erro numa aplicação, este não debilitará outras aplicações. Além disso, os *application tools* permitem separar aplicações diferentes que exigem níveis diferentes de segurança (Thangarathinam 2020).
backlog	Log ou resumo histórico de acumulação de trabalho num determinado intervalo de tempo.
Cron	Processo automático da framework da empresa.
DF4A	Framework desenvolvida pela CPCIT4ALL-Lda. para acelerar e ajudar no desenvolvimento de aplicações Web, acrónimo do inglês Develop Framework 4 All.
DGSwitch	API que faz a ligação do portal da contratação com o OLMC.
estrangulamento	Diz-se que ocorre um estrangulamento quando a capacidade de um sistema é limitada por um único componente, como por exemplo, o gargalo de uma garrafa que diminui o fluxo da água.
front end	Componentes de um sistema de informação (software) usado para interagir com o utilizador, normalmente associado, tem um aplicativo do lado do cliente (Rouse 2019).
Kanban	Processo de gestão e desenvolvimento de projetos.

OLMC O OLMC (Operador Logístico de Mudança de Comercializador) é responsável pela Gestão do Processo de Mudança de Comercializador (GPMC) de eletricidade e gás natural, tem como objetivo garantir que a mudança requerida pelo consumidor final junto dos agentes do mercado, seja efetuada de forma célere, simples e transparente (OLMC 2019).

SaaS Um Software as a Service é um programa que não é instalado em nenhum computador e é usado como um serviço, pela internet.

uptick Um pequeno aumento ou uma ligeira tendência ascendente..

Siglas

API	*Application Programming Interface*.
BC	Dynamics 365 Business Central.
CAE	Classificação Portuguesa das Atividades Económicas.
CD	Continuous Delivery.
CI	Continuous Integration.
CPCis	Companhia Portuguesa de Computadores Informática e Sistemas S.A.
CRUD	*Create*, *Read*, *Update*, *Delete*.
CUI	Código Universal da Instalação.
ERSE	Entidade Reguladora dos Serviços Energéticos.
FFE	*Front End of Innovation*.
FFE	Fuzzy Front End.
GNV	Gás Natural Veicular.
GPMC	Gestão do Processo de Mudança de Comercializador.
IIS	*Internet Information Services*.
NCD	New Concept Development.
NPD	New Product Development.
ORD	Operador de Rede de Distribuição.
REN	Redes Energéticas Nacionais.
RUP	Rational Unified Process.
SLA	Service Level Agreement.
SNGN	Sistema Nacional de Gás Natural.
SOAP	*Simple Object Access Protocol*.
SSMS	SQL Server Management Studio.

TFS Team Foundation Server.

US *User Story*.

VC Valor para o Cliente.

Capítulo 1

Introdução

O fornecimento de gás, como se encontra regulado, tem de ser enquadrado como sendo de interesse geral, isto é, em que o Estado tem um dever de garantir que todos os serviços serão efetivamente prestados a todo o território português, mas esses serviços não têm cariz público, podendo ser prestados por entidades privadas/empresas.

É sobre o serviço prestados pelas empresas privadas de gás natural que o presente trabalho se irá debruçar, nomeadamente no tópico de gestão operacional de obtenção de clientela, pois só com a existência de consumidores é que uma empresa subsiste, uma vez que o lucro é a razão de existência enquanto ente empresarial.

1.1 Contexto

Num mercado cada vez mais competitivo, a que se adiciona a propagação do pressuposto da liberalização dos vários setores, onde se inclui o energético, uma realidade emerge, a de que quem procura pode selecionar livremente a quem pretende recorrer, ou seja, filtra e seleciona a oferta.

No caso em apreço, às empresas de serviços de utilidade pública levantam-se novos desafios no que concerne à sua programação e perspetivas para o futuro, foco que deve abranger novas metodologias programáticas a aplicar, quer no mercado português/interno quer no externo.

Dessa forma são as empresas fornecedoras dos bens que têm o dever, a obrigatoriedade e a necessidade de redesenhar, colmatar e adaptar o planeamento estratégico da sua oferta, por forma a não perderem a atratividade dos seus serviços para os clientes, perante a nova realidade de competitividade implacável dos seus pares empresariais.

A aplicação que se está a desenhar e a desenvolver foca-se no apoio da gestão de novos clientes de uma comercializadora de gás natural. O procedimento de adesão de um novo cliente é longo, complexo e implica, frequentemente, a comunicação entre várias entidades.

Para se aumentar a celeridade do processo de suporte, que se quer mais rápido, ágil e automático, o portal vai comunicar com estas entidades, como se mostra na figura 1.1. Isto proporciona ao utilizador um acesso facilitado à informação do cliente, aliado ao facto de também oferecer mais segurança ao comercial, uma vez que existem processos instantâneos de preenchimento de campos, evitando-se assim erros ortográficos, ou incoerências entre empresas.

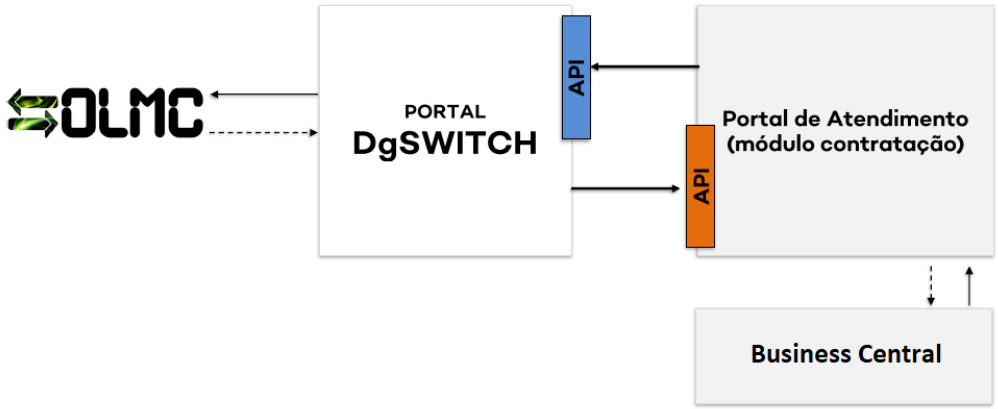

Figura 1.1: Diagrama de fluxo de conteúdo entre APIs

Geralmente, quando um cliente quer alterar de comercializadora de gás natural, este apenas comunica à nova empresa a que vai aderir, e esta, por sua vez, trata de todas as burocracias necessárias para que este fique associado a ela. Não é o cliente que contacta com a antiga empresa para alterar de operadora, nem é ele que contacta com as entidades responsáveis por esta gestão.

Por a responsabilidade ser da comercializadora, torna-se necessário que esta tenha acesso a todos os dados do cliente. Até à presente data, estes processos eram feitos em papel, através de formulários que eram preenchidos à mão pelo comercial.

Após este preenchimento, o comercial, por chamada telefónica, comunicava ao OLMC, que é a entidade que garante que a mudança de comercializador de gás natural solicitada pelo consumidor junto dos agentes do mercado, seja feita de forma célere (OLMC 2019), e o OLMC, por sua vez, comunicava com o Operador com o qual o cliente quer cessar o contrato.

A proposta da CPCIT4ALL veio no sentido de desenvolver um novo módulo para o Portal de Atendimento, que irá, através de mecanismos de comunicação automática, transmitir informação, tanto ao Portal DgSwitch, que é responsável por gerir todos os processos que o OLMC necessitar comunicando-os ao Portal de Atendimento, como ao Dynamics 365 Business Central (BC)[1], ficando este com a responsabilidade da faturação dos clientes.

1.2 Problema

Quase todo o território de Portugal continental encontra-se abrangido pelas diversas redes de empresas de serviços de utilidade pública, contudo também é inegável que o interior continua a ser alvo de discriminação tendo conhecido recentemente uma aposta relevante das empresas dedicadas à comercialização, fornecimento e distribuição de gás natural.

Dentro do universo das empresas acima referidas, cumpre dar ênfase ao projeto que atualmente está a ser desenvolvido pela CPCIT4ALL, projeto esse que será objeto e foco da presente dissertação.

[1] https://dynamics.microsoft.com/pt-pt/business-central/overview/

A CPCIT4ALL pertence ao grupo empresarial português, Companhia Portuguesa de Computadores Informática e Sistemas S.A (CPCis), o qual se dedica há mais de vinte anos ao setor das tecnologias de informação,e neste momento encontra-se a desenvolver um projeto encomendado por uma empresa nacional que se dedica à comercialização e fornecimento de gás natural. Este ente empresarial tem como objetivo da sua atuação o princípio da sustentabilidade, versado na aposta e evolução tecnológica, que lhe permite a obtenção de uma energia mais limpa, ecológica, natural e mais amiga do ambiente.

Assim e no que concerne à presente dissertação, a aplicação que está a ser desenvolvida foca-se no apoio aos procedimentos de gestão de clientes/consumidores de uma empresa cuja atividade se prende com a comercialiazação de gás natural. Por outras palavras, a aplicação irá debruçar-se no processo de adesão de novos consumidores, tornando esta operação mais dinâmica, menos complexa, mais eficaz e eficiente, o que implicará na prática um incremento na rapidez da comunicação entre as diversas entidades.

Como já foi dito, tem-se assistido a um acréscimo no investimento da construção e gestão de infraestruturas de gás natural no interior de Portugal (Silva 2019). A empresa distribuidora para a qual o projeto está a ser desenvolvido, tomou então a decisão de criar uma nova comercializadora. Tal é vantajoso para a empresa, pois esta é uma das principais distribuidoras de gás natural, e também porque passa a ter menos necessidade de recorrer a terceiros.

Como o interior de Portugal tem sido alvo de investimentos avultosos nas ultimas décadas, abrangendo vários setores, é normal que exista uma grande competição entre as comercializadoras já existentes na angariação de novos clientes e na conservação destes.

De facto há uma multiplicidade de mecanismos na angariação de clientela. Podem-se enumerar: as angariações via contacto telefónico realizados por um funcionário a qualquer cidadão; nos contactos entre funcionários da empresa e as pessoas que orbitam no seu foro familiar e social, onde numa conversa pode ser recomendado direta ou indiretamente os serviços da instituição; ou o próprio consumidor pode por sua livre iniciativa contactar com a empresa para aderir a um determinado serviço.

Em todas as situações acima referidas, pode-se verificar na prática perda de informação relevante. Tal pode ocorrer caso o método selecionado pela instituição não esteja devidamente informatizado e automatizado, sendo esta realidade um dos problemas da empresa para a qual está a ser desenvolvido o presente projeto e para o qual se procura fornecer as soluções e respostas tidas como pertinentes.

Atualmente, o processo de contratação da organização em análise, é feito através de formulários em papel. Tal causa inúmeros constrangimentos:

- O facto de ser em papel, obriga a que sejam cumpridos requisitos legais, nomeadamente no arquivo dos mesmos;
- Verifica-se a falta de informação para as chefias das equipas comercias, pois o processo é suportado em papel, o que origina perdas desnecessárias de tempo na procura de documentos no arquivo físico;
- Na integração entre vários sistemas. Não sendo automática e validada por controlos automáticos, não dá a garantia de boa execução, devido ao facto de todo o *input* ser manual e repetitivo;

- Não sendo um processo automático, pode causar grandes intervalos de espera, tanto entre empresas - troca e espera de informação necessária - como por parte do cliente, que pretende que o seu contrato seja processado o mais rapidamente possível.

Deste modo, a quantidade de oportunidades de negócio que são desperdiçadas tem um grande impacto, devido à inexistência de uma plataforma onde toda a informação está concentrada e com fácil acesso.

Um sistema que viesse apoiar os comerciais da organização, que disponibilizasse os dados dos clientes e o estado dos seus contratos e que ao mesmo tempo reunisse toda a informação proveniente de diferentes serviços, iria aumentar a rentabilidade dos funcionários, o que diminuiria consideravelmente os períodos de espera dos clientes.

Para além disto, seria muito vantajoso para os chefes de equipas comerciais, ter acesso à qualidade de trabalho dos seus comerciais. Tal iria permitir que se comparasse a quantidade de contratos feitos por cada um dos membros da sua equipa, e ter a perceção do volume de trabalho diário do seu grupo. Isto seria benéfico para os índices de produtividade e seus rácios, dados importantes para qualquer boa gestão empresarial.

1.3 Objetivos

A aplicação a ser desenvolvida terá como principal objetivo, o de suportar o processo de criação de novos contratos de clientes no âmbito dos serviços de utilidade pública.

O portal permitirá a um conjunto de utilizadores registar e realizar o acompanhamento dos pedidos de contratos. É uma nova aplicação que irá fazer parte de um conjunto de várias aplicações criadas para a gestão do negócio das *utilities*.

As funcionalidades previstas são:

- Portal contratação: para comerciais e chefes de equipa, que irão criar e gerir os contratos dos seus clientes. Aqui os comerciais poderão criar pré contratos, e ter acesso ao seu estado atual;
- Portal BackOffice: para gestores da comercializadora, onde estes irão validar contratos para OLMC, tendo acesso às respostas dadas pelo Switch, quer estas sejam de sucesso ou insucesso. É nesta que a comercializadora vai poder criar as suas equipas e definir os chefes de equipa e comerciais. Para além disto vai poder gerir os agendamentos enviados pelo OLMC com as Entidades Inspetoras;
- Gestão da comunicação com clientes;
- Controlo de qualidade do trabalho desenvolvido pelos trabalhadores da comercializadora;
- API BC: para trabalhadores da comercializadora, os quais vão gerir os contratos e ter acesso à faturação dos clientes da empresa;
- Disponibilização de mecanismos de comunicação automática com aplicações externas (DGSwitch);
- Disponibilização de mecanismos de comunicação automática com o BC.

1.4 Análise de Valor

A CPCIT4ALL, apresenta uma solução para a área dos serviços de utilidade pública para todas as comercializadoras. Após uma análise de mercado, foi notória a falta de aplicações para este efeito, e posto isto, houve a necessidade de idealizar um sistema que criasse uma ferramenta para a monitorização e criação de contratos.

A solução aqui documentada, prevê a automação de processos no decorrer da angariação de um novo cliente para a comercializadora, contribuindo para a diminuição da duração do processamento de um novo contrato. Para além disso, vai também reduzir a probabilidade de erro humano aquando a criação de um contrato, o que atualmente é um dos problemas que a comercializadora enfrenta.

Estas novas condições irão facilitar o trabalho das pessoas envolvidas neste processo, automatizando a integração de contratos em OLMC e criação de contratos em BC, bem como irão aumentar a qualidade e confiabilidade na sua criação, providenciando assim aos clientes desta empresa uma segurança que antes não lhes era assegurada.

A análise do valor da solução em questão será detalhada na secção 2.3 do capítulo 2.

1.5 Metodologia de trabalho

O processo de planeamento e desenvolvimento deste projeto seguiu uma metodologia Kanban, que é baseada numa abordagem ágil, iterativa e incremental.

Sempre que era necessário alterar as funcionalidades já implementadas, uma nova *User Story* (US) era criada e recebia prioridade. Na gestão destas tarefas, foi utilizado um software interno da CPCIT4ALL, para gestão de tempo e da equipa, bem como um software da comercializadora, e é neste segundo onde o próprio cliente cria um ticket e associa ao trabalhador da CPCIT4ALL, permitindo assim controlar o backlog do projeto e manter um registo das atividades desenvolvidas ao longo deste.

Dependendo do tipo de alterações requisitadas, poderia ser necessária, ou não, uma conversa ou reunião com o cliente. Caso fossem erros existentes no software com que o cliente se tivesse deparado, estes recebiam máxima prioridade, e eram corrigidos o mais rápido possível.

No caso de serem solicitadas novas funcionalidades ou alterações a alguma já existente, era necessário perceber todos os requisitos necessários e qual o objetivo da funcionalidade pedida. Deste modo havia uma reunião com o cliente, presencial, ou por chamada de vídeo ou telefónica. Após estas reuniões, era enviado um email à equipa responsável com o detalhe das especificações, permitindo assim que tudo ficasse documentado.

1.5.1 Processo de Desenvolvimento

Como mencionado anteriormente, uma abordagem iterativa e incremental foi escolhida para o projeto, adotando-se uma metodologia ágil baseada no uso da estrutura de desenvolvimento Kanban. Inicialmente, as principais USs a serem desenvolvidas foram definidas e, em seguida, divididas em responsabilidades/tarefas mais simples para se ajustarem às necessidades dos clientes.

Com o progresso no processo de design e desenvolvimento, surgiram novos detalhes de negócio que eventualmente se transformaram em tarefas.

Sendo terminologias relacionadas, a nossa metodologia de trabalho está em conformidade com o Rational Unified Process (RUP), sendo definido por Lines (2012) como um processo de desenvolvimento de software que engloba as ações necessárias para transformar um conjunto de requisitos num sistema de software, combinando a iteratividade, ciclos de vida incrementais e verificação da qualidade do software, de forma a que cada entrega de software num ciclo agregue mais valor ao produto em relação ao ciclo anterior.

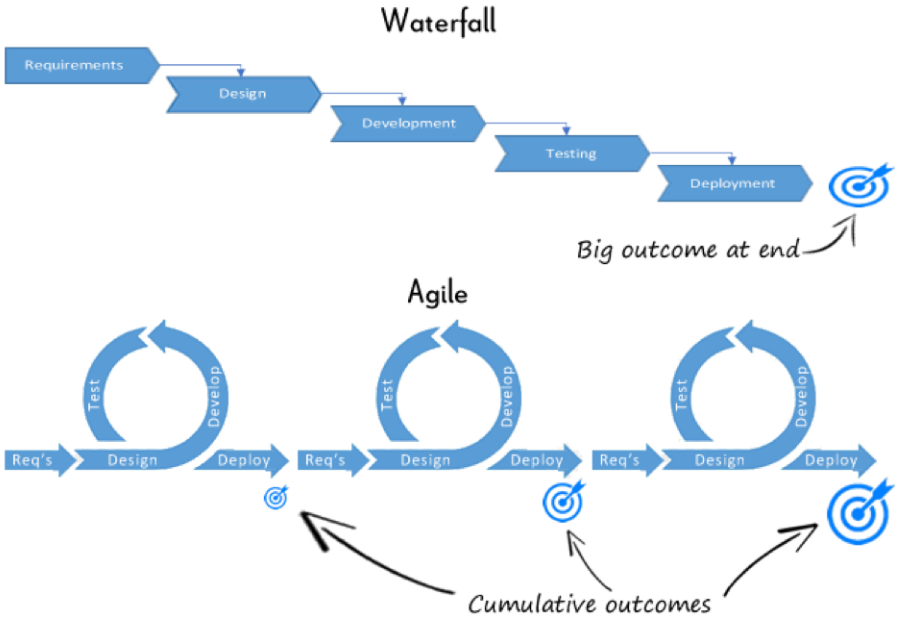

Figura 1.2: Diagrama de comparação entre os méstodos Agile e Waterfall [Fonte: (Schaeffer 2020)].

Segundo Schaeffer (2020), um dos principais recursos de uma abordagem ágil é que a mesma lógica da modelagem contínua também se aplica a outros componentes, seja em termos de design, teste ou mesmo código. A análise consiste em estudar o negócio, a fim de definir as áreas cruciais em que a equipa se deve focar.

Na fase de design, o estudo é mais orientado para a solução dos artefactos desenvolvidos na fase de análise. Dependendo da abordagem, um conjunto variado de documentos pode aparecer nesta fase, servindo como orientação na fase de desenvolvimento de código.

A fase de desenvolvimento, corresponde ao espaço de tempo no qual o código é efetivamente desenvolvido para atender aos requisitos capturados na fase de análise.

Finalmente, a fase de teste corresponde ao desenvolvimento de testes, para validar o que foi desenvolvido ou será desenvolvido, dependendo da abordagem em questão.

O método Kanban requer comunicação em tempo real e total transparência do trabalho. Os itens de trabalho são apresentados visualmente num quadro Kanban, permitindo que os membros da equipa vejam o estado de cada tarefa que está a ser desenvolvida a qualquer momento.

Uma das principais características associadas ao uso de uma metodologia ágil, principalmente quando utilizada em conjunto com o Kanban, é a existência de uma atitude de entrega de protótipos funcionais ao cliente com alta frequência. A existência de curtos períodos de

tempo entre as entregas permite que o cliente monitorize de perto o progresso feito e forneça *feedback* com frequência.

Desta forma, é possível modelar o produto com base nos requisitos do cliente ao longo de todo o processo de desenvolvimento, reduzindo custos, a fim de convergir a solução para os requisitos estabelecidos.

1.5.2 Sistema de Controlo de Versões

O software de controlo de versões permite que uma equipa mantenha um histórico de todas as alterações feitas nos arquivos de um projeto específico (Atlassian 2020). Para este projeto, o Team Foundation Server (TFS) foi usado como sistema de controlo de versões.

1.5.3 Fluxo de trabalho

Um bom *workflow* de desenvolvimento é extremamente importante para todas as equipas que têm a intenção de organizar o seu processo de desenvolvimento. Trabalhar com equipas pode ser bastante moroso e fatigante se não houver um processo para rever e verificar o código, ou mais importante, para impedir que código errado chegue a produção.

O processo de desenvolvimento adotado está divido em 3 ambientes principais: ambiente de desenvolvimento, ambiente de qualidade e ambiente de produção.

O primeiro, trata-se do ambiente restrito à CPCIT4ALL. Como o nome refere, é onde é feito todo o desenvolvimento dos requisitos solicitados pelo cliente. Aqui o código é feito pelo programador, testado, e só depois é enviado para o repositório.

Após todos os testes serem feitos pela equipa de desenvolvimento, o código é colocado no ambiente de qualidade, isto é, é colocado no servidor do cliente, deste modo, só tem acesso à aplicação quem tiver acesso à rede interna do cliente. É nesta fase que o cliente faz testes de usabilidade, com dados o mais próximo do real possível, de modo a verificar a qualidade e viabilidade do programa. É também aqui que a equipa de desenvolvimento recebe o *feedback* do cliente, quer sejam alterações, erros ou a confirmação de que a aplicação funciona corretamente.

Depois dos testes de usabilidade serem realizados por parte do cliente e a veracidade da aplicação ser confirmada, então o projeto passa para o ambiente de produção. É então colocado no servidor de produção, a que os funcionários da comercializadora têm acesso, tanto na rede interna, como através de um link externo, e onde trabalham diariamente.

1.6 Estrutura do documento

O relatório está dividido em cinco capítulos. Para além do presente capítulo, existe o capítulo 2, que está divido em 3 partes principais, onde na primeira se discute o contexto de trabalho e alguns detalhes relevantes para a solução final do sistema. Na segunda parte, denominada Estado da arte, é apresentado o estudo e trabalho de pesquisa realizado acerca de tecnologias existentes e outras abordagens e de como os sistemas serviços de utilidade pública se encontram nos dias de hoje. Por fim é feita a análise de valor da solução projetada, tendo como principal objetivo averiguar o valor real que o sistema final gera para o cliente que usufruir desta aplicação.

O Capítulo 3, Design, passa por apresentar o design desenvolvido até à data, não deixando de apresentar outras abordagens possíveis do problema, referindo os pontos fracos e pontos fortes de cada uma delas, seguindo sempre as boas práticas de Engenharia Informática.

Na Avaliação e Experimentação, Capítulo 4, são descritas as experiências e avaliações que foram realizadas à solução apresentada.

Por fim, o Capítulo 5 conclui este documento, fazendo uma pequena reflexão do trabalho desenvolvido.

Capítulo 2

Contexto e Estado da Arte

2.1 Contexto

2.1.1 Gás Natural em Portugal

O Gás Natural é uma fonte de energia de origem natural, que resulta da decomposição de matéria orgânica vegetal e animal. É um combustível fóssil, deste modo não é um recurso renovável, porém é um dos combustíveis mais versáteis, sendo predominantemente usado na área de transportes, substituindo derivados do petróleo pelo Gás Natural Veicular (GNV).

Cada vez mais existe uma necessidade crescente de novas fontes de energia e gradualmente o gás natural está a ser considerado como uma das principais alternativas, uma vez que tem um preço bastante competitivo em comparação às possibilidades existentes. Por ter um processo de queima eficiente, é possível atingir níveis de rendimento mais elevados nos equipamentos, baixando o consumo de energia e diminuindo os custos de manutenção (Galp 2020).

Uma das grandes vantagens do gás natural é que é o combustível fóssil mais limpo, pois da queima deste resultam menores emissões de óxidos de enxofre e de azoto (responsáveis pelas chuvas ácidas), bem como de dióxido de carbono, que está na origem do efeito estufa.

Segundo a Redes Energéticas Nacionais (REN), em Portugal, no ano de 2017, foi atingido o máximo histórico de consumo de gás natural, ficando 11% acima do máximo anterior (REN 2017). Em janeiro deste ano, também segundo a REN, foi atingido um novo máximo histórico no transporte de gás natural devido ao aumento progressivo no consumo em Portugal, atingindo novamente um máximo histórico, com um crescimento de 7.2% face ao período homólogo (REN 2020).

Estes valores históricos registados nos últimos anos, contribuem para o aumento de interesse nesta área de negócio, fazendo com que a competição entre as comercializadoras de gás aumente, levando a que estas se esforcem para fornecer aos seus clientes o melhor serviço possível. É sobre este ponto que incide o projeto, e tem como principal objetivo ajudar a empresa na gestão dos seus clientes, assim como na organização de informação quanto à possível angariação de novos consumidores.

2.1.2 Detalhes do Contexto

As diferentes atividades no mercado do Gás Natural, envolvem um conjunto alargado de intervenientes (GALP 2020). Estes são:

- **Comercializadores**: Existem dois tipos de comercializadores, os livres e os regulados. Os comercializadores regulados são responsáveis pelo fornecimento de gás a todos os clientes que se encontram no mercado regulado.

  Os comercializadores livres têm a liberdade de comprar e vender gás natural, tendo acesso às infraestruturas do Sistema Nacional de Gás Natural (SNGN). Os clientes têm o direito de escolher o seu comercializador, optando pelas melhores condições de fornecimento;

- **Operador de Rede de Distribuição (ORD)**: Responsável por desenvolver e gerir as redes de distribuição de gás natural numa determinada área do país. Estes são independentes de qualquer comercializador, tendo de assegurar todos os serviços na sua área de concessão referentes ao fornecimento de Gás Natural nos locais de consumo ligados à rede de distribuição;

- **Cliente**: Sendo este o principal foco do SNGN, todos os serviços são pensados e estruturados em função deste e ajustados às suas necessidades. Existem 3 classes de clientes diferentes: Clientes Domésticos; Clientes economicamente vulneráveis; Clientes não domésticos com consumo anual inferior ou igual a 10 000 metros cubicos (n);

- **Entidade Reguladora dos Serviços Energéticos (ERSE)**: Tem como principal objetivo proteger os interesses dos consumidores em relação a preços, qualidade de serviço, acesso à informação e segurança de abastecimento.

Além destes intervenientes, existe também o OLMC, que tem como principal objetivo garantir que a mudança requerida pelo consumidor final junto dos agentes do mercado seja efetuada de forma célere, padronizada e desmaterializada, simples e transparente. Ele é responsável pela Gestão do Processo de Mudança de Comercializador (GPMC) de eletricidade e gás natural, disponibilizando aos consumidores finais acesso fácil à informação a que têm direito. A figura 2.1 indica o fluxo de informação entre os diferentes intervenientes nestas operações.

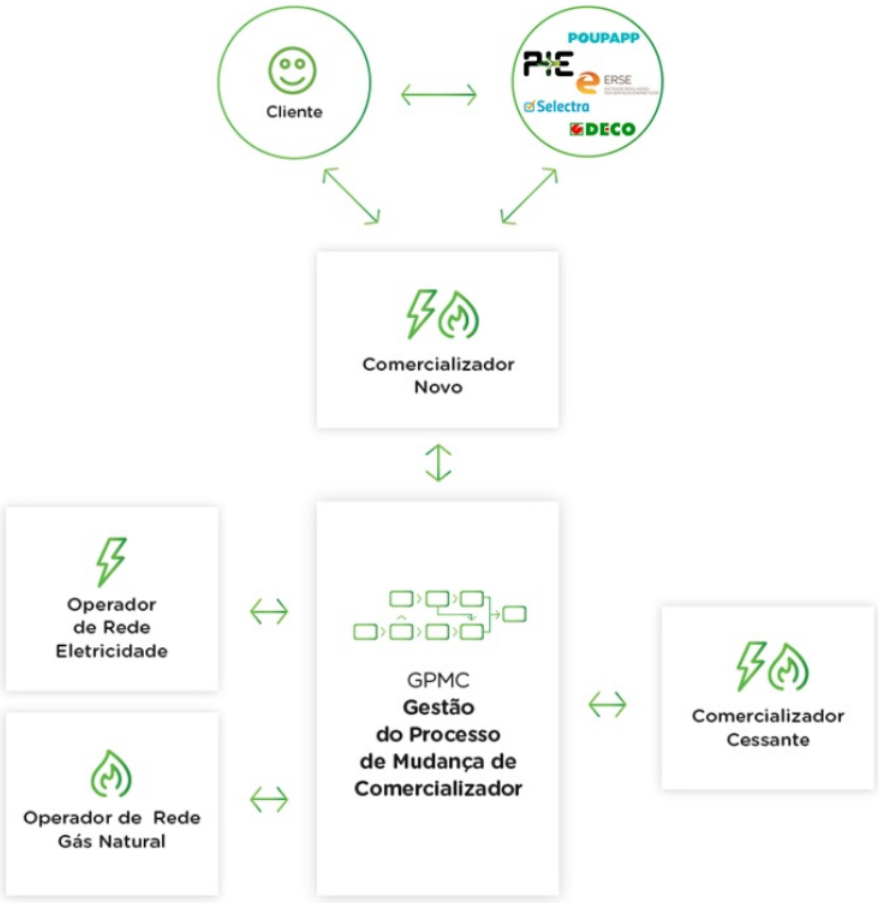

Figura 2.1: Diagrama de representação de fluxo de informação por mudança de comercializador [adaptado de (OLMC 2019)].

2.1.3 Conceitos de negócio

A Figura 2.2 representa, de forma não técnica e de alto nível, o sistema pensado pelos responsáveis do projeto para responder aos problemas.

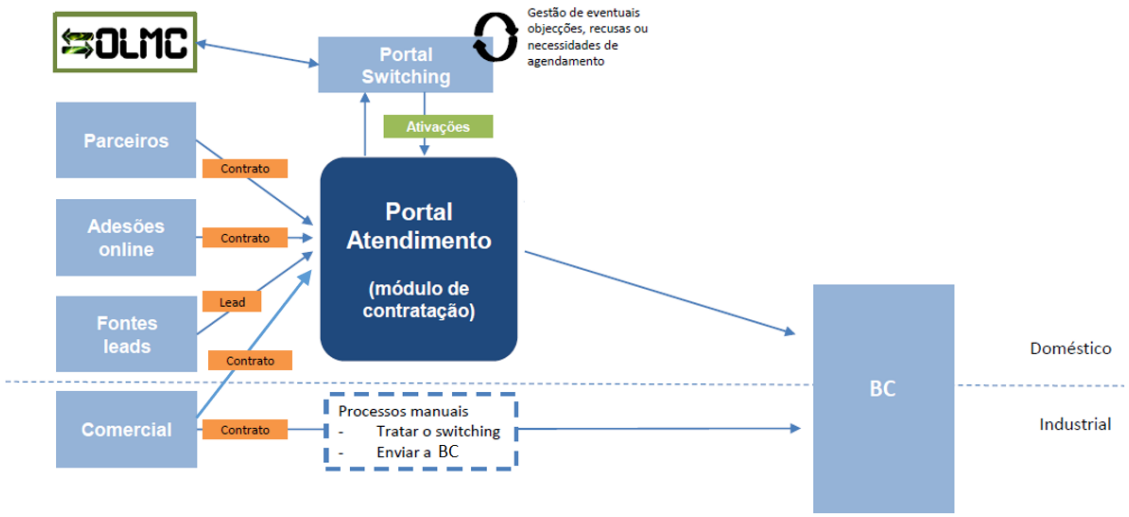

Figura 2.2: Modelo de negócio de alto nível

Através da análise da figura 2.2 é possível identificar os componentes que irão fazer parte do sistema:

- **Módulo de Contratação**: Sistema responsável por gerir os contratos que são criados, enviando-os para o Portal Switching. É também este que vai receber a informação proveniente do switch e enviar contratos para BC;
- **Portal Switching**: Responsável por comunicar com o OLMC e por gerir os eventos provenientes deste, que podem ser recusas, objeções, agendamentos e ativações de contratos;
- **BC**: Responsável pela faturação dos contratos dos clientes da comercializadora, que é feita mensalmente, registando todos os movimentos na contabilidade da empresa, de modo a conferir toda a contabilidade desta com toda a transparência.

2.2 Estado da Arte

2.2.1 Soluções Existentes

Tratando-se de um sistema que é interno a empresas e sendo que o mercado existente tem muito pouca oferta, existe apenas um sistema que é de conhecimento da CPCIT4ALL que tem como objetivo gerir e enviar contratos para OLMC, que se chama GBOX. Além deste produto, não existe conhecimento de nenhum outro sistema que tenha como objetivo gerir os contratos de gás de uma comercializadora, sendo que estes são feitos através de formulários em papel.

GBOX

As particularidades deste sistema são de conhecimento da equipa apenas devido ao facto de a comercializadora para que a CPCIT4ALL está a desenvolver o novo produto ter trabalhadores que já usaram e que deram o seu *feedback* à equipa de desenvolvimento. Esta solução é oferecida como uma Web App e tem como principais serviços:

- Criação de contratos;
- Envio de contratos para OLMC;
- Consulta do estado do contrato;
- Consulta do fluxo de informação com o Switch;

As tecnologias usadas para desenvolver esta aplicação são desconhecidas pela equipa de trabalho, uma vez que, como já referido, é um produto interno de uma outra comercializadora.

Porém, os utilizadores da GBox com que a equipa de desenvolvimento contactou mostraram um grande descontentamento acerca do uso desta aplicação, uma vez que não quiseram manter-se clientes desse produto devido ao facto de este conter inúmeros bugs de software e de não corresponder às expetativas dos clientes.

2.2.2 Comparação entre soluções

A Tabela 2.1 apresenta a comparação entre a solução referida acima e a solução que a CPCIT4ALL pretende implementar (Solução Projetada).

Tabela 2.1: Comparação entre soluções

	Módulo Contratação	**GBox**
Criação de contratos	✓	✓
Envio de contratos para OLMC	✓	✓
Consulta do estado do contrato	✓	✓
Consulta do fluxo de informação com o Switch	✓	✓
Criação de diferentes graus de acesso	✓	✓
Criação de equipas comerciais	✓	✗
Monitorização de equipas	✓	✗
Gestão de Agendamentos com clientes	✓	✗
Processos automáticos	✓	✗
Integração com BC	✓	✗
Gestão de Leads	✓	✗
Possibilidade de extração de relatórios acerca de contratos	✓	✗

Analisando a Tabela 2.1 é possível ver as vantagens do serviço idealizado pela CPCIT4ALL. Este proporciona aos seus utilizadores diferentes graus de acesso, sendo que os gestores da empresa terão acesso ao BackOffice deste módulo, podendo gerir e ter acesso a todas as operações que necessitam, mantendo a informação importante toda no mesmo local. Para além disto, uma das principais mais valias é o facto de ter processos automáticos, que fazem as ligações ao OLMC e ao BC, causando uma diminuição de ocorrência de erros humanos.

Uma das grandes queixas feitas à GBox por parte dos seus utilizadores foi que esta tinha vários bugs e não era *user friendly*, muito devido à complexidade das operações deste negócio. Isto tornou-se então uma das principais preocupações da equipa de desenvolvimento e, uma

vez que existia contacto constante com o cliente, este dava feedback quanto à qualidade de utilização do produto desenvolvido, pois era fulcral que a plataforma criada fosse de fácil compreensão e utilização, sendo intuitiva e simples.

2.2.3 Estado de Arte em Tecnologias Relevantes

Team Foundation Server (TFS)

TFS, recentemente renomeado para *Azure DevOps Server* pela Microsoft, é um sistema de controlo de versões centralizado que permite às equipas de trabalho armazenar qualquer tipo de artefactos no seu repositório.

Para além dessa característica principal, o TFS permite ainda compilações automáticas, ajuda na gestão de projetos e de requisitos e fornece recursos para a gestão de testes (Microsoft 2019a).

Git

Git é um sistema de controlo de versões distribuído e de código aberto, que tem como principal objetivo lidar com todo o tipo de projetos, independentemente da sua dimensão, com velocidade e eficiência.

É usado maioritariamente no âmbito do desenvolvimento de software, para registar o histórico das edições de qualquer tipo de artefacto (GIT 2020).

.NET

Criada pela Microsoft, .Net é uma plataforma de desenvolvimento de código-fonte de vários tipos de aplicações. Através desta é possível a criação de vários produtos, tanto web, como também mobile, desktop, jogos, entre outros.

Esta permite a utilização de várias linguagens de programação, proporcionando acesso a diversas bibliotecas, facilitanto o seu desenvolvimento. Independentemente da linguagem escolhida, o código é executado nativamente em qualquer sistema compatível, sendo que diferentes implementações lidam com diferentes tarefas (.NET 2020):

- .NET Core: é uma implementação de .NET multiplataforma. Permite a criação de websites, serviços e aplicações consola em Windows, Linux e macOS;
- .NET Framework: implementação original da ferramenta. Permite a criação de websites, aplicações desktop, entre outras, em Windows;
- Xamarin/Mono: é uma implementação feita para correr aplicações nos principais sistemas operativos – inclui também IOS e Android.

O .NET Standard é uma especificação formal das *Application Programming Interface*s (APIs) comuns nas implementações do .NET. Isso permite que o mesmo código e bibliotecas sejam executados em diferentes implementações (.NET 2020).

Develop Framework 4 ALL (DF4A)

A DF4A tem como principal objetivo acelerar o desenvolvimento dos seus produtos web, e tem como principais componentes os *services* e a *web*.

Os serviços são responsáveis por persistir os dados na base de dados, disponibilizar serviços e processos comuns às várias aplicações da empresa e implementar a lógica de negócio. É exemplo disto a autenticação de utilizadores, a gestão de perfis, controlo de acessos, integração de sistemas externos, como envio de SMS's e Emails, entre outros.

A componente web é o motor responsável pelo *render* de páginas HTML, desenvolvido em Kendo UI (KendoUI 2020), e está também encarregue das funcionalidades dos *controllers* e das *views*.

C#

O objetivo do C# é fornecer uma plataforma simples, segura, moderna, orientada a objetos, centrada na Internet e de alto desempenho para desenvolvimento em .NET. Criada pela Microsoft, permite o desenvolvimento de uma vasta variedade de aplicações seguras e robustas, tendo como grande propósito a produtividade na elaboração de aplicações Web.

Esta tem como principais influências o C, C++ e o JAVA, e é facilmente reconhecida por qualquer pessoa que esteja familiarizada com estas linguagens (Liberty 2005).

JavaScript

JavaScript é uma linguagem de programação usada principalmente para criar páginas de web dinâmicas. Uma página da web dinâmica é aquela que incorpora efeitos como texto que aparece e desaparece, animações, ações que são ativadas ao pressionar botões e janelas com mensagens de aviso ao utilizador.

Tecnicamente, JavaScript é uma linguagem de programação interpretada, portanto, os programas não precisam de ser compilados para serem executados. Por outras palavras, programas escritos com JavaScript podem ser testados diretamente em qualquer *browser* sem a necessidade de processos intermediários (Pérez 2019).

AJAX

AJAX, ou *Asynchronous JavaScript and XML* é uma metodologia de desenvolvimento de aplicações web rápidas e iterativas, tirando partido de pedidos assíncronos.

É uma das soluções mais procuradas por programadores por ter várias vantagens, tais como:

- Atualizar uma página da web sem recarregar novamente a página completa;
- Solicitar dados de um servidor - após o carregamento da página;
- Receber dados de um servidor - após o carregamento da página;
- Enviar dados para um servidor - em *background*.

Apesar do seu nome aparentar a obrigatoriedade do uso de XML, a utilização de JSON é muito comum (Mahemoff 2006).

jQuery

jQuery é uma biblioteca de JavaScript rápida, pequena e rica em recursos. Este facto torna a manipulação de ficheiros HTML, a manipulação de eventos e AJAX muito mais simples, criando uma API fácil de usar que irá ser funcional em vários *browsers* (Foundation 2020).

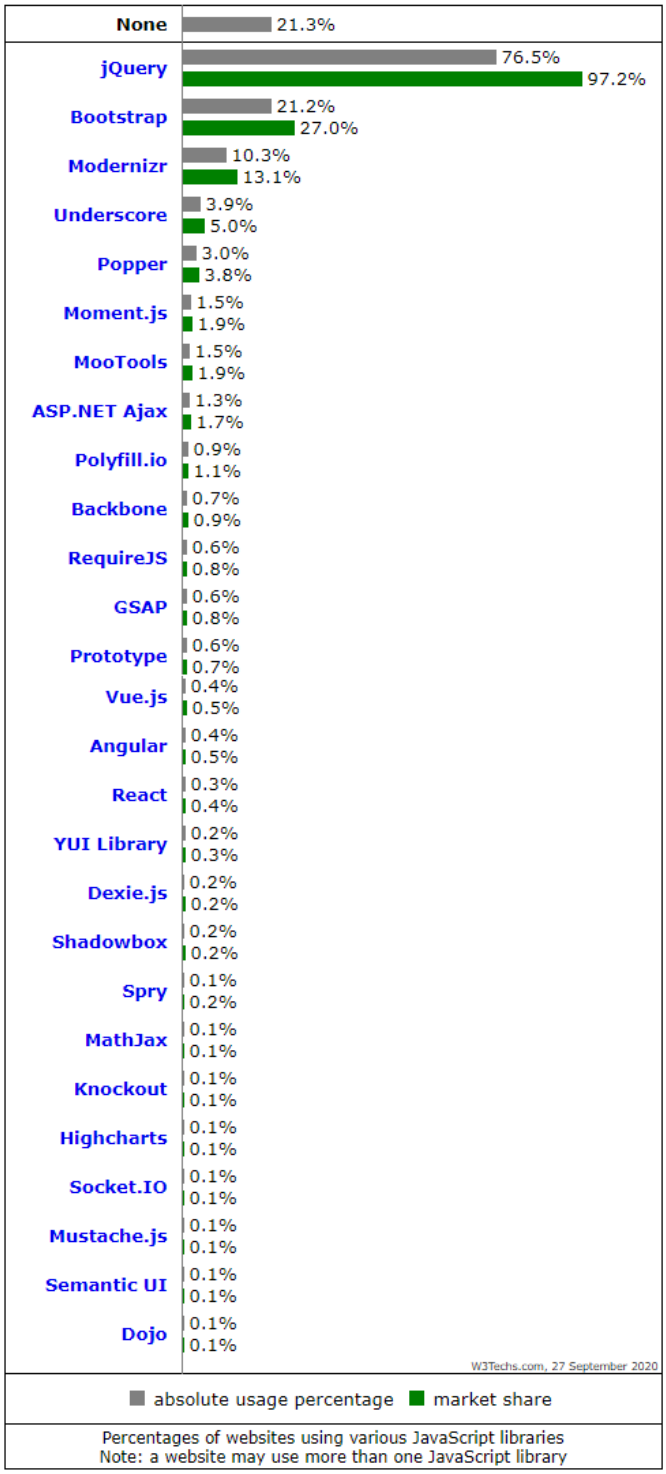

Figura 2.3: Percentagem de sites que usam as várias bibliotecas de JavaScript [Fonte: (w3techs 2020)].

Como visível na figura 2.3, com uma percentagem de utilização de 76.5%, esta biblioteca é de longe a biblioteca de JavaScript mais usada e o seu uso continua em crescimento, embora mais lentamente do que nos anos anteriores.

SQL Server Management Studio (SSMS)

Desenvolvido pela Microsoft, o SQL Server Management Studio (SSMS) é um ambiente de fácil integração com .Net, para gerir qualquer infraestrutura SQL. Geralmente o SSMS é usado para aceder, configurar e desenvolver todos os componentes do SQL Server.

Este combina um amplo grupo de ferramentas gráficas com diversos editores de script avançados para fornecer acesso ao SQL Server a programadores e administradores de bases de dados (Microsoft 2019b).

Oracle SQL Developer

O Oracle SQL Developer é uma versão gráfica do SQL *Plus que oferece aos programadores de bases de dados uma maneira conveniente de executar tarefas básicas.

Neste é possível:

- Navegar, criar, editar e eliminar objetos da base de dados;
- Executar instruções e scripts SQL;
- Editar e Depurar código PL/SQL;
- Manipular e exportar dados;
- Visualizar e criar relatórios.

Este ambiente de desenvolvimento oferece aos seus utilizadores três tipos de interfaces gráficas: uma de Desktop, capaz de correr em qualquer sistema operativo; uma de *Browser*; e uma para linhas de comandos.

Apesar de todos estes recursos oferecidos, a integração com a linguagem .Net é particularmente difícil e complexa (Murray 2007).

2.2.4 Análise de tecnologias

As tecnologias anteriormente referidas foram testadas e estudadas e deste período de análise foi possível tirar ilações, que permitiram entender quais as que se adequavam melhor às necessidades do projeto. Importa ainda referir que a CPCIT4ALL-Lda. se encontra relacionada com a empresa Microsoft, sendo que é um parceiro certificado *Microsoft Gold Certified Partner*. Por este motivo, as tecnologias desenvolvidas pela Microsoft terão prioridade numa situação de nível de igualdade de benefícios.

TFS vs. GIT

Analisando os dois serviços de controlo de versões, referidos , TFS e GIT, a principal diferença é o facto do primeiro ser um sistema centralizado e o segundo um sistema distribuído.

Utilizando o TFS os repositórios são armazenados num servidor central e cada programador faz *check-out* de uma cópia do trabalho, que não é mais do que uma captura instantânea de código num momento específico. Com isto, o ficheiro do qual foi feito *chek-out* torna-se indisponível a qualquer outra pessoa que o queira alterar, evitando assim conflitos dentro de cada ficheiro.

Por outro lado, o GIT usa a técnica de clonar repositórios, permitindo à equipa de trabalho ter um clone do repositório inteiro na sua máquina, incluindo todo o histórico até ao momento.

Para a escolha entre estes, um fator de peso foi o facto do TFS ser um serviço disponibilizado pela Microsoft, e sendo que o IDE que foi usado foi o Microsoft Visual Studio, e apesar de ambos os sistemas serem compatíveis, torna-se relevante o facto da empresa ser parceira da Microsoft. Posto isto, optou-se então pelo uso do TFS como serviço de controlo de versões.

.NET Core vs. .NET Framework

Foram ainda apresentadas várias plataformas de desenvolvimento criadas pela Microsoft, .NET Core e .NET Framework. Tendo em conta que a empresa utiliza uma framework interna para ajudar e acelerar o desenvolvimento de aplicações Web (DF4A), desenvolvida em .Net Framework, e apesar de ser imposto o uso de .NET Framework, é pertinente avaliar cada uma para perceber qual melhor se adapta ao projeto.

Posto isto, tanto .NET Core como .NET Framework partilham muitos dos mesmos componentes e, como dito anteriormente, é possível a partilha de código entre as duas. No entanto, existem diferenças significativas e fundamentais para que a escolha seja adequada (Microsoft 2020).

O uso de .NET Core é recomendado quando:

- Há necessidade de usar várias plataformas;
- Há o recurso a micro serviços;
- Há a utilização de containers Docker;
- Existe a necessidade de alta performance e sistemas escaláveis.

O uso de .NET Framework é recomendado quando a aplicação:

- Usa, atualmente, o .NET Framework (recomenda-se estender em vez de migrar);
- Usa bibliotecas ou packages NuGet não disponíveis para .NET Core;
- Usa tecnologias .NET não disponíveis para .NET Core;
- Usa uma plataforma não suportada pelo .NET Core.

Tendo em conta o referido, o .NET Core realmente oferece benefícios significativos para novas aplicações, porém, e sendo a DF4A desenvolvida em .NET Framework, a segunda torna-se a escolha natural para o desenvolvimento do projeto, uma vez que é recomendada a continuidade da sua utilização em detrimento da migração para .NET Core. Além disto, é importante salientar que no contexto atual, os benefícios do .NET Core não são indispensáveis, e por isso, não se justifica uma mudança na escolha de plataforma.

Tomada esta decisão, o uso da linguagem C# torna-se inevitável no desenvolvimento do projeto, com auxílio de JavaScript para o *scripting* das páginas Web e das suas bibliotecas, AJAX e jQuery.

Microsoft SQL Server Management Studio vs. Oracle SQL Developer

Por fim, é necessário decidir qual ambiente para gerir base de dados que deve ser utilizado. Tendo em conta os ambientes integrados apresentados, SSMS e Oracle SQL Developer, e sabendo que a base de dados utilizada pela empresa é SQL Server, torna-se relevante analisar a capacidade de suportar bases de dados SQL Server e a integração com .NET.

Apesar de ambos os ambientes suportarem SQL Server, apenas o SSMS integra facilmente com .NET, tornando assim a escolha óbvia. Além disto, é também importante referir que nesta escolha pesou também o facto da empresa ser parceira certificada da Microsoft.

2.3 Análise de Valor

A análise de valor é um processo organizado de análise e avaliação de um produto ou serviço, e tem como principal objetivo minimizar o custo de um produto/serviço e, ao mesmo tempo, aumentar o seu valor sem perder a qualidade.

Durante o desenvolvimento de novos produtos no mercado existe um elevado risco no que toca a orçamento e aceitação por parte de possíveis clientes. Com o intuito de reduzir este risco, existe a preocupação de investir numa melhoria das técnicas utilizadas no Processo de Desenvolvimento de Produtos. Este processo está separado em três partes distintas: Pré-desenvolvimento, desenvolvimento e pós desenvolvimento.

O valor é um conceito subjetivo que é interpretado de diferentes maneiras pelos clientes e depende do contexto. Em geral, o valor pode ser definido como uma necessidade, interesse ou preferência. A criação de valor é difícil de entender em teoria, mas pode ser retomada como o equilíbrio entre benefícios e sacrifícios na perspetiva dos clientes.

Esse conjunto de benefícios e sacrifícios influenciam o valor para o cliente, sendo que diferentes clientes têm diferentes valores dos mesmos produtos ou serviços.

Deste modo, o valor é a avaliação geral que um cliente dá a um produto/serviço, com base nas perceções do que ele recebe e do que ele dá. Quando esses benefícios e sacrifícios são bem combinados ou há uma quantidade considerável de benefícios, e o cliente vê vantagens na oferta de uma organização, há Valor para o Cliente (VC).

Neste subcapítulo são apresentados os modelos que incidem sobre este processo, detalhando o valor da solução, e além disto, através do modelo CANVAS, é apresentado o modelo de negócio.

2.3.1 Modelo Fuzzy Front End

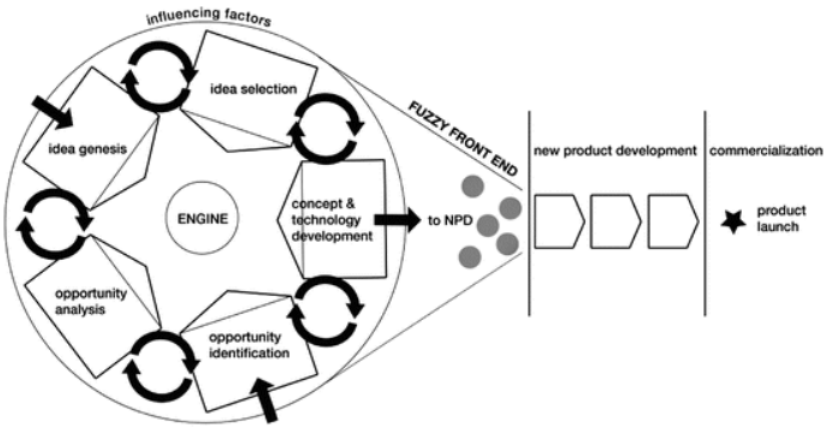

Figura 2.4: Processo de inovação [Fonte: (Kurt e Michael 2014)].

O processo indicado na figura 2.4 está dividido em 3 fases principais, Fuzzy Front End (FFE), New Product Development (NPD) e Comercialização.

O FFE consiste na fase experimental da inovação, sendo que a qualidade do trabalho desenvolvido aqui é decisiva para o sucesso da solução e tem como principais objetivos identificar possíveis oportunidades de negócio (ESCHBERGER 2018).

O NPD trata-se de um método que tem como principal objetivo aperfeiçoar o trabalho feito no FFE.

Por fim a Comercialização é a fase em que é feita a promoção e venda do produto criado.

2.3.2 Modelo New Concept Development

De acordo com (Koen et al. 2001), o modelo New Concept Development (NCD) fornece um bom resumo das atividades que ocorrem anteriormente à fase de desenvolvimento do produto

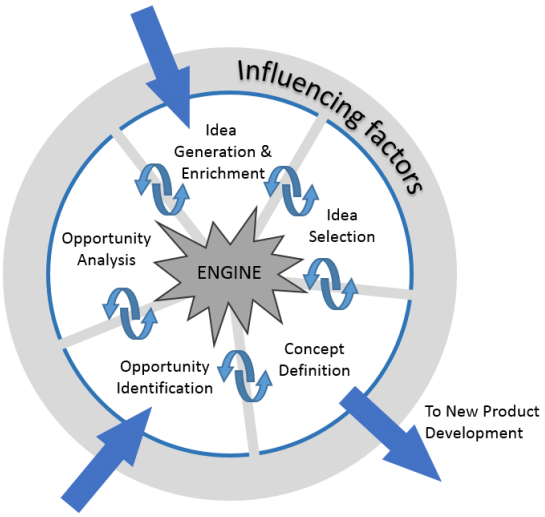

Figura 2.5: Modelo NCD [Fonte: (Sketchbubble 2017)].

O NCD está dividido em três partes, o *Engine* (Motor), as cinco áreas de atividade, relacionadas com o *Front End of Innovation* (FFE) (área interna), e os Fatores Ambientais, representados na figura como *Influencing Factors*

Na figura 2.5 é possível observar que este modelo é representado com uma forma circular, visando sugerir que as ideias devem fluir e iterar entre os cinco elementos, devendo prosseguir de forma não sequencial, como mostrado pelas setas em loop entre os elementos.

É expectável que as interações e a mistura entre as áreas de atividade, os fatores ambientais e o motor ocorram continuamente.

De seguida serão explicados cada um dos elementos que compõem o NCD:

- **Engine**: Representa a liderança, cultura e estratégia de negócios da organização;
- **Cinco Áreas de Atividade**:
  - **Opportunity Identification**: Identificação de oportunidades tecnológicas que podem surgir para atingir os objetivos. Nesta área as fontes e os métodos usados pela empresa podem variar, entre formais e informais. Como oportunidade,

define-se qualquer necessidade de negócio, ou necessidade técnica, de uma empresa que responda a um problema, ou ameaça, que traga vantagens competitivas (Koen et al. 2001);

- **Opportunity Analysis**: Análise das oportunidades identificadas. O nível de esforço depositado em cada uma das oportunidades depende da atractividade da oportunidade em si, da dimensão do esforço de desenvolvimento futuro e da tolerância ao risco de quem toma as decisões (Koen et al. 2001);
- **Idea Generation and Enrichment**: Pode-se descrever como sendo o nascimento, desenvolvimento e amadurecimento da oportunidade numa ideia concreta. Trata-se de um processo evolutivo, onde há combinação de pensamentos até que se consegue chegar a uma solução sólida. Uma nova ideia pode surgir interna ou externamente, por exemplo, através de um fornecedor, ou de um cliente com uma solicitação. (Koen et al. 2001);
- **Idea Selection**: Escolha das ideias que são viáveis de desenvolver. Esta seleção é feita através do risco de implementação tecnológica, riscos de investimento, análises de competição, capacidade organizacional e o retorno possível (Koen et al. 2001);
- **Concept Definition**: Fase em que o plano de negócio é desenvolvido com base em estimativas de mercado, necessidades dos clientes, requisitos de investimento, avaliações concorrentes, entre outras. Este conceito pode ser apresentado de uma forma escrita ou visual, possuindo um nível de formalidade consoante a natureza da oportunidade. (Koen et al. 2001);

- **Fatores Ambientais**: Estes fatores são relativamente incontroláveis e influenciam o Motor e as cinco áreas de atividade (Koen et al. 2001). Consistem em:
  - Capacidade de organização;
  - Ameaças de competidores;
  - Modas de compra dos clientes;
  - Mudança de regulamentos;
  - Conhecimento e capacidade das tecnologias;

Aplicando este modelo ao projeto apresentado neste documento, temos que as cinco áreas de atividade são:

- **Identificação da oportunidade**: A implementação da solução apresentada surgiu pelo facto de não existir um sistema que proporcionasse aos comerciais e às suas equipas a segurança necessária para deixar de usar formulários em papel. No momento de escrita desta dissertação, que a comercializadora tenha conhecimento, existe apenas uma outra plataforma que faz este tipo de serviço, mas que não lhes conferiu confiança suficiente para a escolherem;
- **Análise da oportunidade**: Considerando os dados apresentados no Capítulo 1.2 é percetível a necessidade de um sistema que venha auxiliar as equipas comerciais de uma empresa de comercialização de Gás Natural. Fazendo uma análise ao mercado atual constata-se uma enorme falta de oferta deste tipo de sistemas, sendo que os que existem, ou estão obsoletas ou então não induzem confiança aos seus utilizadores;

- **Elaboração de ideias e enriquecimento**: No caso deste projeto, a ideia foi inicialmente lançada por um cliente da CPCIT4ALL, tanto em momentos formais e em momentos mais informais, pois tinha a necessidade de acelerar o processo de criação de contratos dos seus consumidores e de organizar toda a informação destes;
- **Seleção de ideias**: Para implementar um projeto deste tipo, é normal existirem diferentes métodos/abordagens e tecnologias. É importante analisar e comparar esses métodos e tecnologias e decidir quais são mais adequados para a empresa. Posto isto, e analisando as ideias anteriores, foi feita a seleção que seria mais aconselhável.
- **Definição de conceito**: Com este sistema pretende-se aumentar a produtividades dos trabalhadores da comercializadora e, ao mesmo tempo, evitar perdas de oportunidades de negócio. O seu uso poderá evitar grandes períodos de espera, tornando o processo da adesão de um novo cliente muito mais fluído e moderado. Para além disto, o utilizador deste sistema vai ter ao seu dispor acesso a informação mais rápida e controlada, evitando enganos indevidos.

2.3.3 Valor para o cliente e Valor percecionado

Valor para o Cliente

Segundo (Reis 2016), o valor de um produto pode ser definido como sendo a razão entre os benefícios e os custos/sacrifícios.

$$Valor = \frac{beneficios}{custos} = \frac{praticos + intangiveis}{monetarios + tempo + energia + intangiveis}$$

Tony Woodall apresenta uma perspectiva longitudinal do VC no artigo *"Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis"*, como mostra a Figura 2.6. Ele divide o VC em quatro posições temporais.

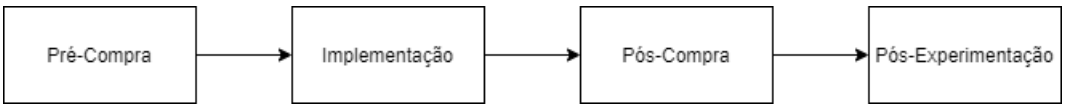

Figura 2.6: Perspetiva Longitudinal [Adaptado de: (Woodall 2003)].

No âmbito do projeto que está a ser elaborado, a tabela a baixo representa os benefícios e sacrifícios necessários em cada uma das fazes descritas por Woodall:

Tabela 2.2: Análise de Valor - Benefícios vs. Sacrifícios

	Benefícios	**Sacrifícios**
Pré-Compra	Benefícios Operacionais Benefícios do Produto	Tempo Esforço Investimento Inicial
Implementação	Confiança	Custo Aumento do Grupo de Trabalho Força de trabalho Tempo
Pós-Compra	Capacidade de Resposta Competência técnica	Tempo/Energia/Esforço
Pós-Experimentação	Qualidade do Serviço Economização de Tempo	

As fases acima apresentadas representam os benefícios e sacrifícios, e como vemos na tabela cada uma delas contém os dois:

- **Pré-Compra**: Esta é a fase em que o Cliente tem de reconhecer que o sistema pode acrescentar valores ao seu negócio, tanto operacionais como de produto. Porém, nesta fase é imperioso ter em conta o investimento necessário para o desenvolvimento deste projeto, quer a nível monetário, de tempo e de esforço.
- **Implementação**: Sendo uma das fases mais críticas para o cliente, consoante a complexidade do produto a ser desenvolvido, pode existir a necessidade de se aumentar a força de trabalho aumentando assim os custos com os recursos humanos.

  É também previsível, que nesta fase seja necessário introduzir o novo sistema aos seus utilizadores, causando um sacrifício ao nível de tempo. Esta formação inclui os gestores de equipas e os seus respetivos comerciais.
- **Pós-Venda**: É a fase em que o sistema é efetivamente utilizado e quando o cliente confirma o valor do sistema que adquiriu. No caso deste projeto, prevê-se que este seja usado para criar e gerir contratos, incorporando componentes automáticas que ajudem os seus utilizadores a fazer o seu trabalho de forma mais rápida, aumentando assim a sua produtividade.
- **Pós-Experimentação**: Neste momento o cliente já está familiarizado com o sistema, não necessitando de acompanhamento, apresenta confiança no sistema e a capacidade de resposta às suas necessidades é também reconhecida.

Valor Percecionado

Segundo (Fernández e Bonillo 2017), valor percecionado pode ser definido como o valor que o cliente atribui a um determinado produto para o seu negócio, tendo sempre uma posição mais cuidada relativamente ao preço do serviço. Este difere do valor percecionado pelo produtor, pois este atribui valor à qualidade do produto.

Relativamente aos possíveis clientes do projeto em análise, estes podem ter diferentes opiniões. Enquanto que para uma comercializadora que tenha uma grande afluência de contratos, torna-se uma mais valia possuir este sistema, uma comercializadora de menor escala

pode conseguir organizar e gerir os seus contratos em papel. Porém, em ambos os casos, este serviço poderá vir a ser útil na redução de custos com materiais e de tempo, resultando na consequente redução de despesas para a empresa.

2.3.4 Proposta de valor

A proposta de valor é essencial para alertar novos clientes sobre o valor dos produtos e serviços de uma organização. Para os autores Alexander Osterwalder e Yves Pigneur, é a visão geral dos produtos e serviços de uma empresa, que são percecionados como valiosos para o cliente (Pigneur e Osterwalder 2003). A proposta de valor deve deixar claro qual é o produto/serviço que se está a fornecer, quais os clientes alvo, quais os benefícios que o cliente vai ter ao usar o produto/serviço e de que forma este difere daqueles que já existem no mercado.

O produto apresentado nesta dissertação, é um sistema de gestão de contratos de Gás, onde se inclui a monitorização de produtividade de equipas, bem como mecanismos automáticos que fazem as ligações necessárias com entidades externas ao comercializador, proporcionando à empresa um grande aumento na produtividade.

Como principal cliente alvo surgem as comercializadoras, sendo que o sistema é adaptável a todas as empresas que desejem gerir contratos relacionados com Gás. Os clientes podem gerir e organizar todos os contratos que possuem, e ter acesso fácil e contínuo a toda a informação que necessitam.

Além de aumentar o grau de confiança do que estão a implementar promovido pelos mecanismos automáticos que visão evitar o erro humano, existirá um aumento correspondente na produtividade da empresa, devido à diminuição dos intervalos de espera entre entidades externas.

A monitorização de comerciais e de equipas, pode também ser vantajosa, permitindo identificar quem tem demasiada carga de trabalho e não consegue escoar todo o trabalho atribuído, permitindo aos gestores da empresa corrigir estas falhas prontamente.

Apesar de já existirem programas concorrentes neste setor, nota-se um descontentamento face a estes, pelo facto da informação não estar concentrada no mesmo local, ou então porque simplesmente o sistema não cumpre com os objetivos dos utilizadores.

O facto de na aplicação desenvolvida a informação estar toda concentrada no mesmo local e ter fácil acesso, acrescenta ao seu valor. Aqui os gestores têm acesso a toda a informação do contrato, bem como do cliente, para além de que inúmeros campos são preenchidos automaticamente e/ou filtrados de modo a evitar erros humanos. As situações descritas atrasavam sempre a validação dos contratos aquando a comunicação com as entidades externas à comercializadora.

2.3.5 Modelo CANVAS

O Modelo CANVAS é uma das principais ferramentas de estratégia que permite desenvolver modelos de negócio de modo a que este seja visto numa página apenas (Pereira 2016). É composto por 9 blocos que comunicam e se complementam entre si:

- **Parceiros Chave**: Especifica quais os parceiros mais importantes para fornecer o serviço/produto;

- **Atividades Chave**: Atividades necessárias para manter o interesse dos clientes pelo negócio;
- **Recursos Chave**: Recursos necessários para a proposta de valor;
- **Proposta de Valor**: Define os problemas a ser respondidos e quais as necessidades a ser satisfeitas;
- **Relações com clientes**: Esclarecimento do tipo de relação que o cliente espera da empresa;
- **Canais de distribuição**: Canais usados para chegar ao cliente;
- **Segmentos de Mercado**: Definição dos clientes mais importantes do negócio;
- **Estrutura de custos**: Indica quais os custos mais importantes para o modelo de negócio;
- **Fontes de receita**: Forma de como os clientes vão pagar.

Tabela 2.3: Modelo CANVAS

<table>
<tr>
<td rowspan="2">Parceiros Chave

-Distribuidoras de Gás;
-OLMC;
-REN.</td>
<td>Atividades Chave

-Manutenção de software;</td>
<td rowspan="2">Proposta de Valor

-Gestão de Contratos;
-Processos automáticos;
-Redução de custos;
-Acesso a informação rápido e fácil;
-Monitorização de equipas</td>
<td>Relações com Clientes

-Assistência pessoal dedicada;
-Suporte pós-venda;
-Feedback dos utilizadores</td>
<td rowspan="2">Segmentos de Mercado

-Comercializadoras de Gás</td>
</tr>
<tr>
<td>Recursos Chave

-Internet;
-Plataformas tecnológicas;</td>
<td>Canais

-Plataforma online a exemplificar como o produto funciona;
-Demonstrações em clientes</td>
</tr>
<tr>
<td colspan="3">Estrutura de custos

-Equipa de desenvolvimento e manutenção de software;
-Marketing</td>
<td colspan="2">Fontes de receita

-Venda do sistema;
-Serviços de desenvolvimento à medida</td>
</tr>
</table>

Capítulo 3

Descrição Técnica

Este capítulo visa oferecer uma visão não apenas do produto desenvolvido, mas também do processo de engenharia adjacente à sua implementação. No presente capítulo serão descritas as etapas de análise, design e desenvolvimento do projeto.

Vai também ser exposta a documentação de análise e design produzida, o processo de análise de requisitos, conceitos de domínio e outros artefactos arquitetónicos, assim como as alternativas existentes e a respetiva justificação das escolhas feitas.

3.1 Engenharia de Requisitos

Na engenharia de requisitos, é apresentada a análise de requisitos não funcionais e funcionais que surgiram no projeto. Os requisitos são classificados de acordo com cada uma das diferentes categorias do modelo FURPS+. A adoção desse modelo está relacionada ao seu reconhecimento histórico e aprovação.

3.1.1 Requisitos não funcionais

Na classificação FURPS +, existem requisitos não funcionais, incluindo usabilidade, confiabilidade, desempenho, capacidade de suporte, design, implementação, interface e restrições físicas (PSPIus 2020).

Usabilidade

A usabilidade captura e avalia os requisitos com base na interface do usuário (por exemplo, acessibilidade, estética e consistência) (PSPIus 2020). No projeto em questão, existem diferentes requisitos não funcionais relacionados à usabilidade e são:

- A resposta do sistema deve ser a mais rápida possível;
- Interface simples, intuitiva e eficiente;
- O sistema como um todo deve ser simples e confortável de usar;

Desempenho

Os requisitos de desempenho estão relacionados com a *performance* do sistema, isto é, quão bem este executa determinadas funções sob condições específicas. No âmbito deste projeto, os requisitos impostos pelo cliente foram:

- Tempo médio de resposta dos endpoints do serviço REST inferior a 1 segundo;

- Capacidade de processamento até 10 pedidos por segundo nos endpoinst do serviço REST;
- Tempo médio de resposta dos endpoints do serviço *Simple Object Access Protocol* (SOAP) inferior a 1 segundo;
- Tempo médio do carregamento das páginas da interface web inferior a 5 segundos;
- O sistema deverá suportar múltiplos acessos concorrentes sem que coloque em causa a produtividade dos utilizadores.

Fiabilidade/Segurança

As restrições de confiabilidade incluem aspetos como disponibilidade, precisão, capacidade de recuperação e tempo médio entre falhas (PSPlus 2020). No contexto deste projeto, diferentes requisitos de confiabilidade estão presentes, a saber:

- O lado do servidor deve permanecer disponível e funcional após falhas;
- Fiabilidade dos dados enviados para OLMC superior a 90%;
- O lado do servidor deve ser resiliente garantindo alta disponibilidade, seguindo um Service Level Agreement (SLA) de *two nines* (99%) (Myerson 2013);
- O sistema deve estar acessível vinte e quatro horas por dia, sete dias por semana;
- As falhas de um componente do lado do servidor não devem ser propagadas para outros componentes;

Suporte

As restrições de capacidade de suporte estão essencialmente relacionadas a limitações de linguagem, adaptabilidade, manutenção, compatibilidade e escalabilidade (PSPlus 2020). Para o desenvolvimento deste projeto, foram capturados diferentes requisitos relacionados à compatibilidade, a saber:

- O sistema deve ter a capacidade de expansão e permitir a adição de novos módulos, quando necessário;
- O sistema deve ser desenvolvido para integrar suporte futuro a outras aplicações, com o mínimo impacto nas funcionalidades já implementadas, adotando bons princípios de manutenção, adaptabilidade e configuração, que permitam a sua evolução.

Restrições de Design

Uma restrição de design limita a maneira como o sistema é delineado (por exemplo, o uso de uma base de dados relacional é necessário) (PSPlus 2020). Para o desenvolvimento do protótipo arquitetónico em questão, as restrições relacionadas ao design da solução foram:

- O sistema deve ser desenvolvido através de um processo iterativo e incremental;
- Uma base de dados relacional deve ser usada para complementar com os módulos já existentes.

Restrições de Implementação

Este tipo de restrição afeta a maneira como o sistema é construído e também pode impactar o design do sistema (PSPlus 2020). As seguintes restrições de implementação foram detetadas:

- O sistema deve ser desenvolvido usando a *framework* usada na CPCIT4ALL;
- O sistema deve suportar o protocolo REST para comunicações com a API de Switch;
- Durante o desenvolvimento do projeto, devem ser usados bons princípios de codificação;

Restrições de Interface

Restrições ao nível da interface são aquelas que afetam a maneira como a comunicação é feita com outro componente. Um tipo de restrição de interface é, por exemplo, restringir o protocolo de comunicação (PSPlus 2020). As seguintes restrições foram capturadas:

- A comunicação entre o switch e o portal deve ser feita através do protocolo REST;
- A comunicação entre o BC e o portal deve ser feita através do protocolo SOAP;

3.1.2 Requisitos Funcionais

Os requisitos funcionais definem as principais características do sistema. Para o caráter arquitetural assumido pelo projeto, esses requisitos assumem apenas os requisitos associados à implementação da prova de conceito. De acordo com a metodologia Kanban, os requisitos que constituíam o *backlog* do produto são referidos na figura 3.1:

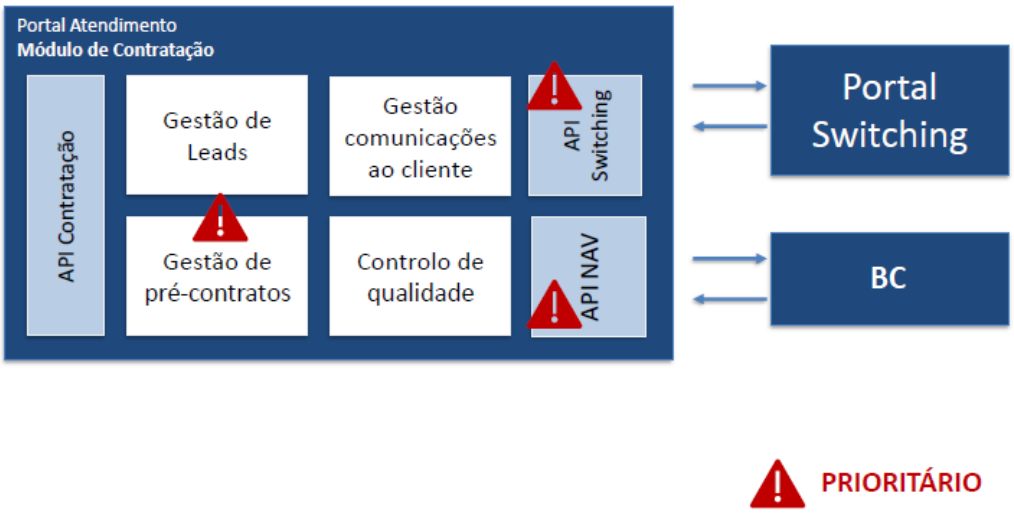

Figura 3.1: Macro Funcionalidades do módulo a desenvolver

Gestão de Leads

- **GL01**: O sistema deve suportar a criação de uma lead (oportunidade) que se resume a um subconjunto de campos de um pré contrato;

- **GL02**: O módulo de contratação deve permitir registar uma lead através de um formulário próprio disponibilizado ao utilizador, ou através da invocação de uma operação na API de contratação.
- **GL03**: Uma lead deve ter a indicação da sua origem;
- **GL04**: Uma lead deve ter um estado associado: "nova", "convertida para pré contrato" ou "anulada";
- **GL05**: O sistema deve permitir converter uma lead num pré contrato, vertendo todos os dados que nela constem no pré contrato destino.

Gestão de Pré-Contratos

- **GPC01**: Pré contrato é a entidade que representa o contrato antes da sua integração para BC, logo deverá conter toda a informação que a entidade "Contrato" do BC requer;
- **GPC02**: O módulo de contratação deve permitir registar um pré contrato através de um formulário próprio disponibilizado ao utilizador, ou através da invocação de uma operação na API de contratação;
- **GPC03**: O pré contrato inclui a seguinte informação:
  - Informação de registo e origem do pré contrato;
  - Informação sobre o cliente;
  - Informação sobre faturação (Periocidade, Morada envio/email);
  - Informação sobre o método de pagamento;
  - Informação sobre as condições comerciais contratadas;
  - Documentos digitalizados associados à contratação;
  - Informação sobre o local de consumo;
  - Informação da cadeia de medida (Preenchida após a conclusão de switch).

API de Contratação

- **APICont01**: Criar uma lead com informação básica, sendo que esta operação será disponibilizada a websites de parceiros externos ou a sites de campanhas especificas criadas pela comercializadora;
- **APICont02**: Criar um pré contrato para submissão para switching;

API de Switching

- **APISwitch01**: Iniciar processo de switching de entrada. O Portal de switching disponibilizará um webservice para o início de processos de ativação de energias. O portal de atendimento deverá iniciar este processo quando tiver o pré contrato em estado adequado a iniciar switching;

- **APISwitch02**: Após iniciar o processo de switching de entrada o portal de atendimento deverá permitir ao utilizador consultar os eventos de switching que tiverem ocorrido para se inteirar do estado do processo

API de BC

- **APIBC01**: Consulta de estado do CUI: O BC disponibilizará um webservice para verificar se existe um contrato ativo para um CUI. O portal de atendimento deverá permitir o registo de um pré contrato caso não existam contratos ativos;
- **APIBC02**: Consulta de dados de cliente: O BC disponibilizará um webservice que permita consultar os dados gerais de um cliente pelo NIF. O portal de atendimento poderá utilizar a informação para pré preencher os dados do cliente num pré contrato;
- **APIBC03**: Consulta de campanhas disponíveis: O BC disponibilizará um webserice que permita consultar as campanhas comercias ativas no registo de uma lead ou pré contrato. O portal deverá confrontar as condições contratuais da campanha com os dados fornecidos no pré contrato e, se for o caso, bloquear o registo do mesmo;
- **APIBC04**: Envio de pré contrato ativo para BC. O BC disponibilizará um webservice que permita o envio de pré contratos com o processo de switch concluído com sucesso para BC. O webservice , além da informação do pré contrato, deverá receber também a informação da cadeia de medida proveniente de Switch. O BC deverá criar o contrato com toda a informação necessária e o Portal deverá registar que o pré contrato foi enviado para BC, caso não tenham ocorrido erros.

Gestão de comunicações ao cliente

- **GCC01**: O sistema deve permitir estabelecer um conjunto de notificações e envio de SMS/email para comunicação com os clientes sobre a evolução do seu processo de contratação;

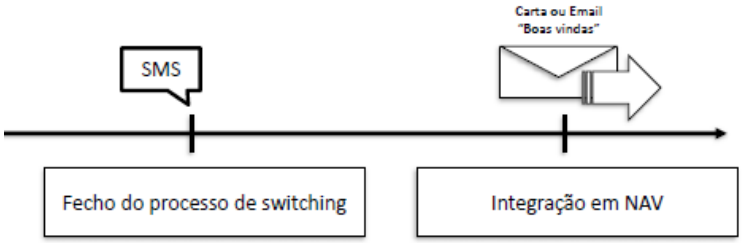

Figura 3.2: Comunicações com Clientes

- **GCC02**: Sempre que o cliente tiver fatura eletrónica ativa, o email deve ser o meio de comunicação preferencial. Nos restantes casos, o SMS deve ser o mecanismo a utilizar;
- **GCC03**: O Sistema deve enviar uma versão digital do contrato (condições gerais + condições particulares) por email ou carta ao cliente.
- **GCC04**: No processo de contratação por atendimento telefónico, o sistema deve permitir o envio de um SMS ou Email a solicitar a aceitação das condições propostas, mantendo a evidência da sua aceitação.

Controlo de Qualidade

- **CQ01**: Anexar ficheiros áudio (chamadas telefónicas) a pré contratos para futura auditabilidade;
- **CQ02**: Gestão de um estado de controlo de qualidade que indique se o registo foi alvo de controlo de qualidade, quem e quando o executou e eventuais motivos de NOK em qualidade;
- **CQ03**: Disponibilizar uma listagem de pré contratos que permita filtrar por entidade que realizou o pré contrato, datas e estados de processos de controlo de qualidade para selecionar amostras que são alvos de controlo;
- **CQ04**: Permitir a criação de alertas/ reminders para situações em que o controlo de qualidade com o cliente não foi possível de realizar.

Controlo de Acessos

- **CA01**: O portal de atendimento deve prever uma gestão de perfis que permita aos utilizadores ter acesso apenas a este módulo de contratação (para utilização das equipas comerciais);
- **CA02**: No campo das permissões o sistema deve estar estruturado em equipas e garantir a segregação:
  - **Nível de gestor**: em que o utilizador vê os seus registos e os registos efetuados pela sua equipa;
  - **Nível de comercial**: em que o utilizador vê apenas os seus registos.
- **CA03**: Em qualquer dos casos uma equipa não deve, em nenhuma circunstância, ter acesso aos registos de outra equipa.

3.2 Análise e Design

Optou-se por descrever a análise e design usando o conhecido Modelo 4+1 de Arquitetura de Software. Devido ao fato, de ser possível ter várias vistas simultâneas da solução, esta abordagem *"permite abordar separadamente as preocupações das várias 'partes interessadas' da arquitetura - usuário final, programadores, engenheiros de sistemas, gerentes de projeto, etc., e para lidar separadamente com os requisitos funcionais e não funcionais. Cada uma das cinco vistas é descrita, juntamente com uma notação para capturá-la. As vistas são projetadas usando um processo de desenvolvimento iterativo centrado na arquitetura, orientado por cenário"*, como podemos ver na figura 3.3 (Kruntchen 1995).

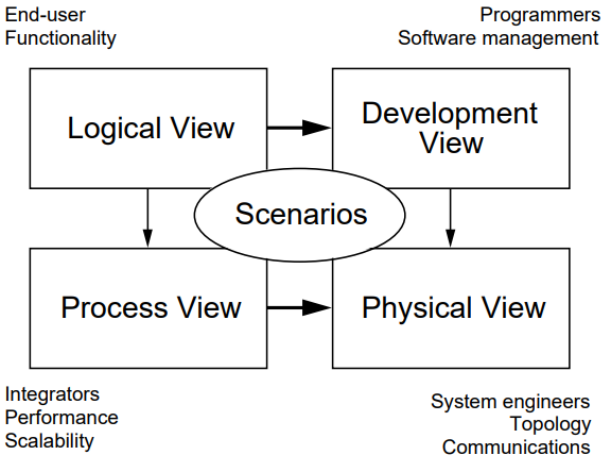

Figura 3.3: Vista do modelo arquitetónico "4+1"[Fonte: (Kruntchen 1995)]

Com base nesta abordagem, nas subseções a seguir será delineado o processo de análise e design, primeiro com uma visão geral da granularidade do sistema e, em alguns casos, mais especificamente os componentes mais relevantes.

Importa referir que o módulo desenvolvido está contido dentro de um projeto já em curso, ou seja, a arquitetura a ser utilizada já estava previamente definida, aliás era um dos requisitos usar a *framework* desenvolvida pela empresa.

3.2.1 Vista de Casos de Uso

Para começar a analisar o sistema, primeiro é necessário identificar os principais casos de uso a que é necessário oferecer suporte.

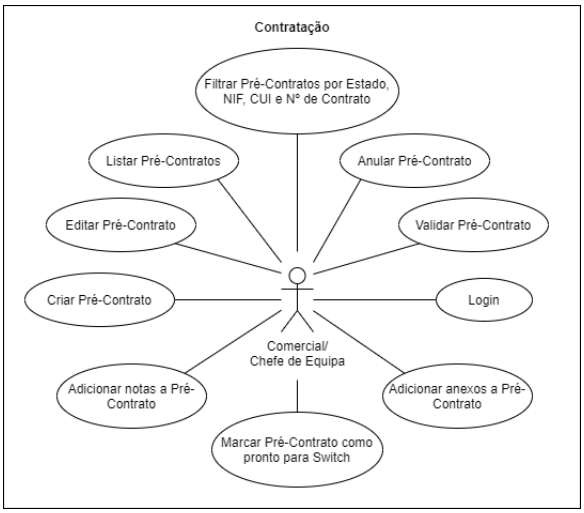

Figura 3.4: Diagrama de Casos de Uso Contratação

Figura 3.5: Diagrama de Casos de Uso BackOffice

Nas figuras 3.4 e 3.5 vemos como os principais participantes irão interagir com a solução e as extensões dessas interações.

Um Comercial ou Chefe de Equipa, figura 3.4, pode criar e listar os pré contratos que lhe pertencem, sendo que o segundo vai poder ver os pré contratos dele e os dos comerciais da sua equipa. Para além disso, e caso o estado do pré contrato seja "Em Validação"ou "Erro de Switch", existe também a funcionalidade de editar e anular.

A qualquer momento é possível adicionar anexos ou notas a um pré contrato. Quando este se encontra no estado "Em Validação", ele pode ser validado, tal consiste na verificação de que todos os campos necessários para o envio de switch estão preenchidos, e caso esteja válido, o seu estado pode ser alterado para "Pronto para Switch".

Quanto ao BackOffice existe apenas um tipo de utilizadores que têm acesso, que são os Gestores da Contratação. Aqui, o utilizador vai poder criar, editar e apagar equipas, assim como adicionar, editar ou eliminar membros a estas.

Relativamente aos agendamentos vai a ser possível listar, adicionar comentários, listar histórico e filtrar por estado, estado do contrato, Código Universal da Instalação (CUI) e Nº de Contrato.

Para finalizar, e no que diz respeito aos pré contratos, estes vão poder ser listados, e aqui irão aparecer todos os pré contratos criados até ao momento. Vai ser permitido filtrar por estado, NIF, CUI, Comercial e Nº de Contrato, bem como vão poder ser anulados ou

editados. Existem ainda as funcionalidades de adicionar notas e anexos e vai ainda ser possível alterar o estado dos pré-contratos para "Pronto para BC".

Estes são os principais recursos da solução, apesar da simplicidade da visão de alto nível, estes requerem um processo mais complexo na implementação de nível inferior, como será explicado mais adiante.

3.2.2 Modelo Domínio

Nesta secção é apresentado o modelo de domínio do projeto que está a ser desenvolvido:

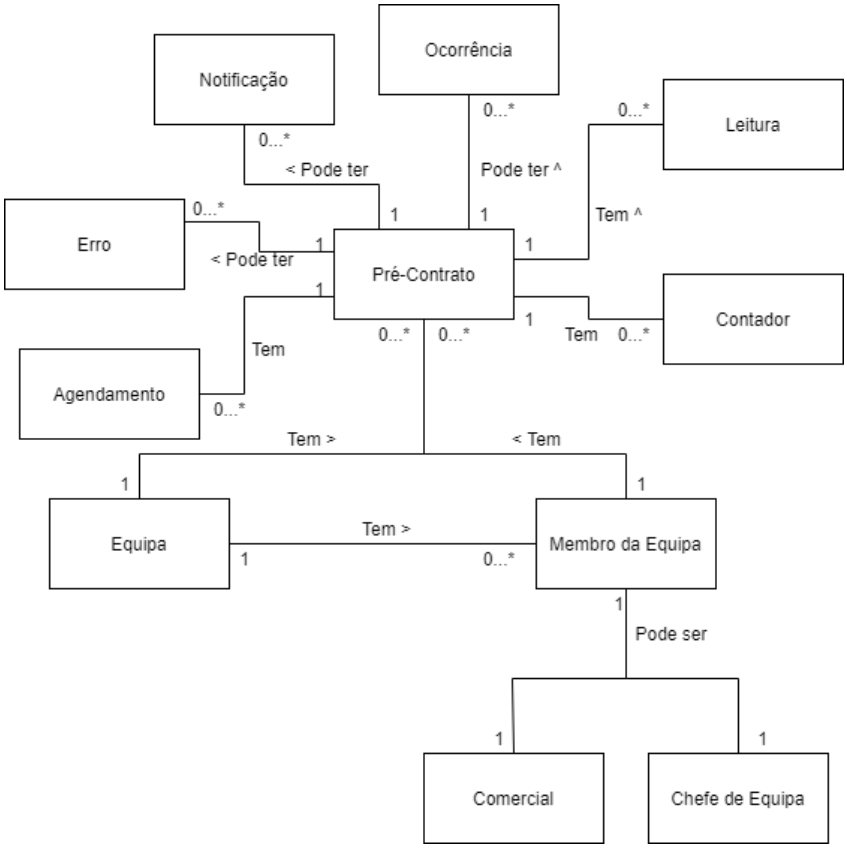

Figura 3.6: Modelo de Domínio

As classes conceptuais representadas na figura 3.6 foram identificadas apartir dos requisitos solicitados pelo cliente, estas são:

- **Equipa**: Equipas comerciais que podem ter chefes de equipa e comerciais. Estas equipas são independentes umas das outras, e por esse motivo uma equipa não pode ter acesso aos registos de outra equipa;
- **Membro de Equipa**: Utilizadores do sistema que são incorporados em equipas comerciais. Em caso algum, um utilizador pode ser membro de duas equipas ao mesmo tempo. Caso o membro seja chefe de equipa, este consegue ver todos os contratos criados pelos membros da sua equipa, no caso de ser apenas um comercial, só terá acesso aos seus próprios contratos;
- **Notificação**: Notificação proveniente do Portal de Switching, relacionada a um contrato, sobre eventos de negócio sinalizados pelo OLMC durante o processo de contratação;
- **Ocorrência**: Provenientes do Portal de Switching, interrompem o processo de switch do contrato que lhes está associado. Para dar novamente seguimento ao processo de contratação é necessário enviar novamente o contrato para Switch;
- **Erro**: Vindos do Portal de Switching, indicam erros que ocorreram no contrato e que impedem que este continue com o ciclo normal de acontecimentos;
- **Pré-Contrato**: Um pré-contrato é a entidade principal do projeto, é sobre este que a maior parte das operações são feitas. Este tem diferentes estados, como está representado na figura 3.7:

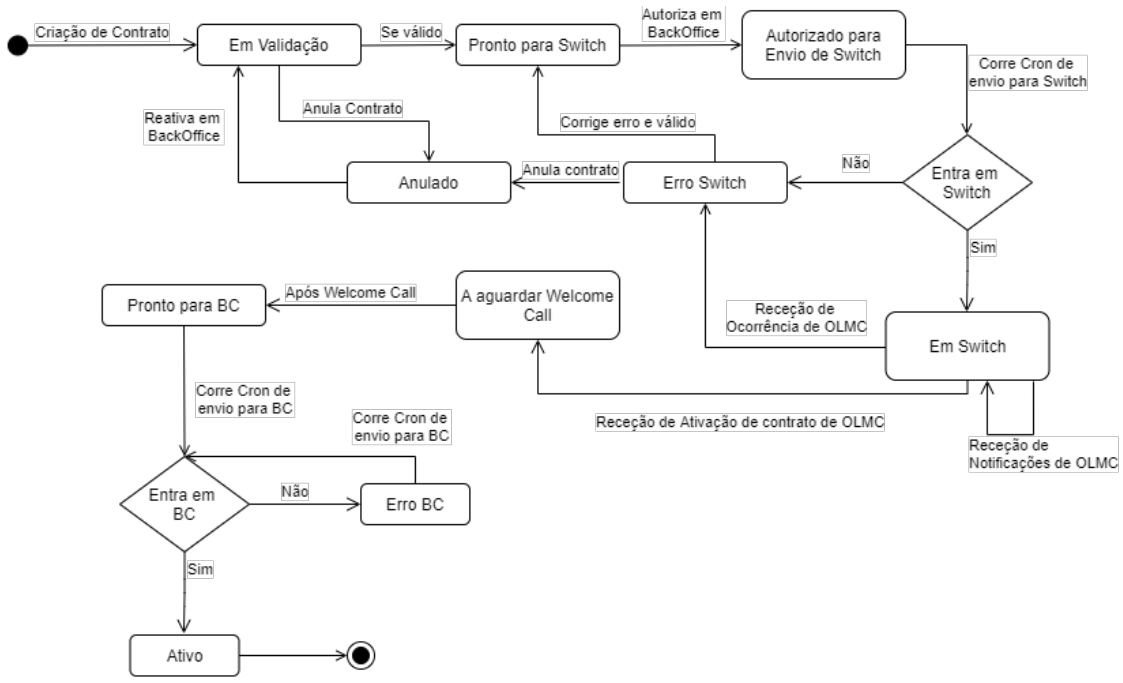

Figura 3.7: Diagrama de estado de um Pré Contrato

- **Agendamentos**: Proveniente do Portal de Switching, indica os agendamentos necessários para um contrato que está em curso no OLMC. Assim como os contratos, também estes têm estados:

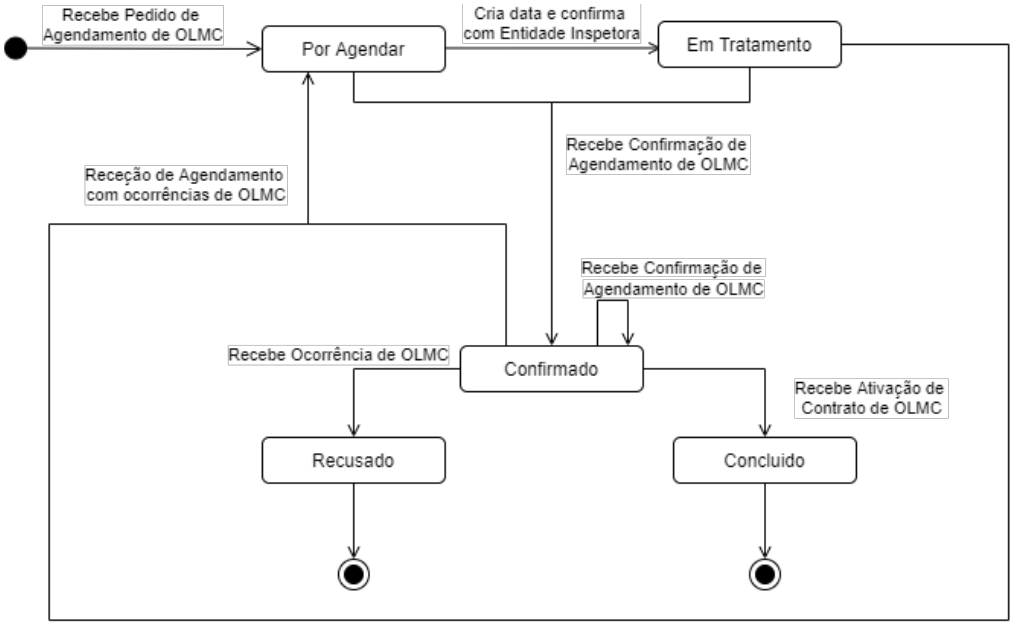

Figura 3.8: Diagrama de estado de um Agendamento

- **Contadores**: Contadores associados a um contrato ativo em OLMC. Estes são provenientes do Portal de Switching quando este envia uma ativação de contrato para o módulo de contratação;
- **Leituras**: Leituras de um contador que estão associadas a um contrato ativo em OLMC. São recebidas do Portal de Switching quando este envia uma ativação de contrato.

O modelo domínio apresentado representa apenas as entidades que estão implementadas no momento de escrita deste documento.

3.2.3 Modelo de Dados

Nesta seção é descrita a base de dados que foi usada na solução. Só será apresentado o modelo relevante para a solução, pois existem conceitos relacionados que estão fora do propósito deste projeto, como conceitos relacionados com a *framework* da empresa.

Para o projeto em análise foi usada uma base de dados relacional, com recurso à ferramenta SSMS.

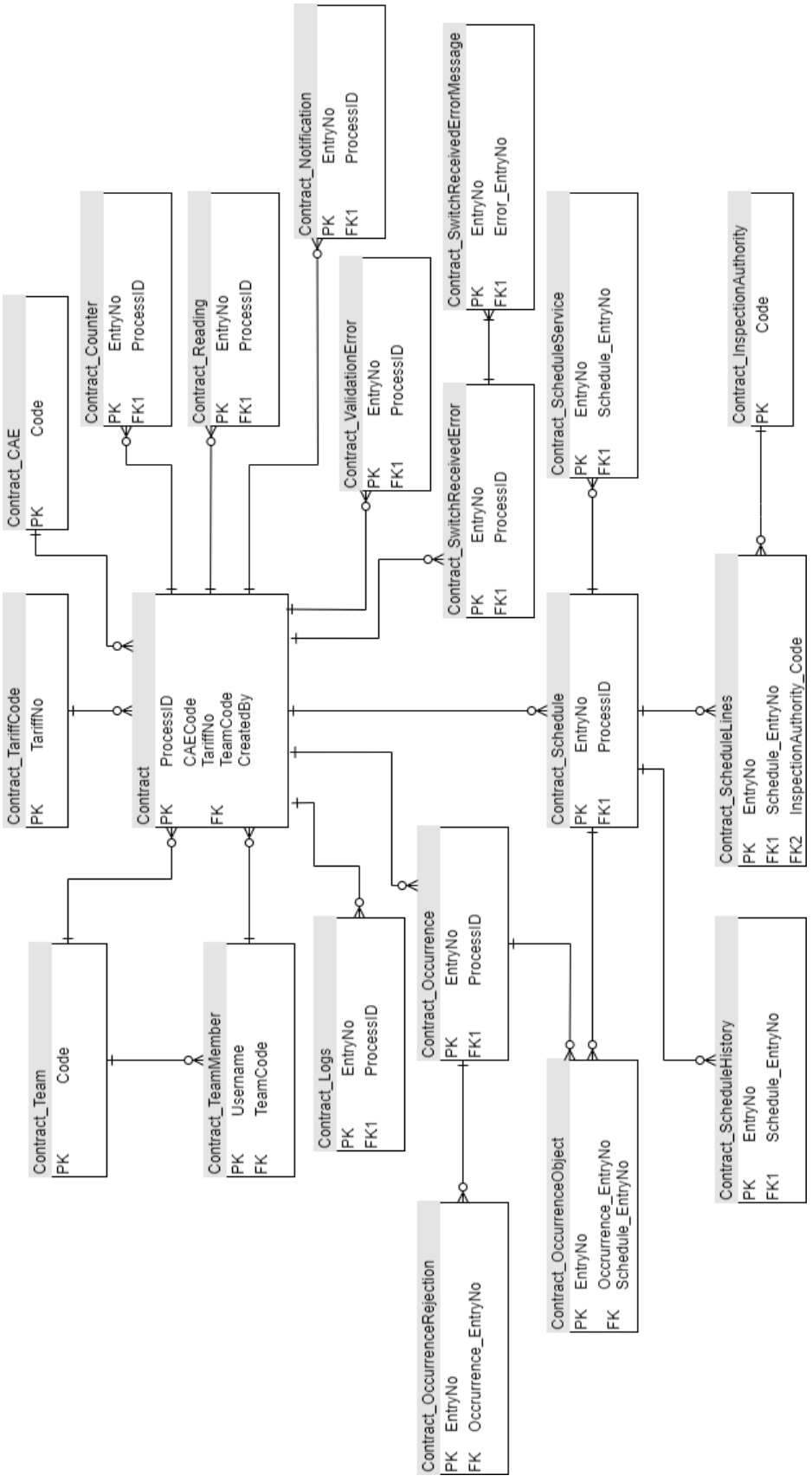

Figura 3.9: Modelo de Dados

Analisando as tabelas da figura 3.9, segue uma breve descrição das tabelas principais do projeto:

- **Contract**: Esta tabela contém toda a informação diretamente relacionada a um pré-contrato. Esta está relacionada a quase todas as tabelas existentes, uma vez que se trata da entidade principal do negócio. Um contrato não pode ser criado sem ter a ele associado uma equipa e um membro de uma equipa;
- **Contract_TariffCode/Contract_CAE**: Ambas as tabelas são preenchidas com dados que vêm diretamente de BC e são essenciais para o negócio. A *Contract_TariffCode* guarda os tarifários existentes na comercializadora entre os quais os clientes vão poder escolher. A *Contract_CAE* persiste os Classificação Portuguesa das Atividades Económicas (CAE), que representam a atividade económica da empresa. No contexto do negócio, e por se tratarem de clientes domésticos ou com consumos inferiores a 10 000 m3 o CAE definido por default na criação de um contrato é o 99921, que representa o consumo doméstico;
- **Contract_Team**: Persiste as equipas de comerciais. Uma equipa pode ser eliminada, caso não tenha a ela associado nenhum contrato, caso contrário, não pode ser apagada. Se esta só tiver membros mas nenhum deles tiver criado nenhum contrato, então pode ser apagada, eliminando também as linhas existentes da tabela *Contract_TeamMember* que a ela estejam relacionadas;
- **Contract_Counter**: Persiste a informação dos contadores associados a um contrato. Esta tabela só é preenchida aquando a ativação de um contrato, que é feito através do *OLMC*;
- **Contract_Reading**: Persiste a informação da primeira leitura de um contador, ou as leituras já existentes naquele contador. Assim como a tabela *Contract_Counter* só é preenchida quando um contrato é ativo em OLMC;
- **Contract_Notification**: Proveniente de OLMC persiste as notificações sobre eventos de negócio durante o processo de contratação;
- **Contract_Schedule**: Proveniente de OLMC guarda o agendamento que ocorre durante o processo de contratação caso o ORD tenha necessidade de efetuar serviços no local de consumo. A esta estão associadas as tabelas:
  - **Contract_ScheduleService**: Serviços a ser efetuados no local de consumo;
  - **Contract_ScheduleLines**: Datas com a entidade inspetora;
  - **Contract_ScheduleHistory**: Histórico de datas de confirmação de agendamentos;
  - **Contract_OccurrenceObject**: Ocorrências associadas ao contrato que influenciam o agendamento.
- **Contract_Occurrence**: Proveniente de OLMC guarda as ocorrências que interromperam o processo de switch. A esta estão associadas as tabelas:
  - **Contract_OccurrenceRejection**: Razão pela qual o contrato foi rejeitado em OLMC;
  - **Contract_OccurrenceObject**: Ocorrências associadas ao contrato que impedem o seguimento do contrato.

- **Contract_SwitchReceivedError**: Erro proveniente de switch que pode ocorrer durante o processamento de um contrato;
- **Contract_ValidationError**: Persiste os erros de validação de um contrato, impedindo que este passe para o estado "Pronto para Switch". Estes erros são guardados para ser possível mostrá-los ao utilizador, e sempre que um contrato é validado são apagadas as linhas desta tabela a ele associado;

Como alternativa a este tipo de base de dados poderíamos ter uma base de dados não relacional, mas dado o contexto e a complexidade das interligações necessárias entre tabelas, esta não se mostrou como sendo uma mais valia neste projeto.

3.2.4 Vista Lógica

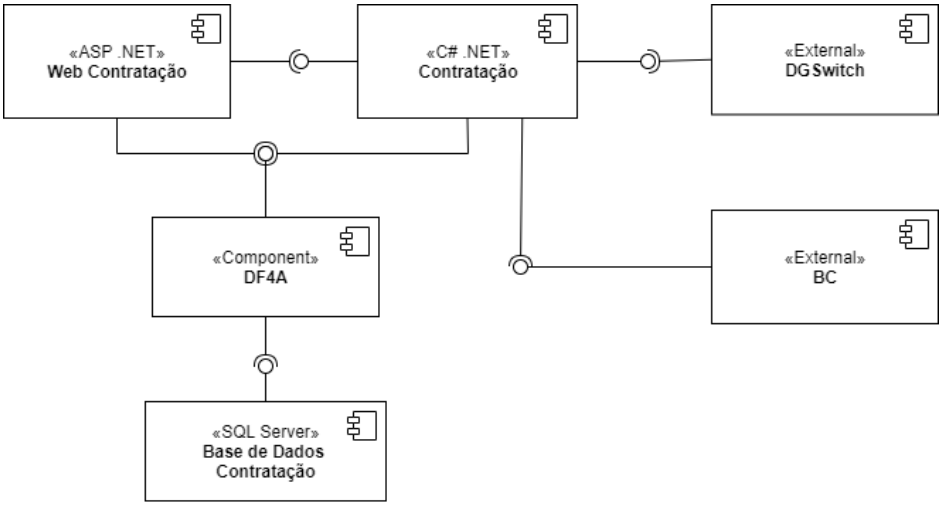

Figura 3.10: Diagrama de Componentes do Sistema

Através da análise da figura 3.10 percebemos que existem 6 componentes principais do projeto.

- **DF4A**: Responsável pela disponibilização dos serviços necessários à construção da aplicação. Esta disponibilização é feita através de pedidos SOAP. Para além disso, esta é responsável por toda a comunicação e persistência na base de dados, bem como pelo controlo de acessos das aplicações da empresa;
- **Contratação**: Local onde é definida a lógica do projeto. Neste são também definidas as tabelas, as páginas e os menus através de ficheiros XML;
- **Web Contratação**: Neste componente são construídas páginas personalizadas que a framework desenvolvida não permitia criar apartir de ficheiros XML. Por exemplo, é nesta onde são feitas as *dashboards* do projeto. Consome recursos do componente Contratação, pois nestas páginas podem ser reutilizadas páginas simples definidas nos ficheiros XML;
- **DGSwitch**: Componente externo desenvolvido por uma empresa externa que tem como objetivo enviar para o componente Contratação todos os fluxos provenientes de OLMC;

- **BC**: Componente externo responsável por gerir a faturação dos contratos criados no portal;
- **Base de Dados Contratação**: Base de dados relacional (SQL) onde é guardada toda a informação da aplicação.

Em relação ao componente Contratação, é importante saber como ele é desenhado.

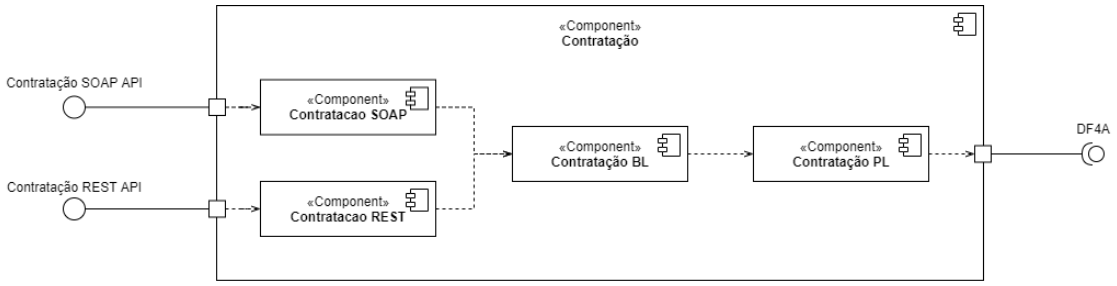

Figura 3.11: Diagrama de Componentes Contratação

A figura 3.11 pretende demonstrar como o componente Contratação foi desenhado. A partir da imagem é possível perceber que foi usado o padrão por camadas de modo a organizar a arquitetura do sistema.

Cada camada possui determinadas responsabilidades e na maioria dos casos depende apenas das camadas inferiores que fornecem serviços à mesma, podendo existir exceções que não impedem a correta implementação do padrão.

Os padrões de arquitetura em camadas são padrões onde os componentes são organizados em camadas horizontais. Este é o método tradicional para a criação da maioria dos softwares e deve ser autônomo, ou seja, onde os componentes estão interconectados, mas não dependem uns dos outros (Wickramarachchi 2017).

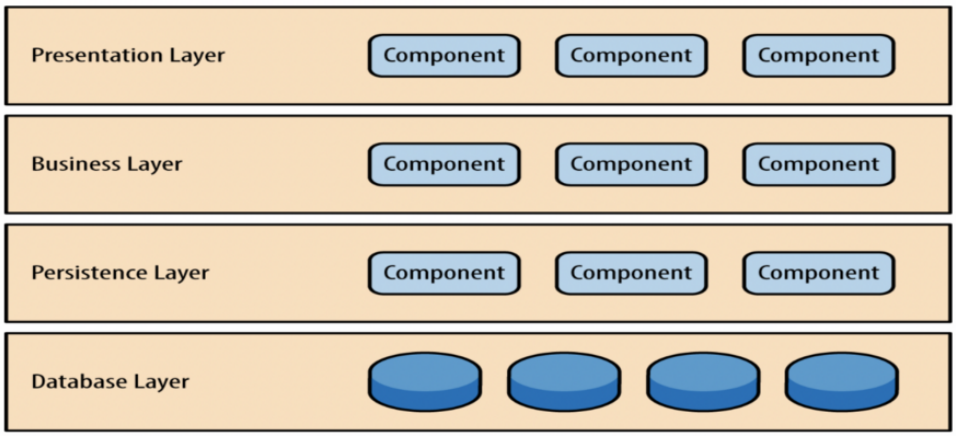

Figura 3.12: Arquitetura por Camadas [Fonte: (Wickramarachchi 2017)]

Na figura 3.12 é possível identificar 4 camadas principais no uso deste padrão.

A *presentation layer* contém o design gráfico da aplicação, bem como qualquer código para manipular a interação do usuário. Na *business layer* é onde estão os modelos e a lógica que são específicos para o problema de negócios. A camada de persistência contém o código para aceder à camada da base de dados, é o conjunto de código que manipula a base de

dados: detalhes de conexão, etc. A camada da base de dados é a tecnologia de base de dados subjacente (que no caso deste projeto é o SQL Server) (Walpita 2019).

Relativamente ao diagrama representado na figura 3.11 é possível dizer que existem 2 componentes na camada persistência, que é o componente Contratação PL e a DF4A.

O primeiro tem a responsabilidade de gerir tudo o que é específico ao negócio da Contratação, e a DF4A tem a responsabilidade geral, de gerir qualquer alteração ou obtenção de dados a qualquer base de dados de qualquer projeto da empresa.

3.2.5 Vista de Implementação

Do ponto de vista do programador, a figura 3.13 descreve a organização e a relação entre os diferentes *packages* de código-fonte.

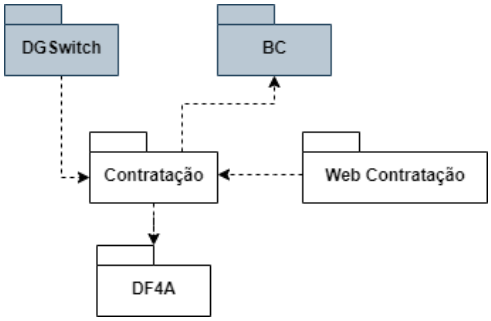

Figura 3.13: Diagrama de Implementação do Sistema

Semelhante ao que se pôde observar na vista lógica, os componentes foram organizados por *package*.

Também são mencionados os dois componentes externos que são fundamentais e insubstituíveis - DGSwitch e BC.

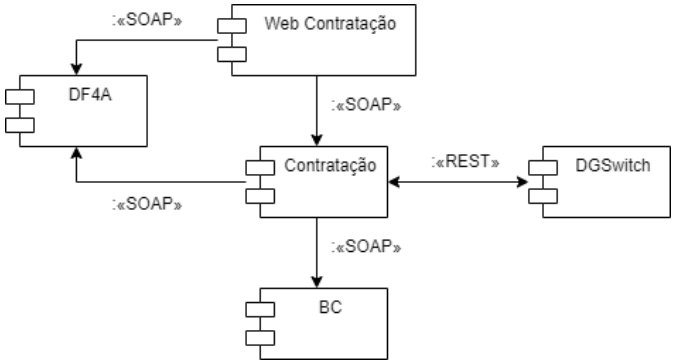

Figura 3.14: Diagrama de Comunicação do Sistema

A figura 3.14 apresenta os protocolos implementados para a comunicação entre os componentes. Para a comunicação entre a Contratação e o DGSwitch, foi usado o protocolo REST, muito comum e o mais conhecido atualmente.

Quanto às comunicações restantes, foi usado o protocolo SOAP que é o que a empresa utiliza na maioria dos seus projetos e com o qual está mais familiarizada.

Para uma melhor compreensão do projeto, é também importante representar o diagrama de implementação do componente Contratação.

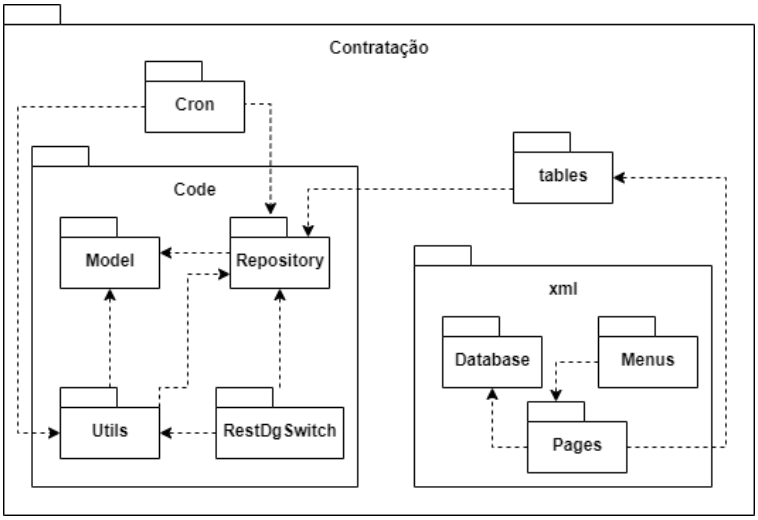

Figura 3.15: Diagrama de Implementação do Sistema

Como é possível ver na figura 3.15 existe alguma complexidade dentro deste componente. É necessário referir que os componentes contidos na pasta xml, tables e Cron são obrigatórios para o uso da *framework* da empresa (DF4A).

Como já foi referido anteriormente, no capítulo 2 na secção 2.2, a *framework* da empresa é responsável pela persistência na base de dados e pelo render das páginas HTML, ou seja, o processo pelo qual se obtém o produto final da página.

Desta forma, a definição tanto da base de dados como das interfaces gráficas da *framework* é feita através de ficheiros XML, respeitando cada um Schemas diferentes.

Além disso, torna-se importante entender como é a estrutura do código e principalmente como ele é codificado, sendo isto explicado mais à frente neste documento.

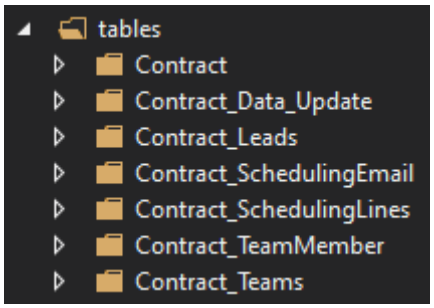

Figura 3.16: Conteúdo do Componente tables

O tables contém código dinâmico que está divido em funções e os eventos de cada tabela a que é pretendido que essa ação fique associada, e na figura 3.16 é possível ver quais as tabelas que têm ações destas diretamente associadas a elas.

As funções são acedidas no XML das páginas, contidas no Pages. Nos eventos é implementado código, que irá ser executado antes ou depois de qualquer operação *Create, Read, Update, Delete* (CRUD), ou seja, toda essa lógica e verificações que sejam feitas, são implementadas em código, que segue a lógica de desenvolvimento de um trigger.

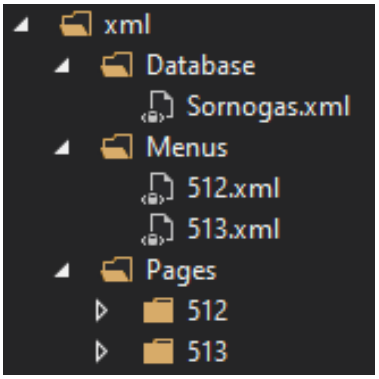

Figura 3.17: Conteúdo do Componente xml

No componente xml, visível na figura 3.17, estão representados os Menus, as Pages e a Database. Como já referido anteriormente, todos este ficheiros são ficheiros XML que a DF4A vai conseguir interpretar se estes seguirem o Schema definido para cada um deles, e para os quais mais à frente serão dados exemplos e todas as explicações necessárias.

Na *framework* um Cron é um processo automático e toda a gestão da programação deste é responsabilidade da DF4A, isto é, o controlo de quando este processo é suposto correr, como por exemplo, as horas do dia a que corre, o intervalo de horas entre cada execução, e a quantidade de vezes que este é feito num dia.

No componente Cron é então implementado o que este irá realmente fazer em concreto. No caso do projeto em análise, existem dois Crons diferentes, o que envia os pré-contratos para OLMC e o que por sua vez envia para BC.

Ambos têm como finalidade libertar o utilizador de processos que não necessitam da sua intervenção direta, aligeirando a sua carga de trabalho.

Para concluir, e em relação ao componente Code, representado na figura 3.15, que é único que não é gerido pela *framework* e por isso segue uma arquitetura mais comum.

A implementação deste foi inspirada no *Repository Pattern*, possuindo dois dos componentes principais deste padrão, o Model e o Repository.

Como já referido anteriormente, a responsabilidade de Controller é da DF4A, e por isso esse componente do padrão *Repository* não está presente neste módulo. Por essa mesma razão é referido que a implementação foi apenas inspirada neste.

Este padrão tem como principais características ser de fácil implementação e gestão. O seu principal objetivo é o de possibilitar uma forma abstrata e genérica da aplicação trabalhar com a camada de dados, sem se preocupar se a implementação é para uma base de dados local, ou para uma API on-line. Este adiciona uma camada de abstração ao acesso aos dados, fazendo com que o código de acesso à base de dados seja gerido num só lugar (Peretti 2020).

Os repositórios são classes que encapsulam a lógica necessária para aceder aos dados, fornecendo uma capacidade de manutenção e desacoplando a infraestrutura ou tecnologia usada para aceder à base de dados da camada de domínio (Microsoft 2018).

As respetivas responsabilidades dos componentes do Code são:

- **Model**: Local onde estão presentes todas as classes de domínio;

- **Repository**: Responsável por todo o conhecimento de persistência, incluindo o mapeamento de tabelas para objetos de domínio;
- **RestDGSwitch**: Onde se encontram as routes e respetivos processos a ser implementados quando é chamado um serviço do portal pelo DGSwitch;

3.2.6 Vista de Implantação

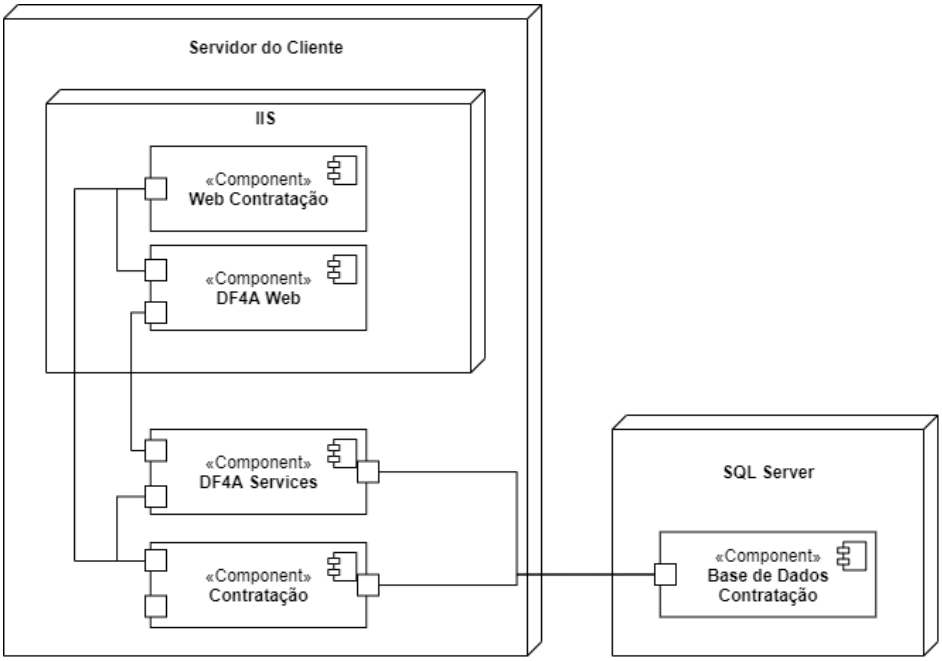

Figura 3.18: Diagrama de Implantação

Na figura 3.18 podemos identificar 2 nós diferentes:

- **Servidor do Cliente**: nó no qual se encontram hospedadas as aplicações;
  - **IIS**: aqui encontram-se hospedadas ambas as aplicações Web, desenvolvidas em ASP .NET.
- **SQL Server**: Nó no qual se encontra a base de dados da contratação.

3.2.7 Vista de Processos

Nesta subsecção, vão ser explicados os principais processos e demonstrar como os componentes do projeto interagem entre si.

Criação de um Contrato

Para a criação de um contrato é usada a *framework* interna da empresa, e por essa razão torna-se relevante exemplificar como os componentes interagem neste caso.

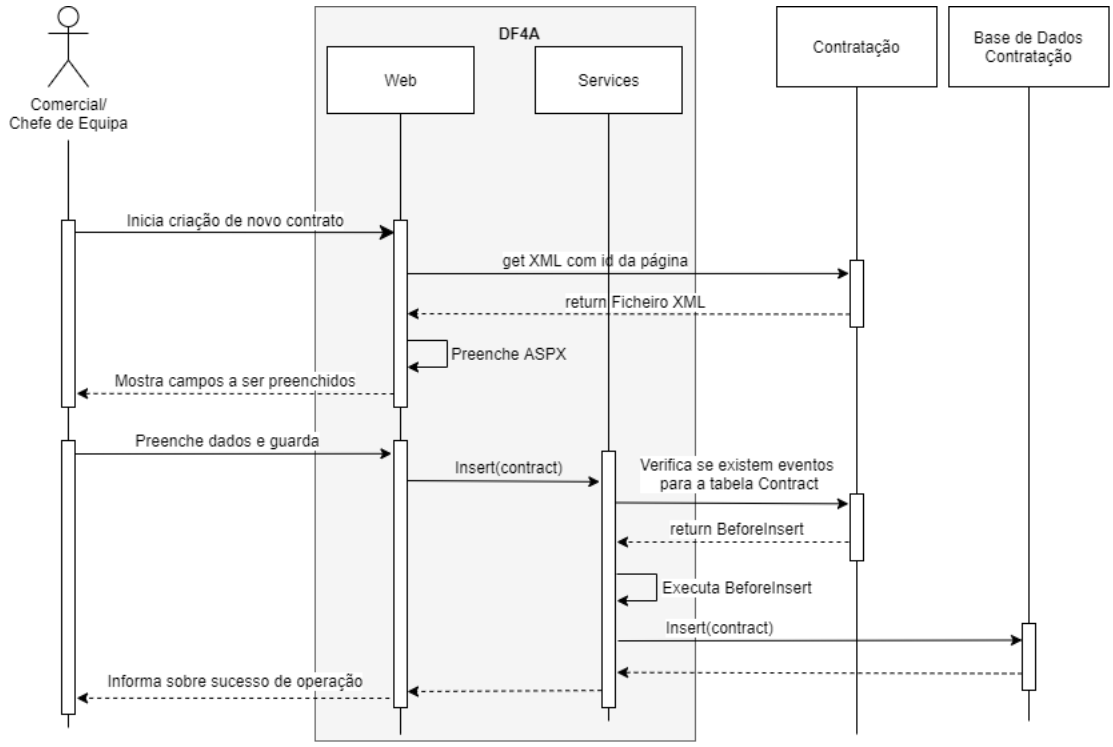

Figura 3.19: Diagrama de Processo da criação de um pré contrato

Como é visível na figura 3.19 é a DF4A que trata da maior parte da lógica importante neste caso de uso. Esta começa por obter o ficheiro XML necessário do componente Contratação para conseguir dar *render* da página e preencher o ASPX com os elementos necessários.

Para além disso, é percetível como e quando os eventos definidos nas tabelas irão funcionar e de que forma são usados, e também perceber que a iteração com a base de dados passa sempre pelos serviços da *framework*.

Envio para Switch de pré contratos

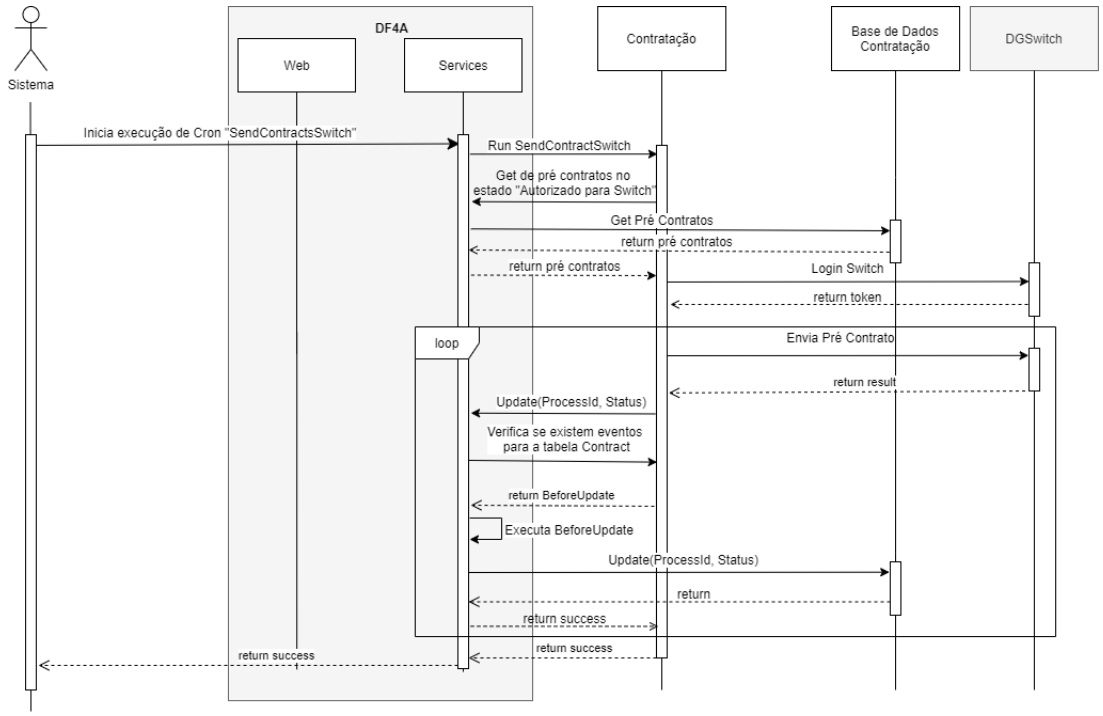

Figura 3.20: Diagrama de Processo do Envio de pré contratos para switch

O envio para o DGSwitch de pré contratos é iniciado pelo próprio sistema, por se tratar de um Cron.

Na figura 3.20 vê-se que em primeiro lugar é necessário ir buscar os pré contratos que se encontrem no estado "Autorizado para Switch", e mais uma vez, é possível perceber que é a componente Services da DF4A que comunica com a base de dados.

Após isto, é feito o login na plataforma DGSwitch e um a um são enviados os pré contratos, alterando o seu estado para "Em Switch".

Envio para BC de pré contratos

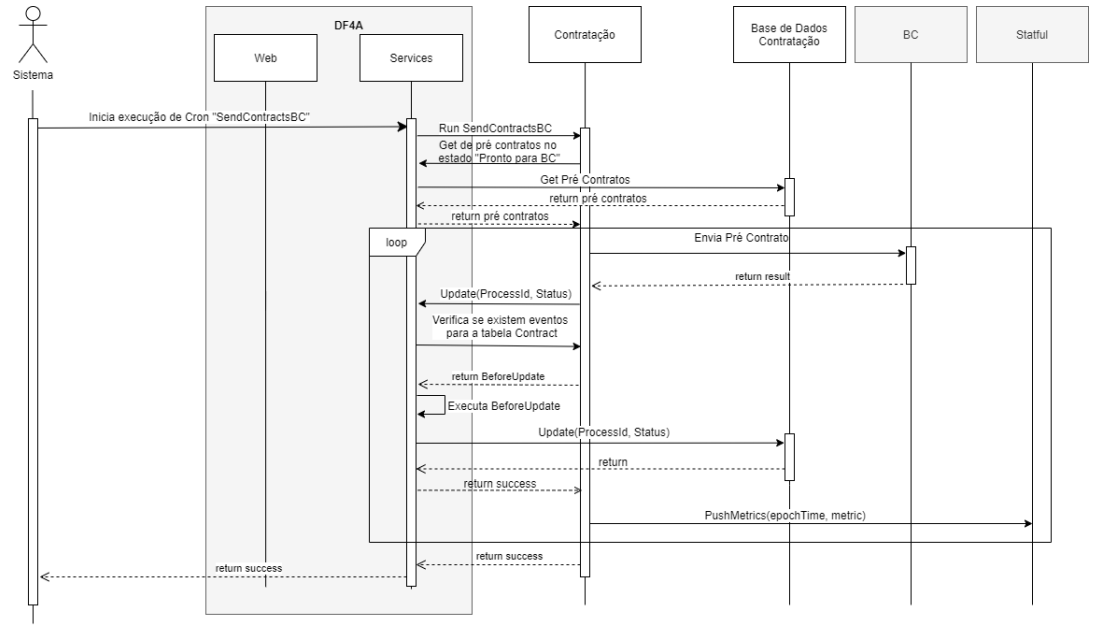

Figura 3.21: Diagrama de Processo do Envio de pré contratos para BC

A figura 3.21 representa o diagrama de processo de um envio para BC no caso de sucesso. De forma semelhante ao envio de pré contratos para switch, também o envio para BC é executado pelo sistema por se tratar de um Cron.

Para começar é necessário aceder à base de dados para ir buscar os pré contratos no estado "Pronto para BC"e depois um a um serem enviados para BC. Após a confirmação de entrada em BC o estado do pré contrato enviado é alterado para "Ativo".

Dentro do ciclo *for* representado, são ainda enviadas as métricas para validação do requisito não funcional de tempo médio de resposta dos *endpoints* do serviço SOAP inferior a 1 segundo.

Receção de Ativação de Contrato de Switch

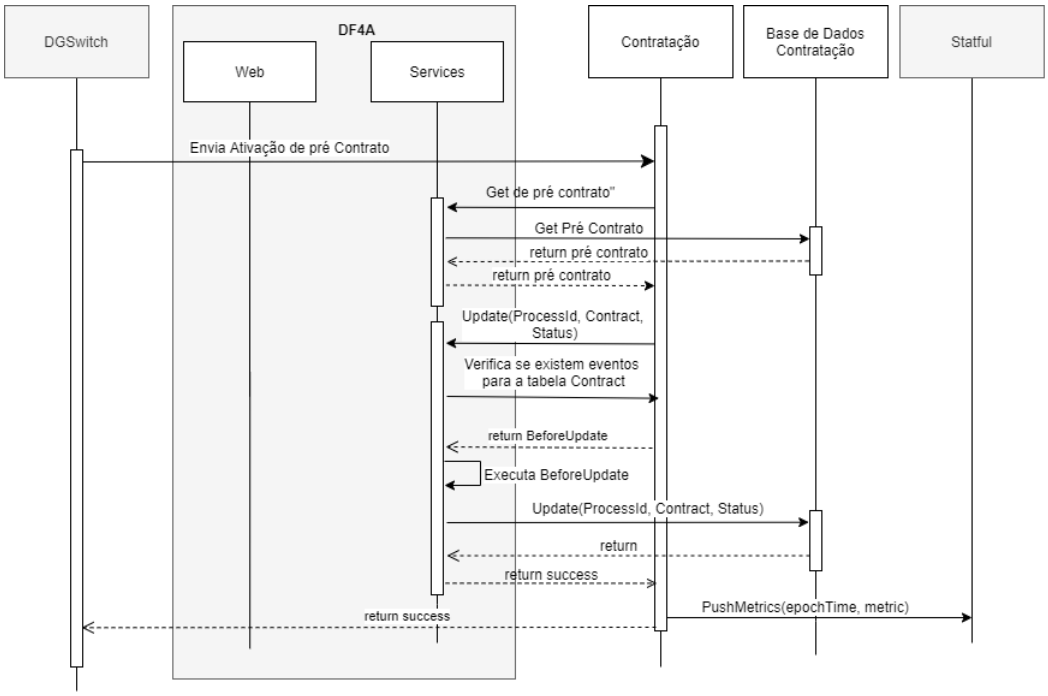

Figura 3.22: Diagrama de Componentes Receção de Ativação de Contrato de Switch

Como é possível ver na figura 3.22 quem despoleta a ativação de um contrato é a plataforma externa DGSwitch. Este envia uma ativação de contrato para um serviço REST da Contratação.

Após isto é necessário ir buscar o contrato à base de dados e o seu estado é alterado para "A Aguardar Welcome Call".

Por fim, são enviadas as métrica necessárias para a validação do requisito de tempo médio de resposta dos *endpoints* do serviço REST inferior a 1 segundo.

3.2.8 Alternativas

A arquitetura referida nas secções anteriores está alinhada com a estratégia tecnológica da empresa, pelo que não existem alternativas quanto a este ponto, porém, os pontos seguintes são pontos para os quais existiam opções de implementação e para os quais vai ser exposto o estado atual, e as alternativas existentes:

- Base de Dados;
- CI/CD;
- Monitorização em tempo real/Serviço centralizado de logs;
- Qualidade de código.

Base de Dados

A base de dados desta aplicação é relacional (SQL), ou seja, define relações sob a forma de tabelas. A programação SQL facilita o uso de funções CRUD (Create, Read, Update, Delete), no entanto, vai para além disso, com o auxílio na otimização e manutenção da base de dados (Krishna 2020).

Para além deste modelo, existem ainda as bases de dados não relacionais, ou NoSQL. Este evita ligações, é horizontalmente escalável e não requer um esquema fixo.

Numa aplicação em que existe uma grande correlação entre tabelas e onde o número de transações necessárias é elevada, as bases de dados relacionais são as mais indicadas. Estas seguem as propriedades ACID, que garantem Atomicidade, Consistência, Isolação e Durabilidade (IAN 2016).

Por outro lado, as bases de dados não relacionais seguem as normas BASE, Basicamente disponível, Estado suave, Eventualmente consistente (Krishna 2020). Estes dois modelos são então apresentados na figura 3.23.

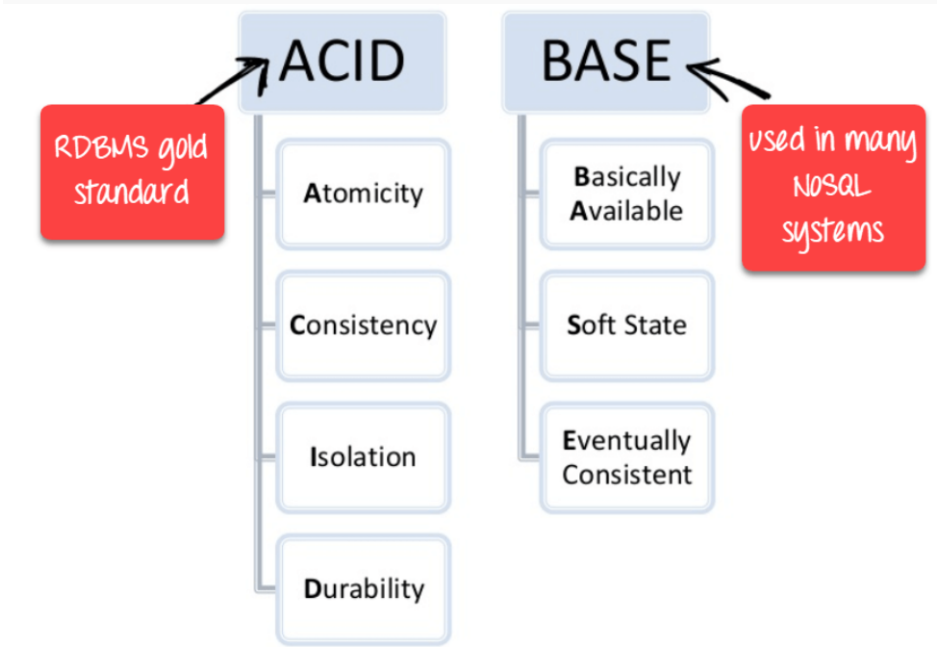

Figura 3.23: Propriedades ACID vs BASE [Fonte (Krishna 2020)]

Depois de avaliadas as alternativas, optou-se por se recorrer a uma base de dados relacional. Apesar de a *framework* da empresa tirar partido de bases de dados relacionais, no contexto deste projeto torna-se claro que uma base de dados relacional é a escolha correta. Não só por se adaptar melhor às necessidades do projeto, devido ao facto de existirem muitas relações entre tabelas, como também por já se utilizarem os serviços semelhantes noutros produtos da empresa.

CI/CD

Um bom fluxo de trabalho de desenvolvimento é extremamente importante para todas as equipas de modo a organizar o seu processo de desenvolvimento. Duas práticas recomendadas que ganharam muita força nos últimos anos são Continuous Integration (CI) e Continuous Delivery (CD) (Motlik 2017).

Trabalhar com equipas pode ser bastante desafiante se não houver um processo para rever e verificar o código antes deste ser integrado, ou mais importante para impedir que erros cheguem a produção.

Como referido na secção 1.5.3, atualmente existem 3 ambientes, em que o processo de alteração de código entre eles é feito à mão, sendo feito um *build* do que está feito no ambiente de desenvolvimento e é colocado em qualidade por um membro da equipa. Se o *build* que está em qualidade passar todos os testes de usabilidade feitos pelo cliente, então este *build* é colocado em produção.

Uma das melhores alternativas ao que está a ser implementado de momento no processo de desenvolvimento, devia ser a existência de três *branchs* - Master, Qual e Dev - correspondentes ao ambiente de produção, qualidade e desenvolvimento, respectivamente. As ramificações Dev e Qual seriam uma réplica da ramificação Master, onde todas as ramificações de tarefas são integradas após as *pull requests* terem sido cuidadosamente verificadas, corrigidas e todos os testes serem executados.

Como referido por (Bo 2019) no seu artigo, é possível automatizar o processo de compilação e de *deployment* para aplicações ASP.NET em execução no IIS usando pipelines do Azure.

Conectando o ambiente de desenvolvimento ao Azure, seria possível ter uma pipeline de testes, onde todas as *builds* fossem verificadas antes do *deploy* automático no ambiente de qualidade (se a *build* for bem sucedida).

Além disto, o uso de Git seria uma das grandes alternativas à forma de como o projeto está a ser desenvolvido. Usando o *branch gitflow* daria oportunidade à criação de *branchs* para cada tarefa desenvolvida. Cada ramificação desenvolvida deveria dar *merge* apenas à ramificação de dev e, posteriormente, ao *branch* de qualidade, e só depois ao *branch* de produção após todos os testes de usabilidade serem feitos.

Resumindo o processo:

- Selecionar uma tarefa do *backlog*;
- Criar um *branch* de tarefa;
- Implementar testes e o requesito pedido;
- Criar um *pull request*;
- Um membro da equipa aprova a solicitação e dá *merge*;
- Azure cria uma *build* e executa o pipeline de teste, se bem-sucedido, dá *merge* ao ramo de qualidade;
- Após todos os testes feitos em qualidade por parte do cliente, criar um *pull request* para produção;
- Um membro da equipa aprova a solicitação e dá *merge*;

Este poderia ser o processo usado para automatizar o fluxo de trabalho do desenvolvimento à produção, além de melhorar a colaboração com outros *developers* do projeto.

Monitorização em tempo real/Serviço centralizado de logs

Como com qualquer site, também neste projeto há eventos incertos que podem criar conflitos com o código, com o servidor ou com a rede. De modo a conseguir monitorizar estes *downtimes* que podem acontecer está a ser usado o Uptime Robot. Trata-se de um serviço grátis que tem como principal função monitorizar os sites a cada 5 minutos, alertando se estes estão inativos (UptimeRobot 2020). Este segue os seguintes passos:

- Solicita os *headers* do site e obtém o código do *status*, por exemplo, "200-OK", "404-not found", a cada 5 minutos (este intervalo de tempo é configurável);
- Caso o *status* não indique um problema, então está tudo de acordo com o normal;
- Caso o *status* seja do tipo 400+ ou 500+, então algo de errado se está a passar com o serviço, pois este não está a ser carregado corretamente. De modo a garantir que não é um problema momentâneo, o UptimeRobot faz várias verificações nos 30 segundos seguintes;
- No caso do site permanecer inativo, é enviado um alerta;

Este tipo de monitorização que está a ser usado é essencial para saber se o serviço disponibilizado em produção está ativo e a funcionar de acordo com o que o cliente necessita.

Para além deste serviço, a equipa de trabalho está ainda a recolher logs de erros do sistema em produção. Nos serviços isto está a ser feito através de um ficheiro de logs que controla a verbosidade dos erros, isto é, tem níveis diferentes de logs: INFO, DEBUG e ERROR. Podemos ver um exemplo este ficheiro na figura 3.24.

```
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - ERROR: Missing column on insert TipoFicheiro Code
2020-02-17 06:02:33.9510 - INFO: SELECT convert( bigint, mt.timestamp) as timestamp,[XML Data] FROM [Sonorgas_Object Custom] as
2020-02-17 06:02:33.9354 - INFO: Code ******> GE_Clientes_MD Contratos
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - INFO: Type ******> 0
2020-02-17 06:02:33.9510 - INFO: SELECT convert( bigint, mt.timestamp) as timestamp,[timestamp], [Entry No], [Name], [Descripti
2020-02-17 06:02:33.9510 - INFO: External_Key ******> DocREN:1524
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - INFO: SELECT convert( bigint, mt.timestamp) as timestamp,[timestamp], [No], [Data] FROM [Sonorgas_eD
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - INFO: No ******> 11461
2020-02-17 06:02:33.9510 - INFO: Type ******> 0
2020-02-17 06:02:33.9510 - INFO: Code ******> GE_Dia Gas
2020-02-17 06:02:33.9510 - INFO: SELECT convert( bigint, mt.timestamp) as timestamp,[timestamp], [Entry No], [Name], [Descripti
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - INFO: External_Key ******> DocREN:1529
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
2020-02-17 06:02:33.9510 - DEBUG: Opening Data Connection.
```

Figura 3.24: Ficheiro de Logs dos serviços

Para além disto, existem diferentes tabelas que guardam informações relevantes ao sistema.

- **RestLog**: Esta tabela guarda todos os pedidos feitos para o portal de atendimento vindo de entidades externas. Esta guarda o endereço que fez o pedido, o tipo do pedido (PUT, POST, GET, etc), qual o *endpoint* que foi chamado, a mensagem recebida, a mensagem enviada e a data em que o pedido foi feito, como mostra a figura 3.25;

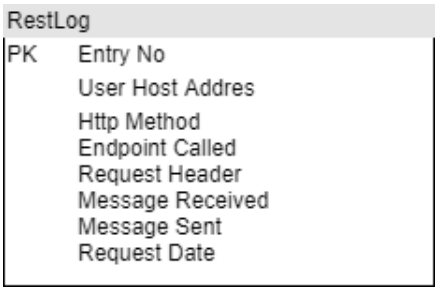

Figura 3.25: Tabela RestLog

- **CronLog**: Esta tabela regista os dados do Cron, composta pelo Entry No do Cron corrido, a data de inicio e a data de fim, e em caso de erro regista a mensagem, como mostra a figura 3.26;

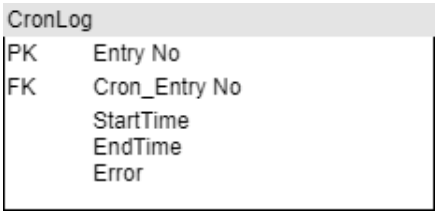

Figura 3.26: Tabela CronLog

- **SmsArchive**: Guarda o registo das mensagens enviados pelo sistema, tendo o registo do número para que foi enviado, a mensagem enviada, a data de criação e a data de envio, o estado, e no caso de haver erro, a mensagem de erro. Os estados possiveis são Registado, Enviado, Cancelada, Erro;

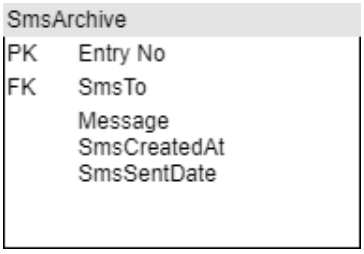

Figura 3.27: Tabela SmsArchive

- **ContractLogs**: Nesta tabela são registados todos os pedidos feitos a APIs externas relativamente a contratos. Esta guarda o Process Id do contrato, o número do contrato, o tipo (Enviado Switch, Enviado BC), Mensagem Enviada, Mensagem Recebida, Status, se deu erro ou não (Error) e a data do pedido, visível na figura 3.28;

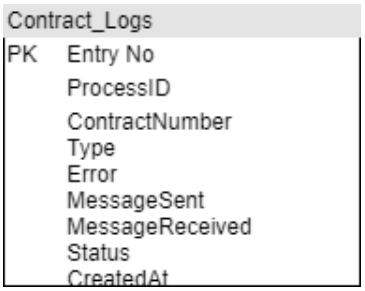

Figura 3.28: Tabela ContractLogs

Apesar do que acima já foi descrito ser por si um bom serviço de monitorização e de controlo de logs, na verdade existem frameworks que oferecem este tipo de serviço de uma forma mais eficiente e centralizada, onde é possível aceder aos logs mais facilmente e fazer uma melhor monitorização da app.

Após pesquisa, percebeu-se que o melhor sistema para providenciar o tipo de serviço necessário, era o Datadog. Este caracteriza-se como sendo um serviço de monitorização para aplicações na *cloud*, e tem como principais funções monitorizar servidores, bases de dados, ferramentas e serviços, tendo a capacidade de guardar logs, através de uma plataforma de análise de dados baseada em SaaS (Datadog 2020b).

Esta plataforma tem uma abordagem de monitorização muito abragente do IIS, reunindo análises de logs, alertas automáticos e *dashboards* prontas para usar. O Datadog integra-se às tecnologias usadas no projeto além do IIS, como o Azure e o SQL Server, oferecendo visibilidade total em relação a todo o programa. Aqui reuném-se todas as métricas importantes para qualquer serviço, que quanto ao IIS podem ser (Datadog 2020a):

- **Métricas de solicitação HTTP**: o rastreamento do volume de solicitações fornece uma ideia de como o servidor está ocupado e serve como ponto de partida para entender como a configuração do IIS está a funcionar. As métricas de solicitação HTTP também podem ajudar a identificar estrangulamentos, e determinar a exigência que o código da app coloca nos recursos do sistema;
- **Métricas de resposta HTTP**: O controlo de respostas HTTP é a forma mais direta de ver como os sites do IIS estão a satisfazer os utilizadores. É de extrema importância localizar os *upticks* na latência de resposta antes que os utilizadores o façam;
- **Métricas de disponibilidade**: Para garantir que os utilizadores podem aceder ao conteúdo, é necessário monitorizar a disponibilidade de vários componentes do IIS. É importante saber quando os serviços foram reiniciados ou pararam de funcionar completamente. O Datadog também alerta se o IIS não estiver a conseguir responder a pedidos HTTP;
- **Métricas de recursos**: Como cada *Application Pool* é isolado dos outros com os seus próprios processos de trabalho, é importante monitorizá-los individualmente. Uma *application pool* pode enfrentar contenção de recursos ou até *crashar*, enquanto que o IIS como um todo parece estar a funcionar.

Comparando estas duas soluções de implementação, percebemos que o Datadog oferece muito mais capacidade de monitorização do sistema, além de providenciar acesso fácil a informação e onde é possível visualizar tanto os logs como as métricas todas no mesmo local. Porém este serviço é pago, e pode ser uma despesa que a empresa não está disposta a ter por enquanto. Por isso, e de forma a conseguir monitorizar o sistema o máximo possível, o que foi referido anteriormente foi a estratégia adotada pela equipa.

Qualidade de código

Relativamente à qualidade de código, o projeto não está a ser avaliado através de nenhuma *framework*. Ter uma plataforma que faça este tipo de análise é importante, não só para evitar erros mas também para evitar repetição de código.

Após pesquisa foi encontrada uma alternativa que é adequada às tecnologias usadas neste projeto, sendo ela o SonarQube. O SonarQube é uma plataforma para inspeção contínua da qualidade do código, para executar revisões automáticas com análise estática do código para detectar erros e vulnerabilidades de segurança. Este oferece relatórios sobre código duplicado, padrões de codificação, testes unitários, cobertura de código, complexidade de código, comentários, erros e vulnerabilidades de segurança (Sonarqube 2020).

A extensão do SonarQube, SonarScanner para o Azure DevOps Server facilita a integração da análise da pipeline, permitindo a análise de soluções de .NET.

3.3 Implementação

3.3.1 Contexto Tecnológico da Empresa

O desenvolvimento de toda a aplicação recai muito sobre a *framework* interna da empresa. Como mencionado em pontos anteriores, 2.2.3, a Develop Framework 4 ALL (DF4A), é responsável desde a persistência na base de dados ao render das páginas HTML.

Dito isto, cada um destes componentes necessita de um Schema de XML especifico, tornando-se muito relevante demonstrar como cada um é criado, o que será feito nas secções seguintes.

3.3.2 Definição de Tabelas da Base de Dados

A *framework* permite a definição de tabelas e das suas colunas na base de dados, fornecendo também a possibilidade de definir qualquer propriedade das mesmas (ex.: chaves primárias, tipo de dados, tamanho, etc.).

```
1  <Table Name="Contract">
2     <Column Name="Estado" Type="Option" OptionCodes="0,1,2,3,4,5,6,7,8,9" Options="pt-PT=Em Validação,
3       Anulado,Pronto para Switch,Autorizado para Envio de Switch,Em Switch,Erro Switch,
4       A Aguardar Welcome Call,Erro BC,Ativo,Pronto para BC;
5       en-EN=In Validation,Aborted,Ready to Switch,Authorized for Switch Shipping,In Switch,Switch Error,
6       Waiting for Welcome Call,ERROR BC,Active,Ready for BC"
7       Label="pt-PT=Estado;en-EN=Status"/>
8     <Column Name="ProcessID" Type="int" Label="pt-PT=Process ID;en-EN=Process Id"
9       Identity="true" Size="100"/>
10    <Column Name="TeamCode" Size="50" Label="pt-PT=Codigo de Equipa;en-EN=Team Code"
11      TableRef="Contract_Team" TableRefColumns="Code,Description"
12      TableRefMapColumns="CompanyCode=CompanyCode" />
13    <Column Name="ContractNumber" Type="int" Size="50" Label="pt-PT=Nº de Contrato;en-EN=Contract Number"/>
14    <Column Name="EntryType" Size="50" Label="pt-PT=Tipo Contratação;en-EN=Type Hiring"
15      DomainTable="Contract_Domain"/>
16    <Column Name="CUI" Size="20" Label="pt-PT=CUI;en-EN=Cui;es-ES=Cui"/>
17    <Column Name="OppositionAccessRPE" Type="bit" Label="pt-PT=Oposição à integração no regime de acesso
18     massificado ao RPE;en-EN=Opposition to integration into the mass access regime to the RPE"
19     Null="True" DefaultValue="false" />
20    <Column Name="ContractStartDate" Type="Date" Label="pt-PT=Data de ínicio de contrato;
21      en-EN=ContractStartDate" Null="True" />
22    <CColumn Name="CanUseAnular"
23      Expression="CASE [Estado] WHEN '0' THEN 1 WHEN '2' THEN 1 WHEN '5' THEN 1 ELSE 0 END" />
24    <Constraint>
25      <add Type="PK" Columns="ProcessID" />
26    </Constraint>
27    <Query>
28      <Add Name="GetListOfErrors" Params="ProcessID" ParamsTypes="int">
29        <![CDATA[
30              select [ProcessID], MensagemRecebida, [CreatedAt]
31              from [#TENANT#Contract_ContractLogs]
32              where [Process ID] = @Process_ID
33                and Error = 1 and Tipo = 0
34              UNION ALL
35              SELECT [ProcessID], MessageError, srec.[CreatedAt]
36              FROM [#TENANT#Contract_SwitchReceivedError] srec
37              left outer join [#TENANT#Contract_SwitchReceivedErrorMessage] srems
38                on srems.[Error_Entry No] = srec.[Entry No]
39              where srec.[ProcessID] = @Process_ID
40                  ]]>
41      </Add>
42    </Query>
43  </Table>
```

Figura 3.29: Exemplo da definição da tabela Contract no ficheiro XML

A figura 3.29 apresenta algumas das colunas da tabela Contract de modo a ilustrar como uma tabela é criada através da *framework*. Analisando a figura é então possível perceber que na linha 1 é onde fica explícito o nome da tabela que queremos criar na base de dados.

Para atribuir a chave primária é usada a tag "Constraint"e é adicionado um elemento do tipo PK (linha 25).

Cada coluna é criada pela tag "Column"que por defeito assume o tipo de dados nvarchar, e por isso não é necessário definir o tipo usando o atributo “Type”, a não ser que seja pretendido outro tipo de dados.

Nesse caso, podemos ter um "Type"int (linha 8 e 13), que representa um inteiro, decimal, bit (linha 17) que é um booleano, Date (linha 20), datetime para guardar dados do tipo data e hora, nvarchar para uma string e Option (linha 2). Este último tipo visa criar colunas que podem adotar diversos valores, números inteiros, e no caso em análise estes valores

representam o estado do pré-contrato. Quando este tipo é usado é necessário adicionar o atributo "OptionCodes"com os códigos que vão poder ser atribuídos.

Nesta tag é ainda possível ter atributos como “Name” que corresponde ao nome da coluna, “Size” que é o tamanho que o registo pode ter, “Null” que informa se o registo pode ser null na base de dados ou não (linha 21), e a “Label”, isto é, o título dado à coluna. Este último não é guardado na base de dados, é apenas acessível através dos ficheiros de configuração das interfaces gráficas.

O atributo "Identity"é utilizado nas colunas de identidade, isto é, estas colunas são usadas para gerar valores-chave. A propriedade de identidade numa coluna garante que cada novo valor é gerado com base no incremento atual e que cada novo valor para uma determinada transação é diferente de outras transações simultâneas na tabela. No caso da figura 3.29 temos o caso da coluna com o nome ProcessID (linha 8 e 9).

Para fazer referência a outras tabelas, existem os atributos TableRef, TableRefColumns e TableRefMapColumns, e através dos quais é possível criar *Foreign Keys* (linhas 10, 11 e 12).

Quando existe a necessidade de, à semelhança das colunas com o "Type"Option, ter uma coluna que adota diferentes valores mas que se pretende que esses valores sejam códigos que não são apenas um valor numerário, existe o atributo "DomainTable". Neste atributo é indicada a tabela de domínio que contém os valores que se quer que sejam atribuídos nesta tabela.

Domain Code	Code	Description	Flag Edit	Flag Archive	Created By	Created At
EntryType	AT	Alteração de Titular	1	0	SYSTEM	2020-01-19 20:55:08.8...
EntryType	ED	Entrada Direta	1	0	SYSTEM	2020-01-19 20:55:08.6...
EntryType	MC	Mudança de Comercializador	1	0	SYSTEM	2020-01-19 20:55:08.7...
EntryType	MCAT	Mudança de Comercializador e Alteração de Titular	1	0	SYSTEM	2020-01-19 20:55:08.7...

Figura 3.30: Exemplo da definição de um domínio na tabela Contract_Domain

Como é possível ver na figura 3.30 foi criado um domínio na tabela Contrat_Domain EntryType com vários valores associados a si. De modo a fazer referência a este domínio, na figura 3.29 existe a coluna EntryType (linha 14), que tem de ter exatamente o mesmo nome que na tabela de domínios, e vai poder adotar os valores AT, ED, MC e MCAT.

Para além disto, uma mais valia da criação da tabela de domínios é que se existir a necessidade de numa outra tabela guardar exatamente o mesmo domínio, é apenas necessário o uso do atributo "DomainTable"sem ser necessário voltar a criar o mesmo registo.

Para finalizar a tag "Column", é necessário referir o atributo "DefaultValue"que é apenas usado para atribuir um valor por omissão. No caso do exemplo dado é de um booleano, que na criação de uma nova linha na tabela a coluna OppositionAccessRPE vai ter sempre o valor *false* (linhas 17, 18 e 19).

É também possível definir variáveis personalizadas que não ficam guardadas na base de dados mas que são acessíveis através dos ficheiros de configuração das interfaces gráficas. Isto é, criação de colunas virtuais capazes de executar expressões SQL chamadas de CColumn (linhas 22 e 23).

Para terminar, é também possível a definição de Querys que podem mais tarde ser usadas nas páginas da *framework* mas que não são guardadas na base de dados. Estas têm de ter

o atributo "Name"que será o nome pelo qual a query será referida na página, o "Params"é onde são definidos os parâmetros que irão ser usados na query, e o "ParamsType"que representa o tipo destes.

3.3.3 Definição de Menus

A DF4A permite a criação de menus através da definição de um ficheiro XML. Por definição da empresa, cada aplicação é identificada através de um ID numérico. No caso da aplicação de Contratação vão então existir dois IDs diferentes, 512 e 513, o portal dos comerciais e chefes de equipa, e o BackOffice, respetivamente.

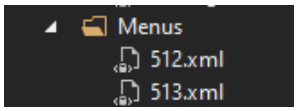

Figura 3.31: Exemplo dos IDs dos Menus

Como é possível ver na figura 3.31 existem dois ficheiros XML diferentes, um para cada uma das aplicações.

```
<?xml version="1.0" encoding="utf-8"?>
<Menus xmlns="http://tempuri.org/WebMenuSchema.xsd">
  <Menu Code="512" Title="pt-PT=Contratação;en-EN=Hiring" Application_Code="512">
    <Option ID="512003" Url="/app.contract/Dashboard.aspx" Title="pt-PT=Dashboard;en-EN=Dashboard;"
      Tooltip="pt-PT=;en-EN=" Icon="fal fa-home-lg" IDPagina="512003.0" ObjectID="512003.0">
    </Option>
    <Option ID="512001" Url="/framework/PageList.aspx?IDPagina=512001.0"
      Title="pt-PT=Pré Contratos;en-EN=Pre Contracts" Tooltip="pt-PT=Pré Contratos;en-EN=Pre Contracts"
      Icon="fal fa-file-signature" ObjectID="512001.0">
    </Option>
  </Menu>
</Menus>
```

Figura 3.32: Exemplo do XML de um Menu

Na figura 3.32 está explícita a criação de um menu através da *framework*. Aqui temos a criação de 2 opções de menu, a Dashboard e a Pré Contratos, onde é possível definir icons para cada uma delas. Além disso, cada opção tem um ID definido no atributo "ID".

Uma página de menu pode ser uma página dinâmica, como é o exemplo da Dashboard, onde no atributo "URL"se indica o caminho para a página dinâmica existente. Além disso, e o que é mais recorrente é a utilização de páginas da *framework*, que no caso da opção Pré Contratos é uma PageList.

O código descrito gera o menu presente na figura 3.33.

Figura 3.33: Menu na aplicação

3.3.4 Definição de Páginas da Framework

Nos ficheiros de definição das páginas da *framework* é feita referência à tabela sobre a qual a página incide e é definida a estrutura onde se pretende apresentar a informação. Deste modo é possível definir: as dimensões das páginas, a paginação, o tipo de navegação e a aplicação de filtros de pesquisa.

Por definição da empresa, cada página é identificada através de um ID numérico de acordo com a aplicação a desenvolver. No caso do projeto em análise existem dois IDs principais, como referido anteriormente. No caso do portal dos comerciais será o 512 e para o BackOffice o ID escolhido foi o 513. Todos os IDs das páginas terão que começar pelo seu respetivo número, na figura 3.34 encontra-se um exemplo dos IDs atribuídos às páginas.

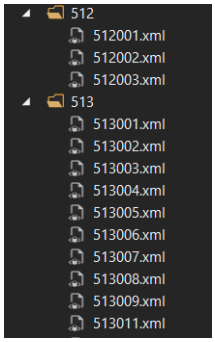

Figura 3.34: IDs das páginas criadas

PageList

Uma PageList é uma página que vai apenas mostrar uma lista da tabela que pretendemos.

```
1  <Pagina IDPagina="513004.0" Title="pt-PT=Equipas Comerciais;en-EN=Contract Teams">
2      <Table Name="Contract_Team" DataKeyNames="Code" />
3      <NavBar>
4        <add Type="Refresh" Text="pt-PT=Atualizar;en-EN=Refresh" />
5        <add Type="New" Text="pt-PT=Novo;en-EN=New"
6          Url="/framework/PageFormulario.aspx?IDPagina=513004.0001" WindowType="PopUp" WindowSize="600" />
7      </NavBar>
8      <Parametros>
9        <Formulario>
10         <Area Rows="1" Cols="2" PosLabel="T" Flow="H">
11           <Column ID="1" Name="Code" Width="200px" Required="false" />
12           <Column ID="2" Name="Description" Width="200px" Required="false" />
13         </Area>
14       </Formulario>
15     </Parametros>
16     <Grid PageSize="20" AllowColumnSort="true">
17       <Field Type="Icon" Name="Edit" IDPagina="513004.0002" Width="50px" Url="~/framework/PageFullEdit.aspx"
18         WindowType="PopUp" WindowSize="800" />
19       <Column Name="Code" Width="250px" />
20       <Column Name="Description" Width="250px" />
21       <Field Type="Icon" Name="Delete" Width="50px" ConfirmMessage="pt-PT=Confirma eliminação da equipa?;
22         en-EN=Do you want to delete this team?" />
23     </Grid>
24 </Pagina>
```

Figura 3.35: Exemplo de PageList da listagem de Equipas na aplicação

No exemplo da figura 3.35, inicialmente é definida a tabela sobre a qual esta página irá trabalhar e qual a sua chave primária, que no caso em específico é a tabela Contract_Team (linha 2).

De seguida é definida a existência de uma "NavBar"através de uma tag onde é possível definir os vários botões que irão aparecer. Depois são definidos filtros de pesquisa, através da tag "Parametros"na linha 8.

Por fim é definido o formato em que serão apresentados os dados, na linha 16 é definido que será uma “Grid” e, dentro desta, os dados que queremos mostrar.

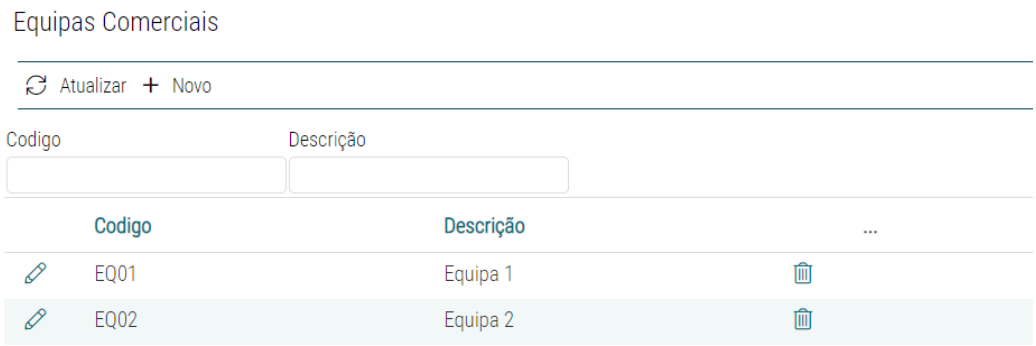

Figura 3.36: PageList de Equipas

A figura 3.36 mostra a interface gráfica gerada pelo código definido, tanto a "NavBar"como todos os botões definidos, os filtros de pesquisa e todos os registos da tabela.

Assim como na definição da base de dados, existem também componentes gerais predefinidos na definição das interfaces gráficas como, por exemplo, o botão atualizar (linha 4). No ficheiro XML basta colocar “Type=’refresh’” que irá aparecer um botão de atualizar que dá *refresh* à página.

Para além disso, e como podemos verificar nas linhas 21 e 22 da figura 3.35, existem botões com funções que são executadas pela *framework*, como é exemplo do botão de eliminar. Ao adicionar o atributo "Name=Delete"a framework já sabe que o objetivo é eliminar o respetivo registo da tabela.

De modo a exemplificar o uso de querys e CColumn que foram referidas na secção 3.3.2 é necessário recorrer a outras páginas.

```
1  <Pagina IDPagina="512001.20" Title="pt-PT=Erros Switch;en-EN=Switch Errors">
2    <Table Name="Contract" DataKeyNames="ProcessID" />
3    <Grid PageSize="20" AllowColumnSort="true" QueryName="GetListOfErrors" Parametros="ProcessID"
4      Sort="Default:[Created At] DESC">
5      <Column Name="MensagemRecebida" Label="pt-PT=Erros de Switch;en-EN=Switch errors" />
6      <Column Name="CreatedAt" Label="pt-PT=Data de Erro;en-EN=Error Date" />
7    </Grid>
8  </Pagina>
```

Figura 3.37: Exemplo de uma Query numa página da *framework*

Na figura 3.37 está presente um exemplo do uso de uma query numa página. A query que se pretende usar tem de estar definida na tabela referida na página, neste caso é a tabela Contract. Esta é chamada através do atributo "QueryName", e para além disto existe também a necessidade de se definir os parâmetros no atributo "Parametros". Na figura 3.38 vemos a interface gráfica gerada por esta página.

Erros de Switch

Erros de Switch	Data de Erro	...
NIF inválido	11-09-2020 14:54:11	
Empresa Destino -> Código: 115DOUROGN | Tipo de Entidade: E0003	11-09-2020 14:54:11	
NIF inválido	11-09-2020 10:28:49	
Empresa Destino -> Código: 115DOUROGN | Tipo de Entidade: E0003	11-09-2020 10:28:49	

Figura 3.38: Exemplo de uma Query numa página da *framework*

Relativamente às variáveis personalizadas que não ficam guardadas na base de dados, estas podem ser mostradas como mais uma coluna na interface ou podem ser usadas como validações de botões, e é este último que vai ser exemplificado.

```
1 <Grid PageSize="20" AllowColumnSort="true" Sort="Default:[Created At] DESC">
2       <Field Type="ellipsis" Name="ButtonAction" VerifyColumn="CanUseAnular" DisplayOnValue="1"
3         Label="pt-PT=Anular;en-EN=Cancel" Icon="fal fa-times-circle" Width="50px"
4         Url="/framework/PageFormulario.aspx" IDPagina="512001.21" WindowSize="500"
5         Parametros="ProcessID,{ProcessID}" />
6       <ColHidden Name="CanUseAnular"></ColHidden>
7 </Grid>
```

Figura 3.39: Exemplo do uso de uma "CColumn"numa página da *framework*

Na figura 3.39 está presente o uso da "CColumn"que foi referida na secção 3.3.2, CanUseAnular, numa "Grid", para defnir a visibilidade do botão consoante o estado do pré-contrato.

Este botão está contido numa "ellipsis"mas poderia ser um botão numa coluna da "Grid"que seguiria a mesma lógica. Para ser possível usar a "CColumn"é necessário usar a tag "ColHidden"para fazer referência a esta, e depois através da tag "Field"definir o seu uso.

Com o atributo "VerifyColumn"a verificação da expressão definida na "CColumn"será analisada, e o atributo "DisplayOnValue"irá ser o valor em que o botão irá aparecer ao utilizador, neste caso é quando a expressão retornar um 1.

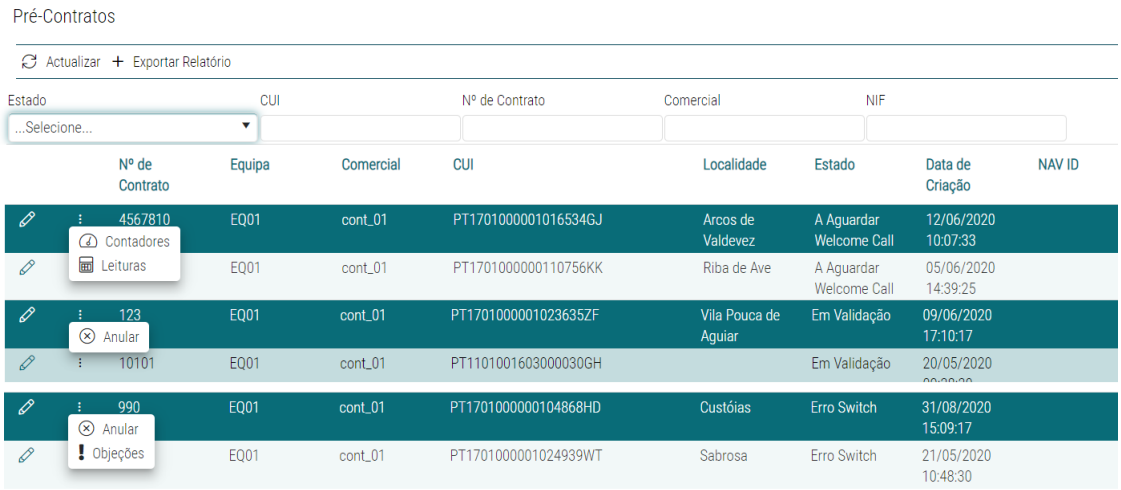

Figura 3.40: Exemplificação do botão Anular

Na figura 3.40 é possível ver a visibilidade do botão "Anular"em diferentes estados de um pré contrato e em simultâneo ver como um Field do tipo "ellipsis"é apresentado na interface gráfica.

PageFormulario

Uma PageFormulario é usada na criação de novos registos ou então para a edição destes.

```
1  <Pagina IDPagina="512001.01" Title="pt-PT=Novo;en-EN=Create;es-ES=Nuevo">
2      <Table Name="Contract" DataKeyNames="ProcessID" />
3      <Formulario>
4        <Area Cols="2" Rows="5" Flow="V">
5          <Column ID="0" Name="ContractNumber" Width="300px"/>
6          <Column ID="1" Name="EntryType" Width="300px" DefaultValue="ED"/>
7          <Column ID="2" Name="ContractHolderName" Width="300px"/>
8          <Column ID="3" Name="ContractHolderSurname" Width="300px"/>
9          <Column ID="4" Name="Birthday" Width="300px"/>
10         <Column ID="5" Name="IdentificationDocumentType" Width="300px" DefaultValue="CartaoCidadao"/>
11         <Column ID="6" Name="IdentificationNumber" Width="300px"/>
12         <Column ID="7" Name="ContractVATID" Width="300px"/>
13         <Column ID="9" Name="CUI" Width="300px"/>
14       </Area>
15       <Hidden>
16         <Add ID="10" Name="ProcessID" />
17         <Add ID="11" Name="Estado" DefaultValue="0" />
18         <Add ID="12" Name="IntegrationType" DefaultValue="0" />
19         <Add ID="14" Name="CAE" DefaultValue="99921" />
20         <Add ID="15" Name="ClientType" DefaultValue="C91" />
21       </Hidden>
22     </Formulario>
23     <ActionButton>
24       <Add Text="pt-PT=Gravar;en-EN=Save" Action="saveredirect"
25         IDPagina="512001.02" Url="~/framework/PageFullEdit.aspx"/>
26     </ActionButton>
27   </Pagina>
```

Figura 3.41: Exemplificação de uma PageFormulario

A figura 3.41 representa a criação de um pré contrato. Para a criação deste é apenas necessário que sejam indicadas quais as colunas da base de dados que se pretende que sejam preenchidas e qual o tamanho do campo.

Para além disso é possível definir valores pré-preenchidos, através do atributo "DefaultValue"para facilitar ao utilizador o preenchimento dos campos (linhas 6 e 10).

É também possível a definição de valores por omissão aos quais o utilizador não terá acesso. Estes ficam dentro da tag "Hidden", e deste modo, sempre que forem criados pré contratos através desta página em específico, o registo assumirá os valores que estão dentro desta tag.

Além disso podem ser adicionados botões com a tag "ActionButton", e no caso em análise, ao carregar neste botão o registo será guardado e o utilizador será redirecionado para uma outra página. Para isso foi necessário indicar que a "Action"era do tipo saveredirect, qual o "IdPagina"que deveria ser aberto e qual o seu "URL", que neste caso é uma página da *framework* PageFullEdit.

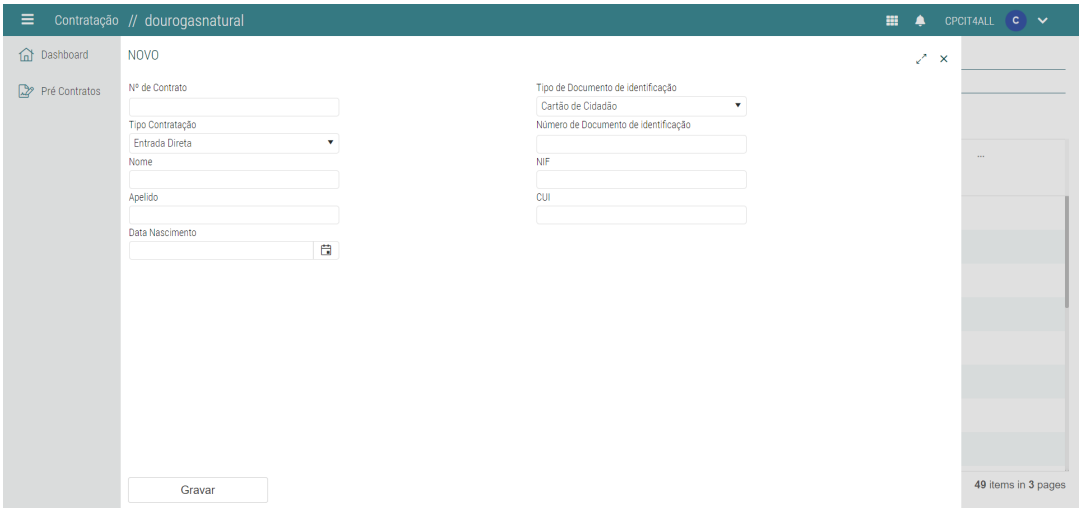

Figura 3.42: Interface de criação de novo pré contrato doméstico

A figura 3.42 mostra a interface apresentada ao utilizador que é gerada através do ficheiro XML anteriormente referido. Aqui é possível ver como a *framework* cria os campos de preenchimento sozinha.

Por exemplo, a coluna "EntryType"(Tipo Contratação na figura) é uma coluna que tem o atributo "DomainTable"associado na tabela da base de dados, e deste modo a *framework* sabe que vai ter de mostrar ao utilizador uma *ComboBox* com os domínios definidos para o utilizador ser capaz de selecionar aquele que pretender.

Outro exemplo é o campo Data de Nascimento, que está definido com o tipo date no ficheiro xml da tabela, e aqui aparece como uma data.

PageFullEdit

```
1  <Pagina IDPagina="513006.02" Title="pt-PT=Pedido de Agendamento;en-EN=Schedule">
2      <Table Name="Contract_Schedule" DataKeyNames="EntryNo" />
3      <NavBar>
4        <add Type="Comments" Text="pt-PT=Comentários;en-EN=Comments" Url="/framework/PageComments.aspx"
5          DockToUpdate="Comments" Parametros="Internal Key,Contract_Schedule:{EntryNo}" VisibleOnInsert="false"
6        </add>
7        <Add Type="Icon" Name="SendEmail" Icon="fal fa-envelope" Width="100px"
8          Url="/framework/PageFormulario.aspx?IDPagina=513006.09" Text="pt-PT=Enviar Email;en-EN=Send Email"
9          Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
10     </NavBar>
11     <Docks ViewOnly="true">
12       <add ID="AreaInformacao" Title="pt-PT=Informação;en-EN=Information" IDPagina="513006.04"
13         Type="Dynamic" AllwaysOpen="true" Parametros="EntryNo,{EntryNo}" />
14       <add ID="AreaHist" Title="pt-PT=Histórico de Datas de OLMC;en-EN=OLMC Date History" IDPagina="513006.08
15         Type="Grid" Open="false" Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
16       <add ID="AreaOccurrence" Title="pt-PT=Ocorrências no Local de Consumo;
17         en-EN=Occurrences in the Place of Consumption" IDPagina="513006.05" Type="Grid" Open="false"
18         Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
19       <add ID="AreaService" Title="pt-PT=Serviços a efetuar;en-EN=Services To Perform" IDPagina="513006.03"
20         Type="Grid" Open="false" Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
21       <add ID="AreaDatas" Title="pt-PT=Observações;en-EN=Dates" IDPagina="513006.06" Type="Grid"
22         Open="false" Parametros="Scheduling_EntryNo,{EntryNo},EntryNo,">
23         <Command>
24           <Add Type="New" Text="pt-PT=Nova Data;en-EN=New Date" IDPagina="513006.07" Method="PopUp"
25             Refresh="AreaDatas" WindowSize="500" Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
26         </Command>
27       </add>
28       <add ID="AreaEmails" Title="pt-PT=Emails Enviados;en-EN=Sent emails" IDPagina="513006.11"
29         Type="Grid" Open="false" Parametros="Scheduling_EntryNo,{EntryNo},EntryNo," />
30     </Docks>
31     <Panes Width="300px">
32       <add ID="Comments" Title="pt-PT=Comentários;en-EN=Comments"
33         Parametros="Internal Key,Contract_Schedule:{EntryNo}">
34       </add>
35     </Panes>
36   </Pagina>
```

Figura 3.43: Exemplo de uma página PageFullEdit

A figura 3.43 representa um XML de uma página de *framework* PageFullEdit.

Este tipo de páginas é composta por Docks, que podem ser do tipo "Dynamic"ou "Grid". Caso sejam do tipo "Dynamic"criam um formulário não editável, se forem do tipo "Grid"então geram uma lista consoante aquilo que é pretendido mostrar ao utilizador.

Cada elemento da Dock é uma página PageFormulario ou PageList, deste modo, o atributo "IDPagina"é sempre página desse tipo.

Importa ainda referir a PageComment presente na linha 4 da figura 3.43. Este tipo de páginas da *framework* tem como objetivo criar um espaço em que é possível adicionar notas que ficam associadas a um registo de uma tabela.

Todo o tratamento de gestão e persistência destas notas/cometários é feito pela DF4A, pelo que a única coisa que é necessária ser feita pelo programador é definir uma "Internal Key"no atributo "Parametros", como é visível na linha 5.

Esta "Internal Key"é defenida pelo programador, mas a empresa tem por convenção defenir como "NomeDaTabela:IDdoRegisto"de modo a garantir que não existem repetições de "Internal Keys".

Este botão da "NavBar"apenas tem o intuito de criar um comentário, para mostrar o comentário na página é necessário adicionar a tag "Panes"(linha 31) e indicar a mesma "Internal Key".

Este tipo de páginas geram interfaces gráficas como a representada na figura 3.44.

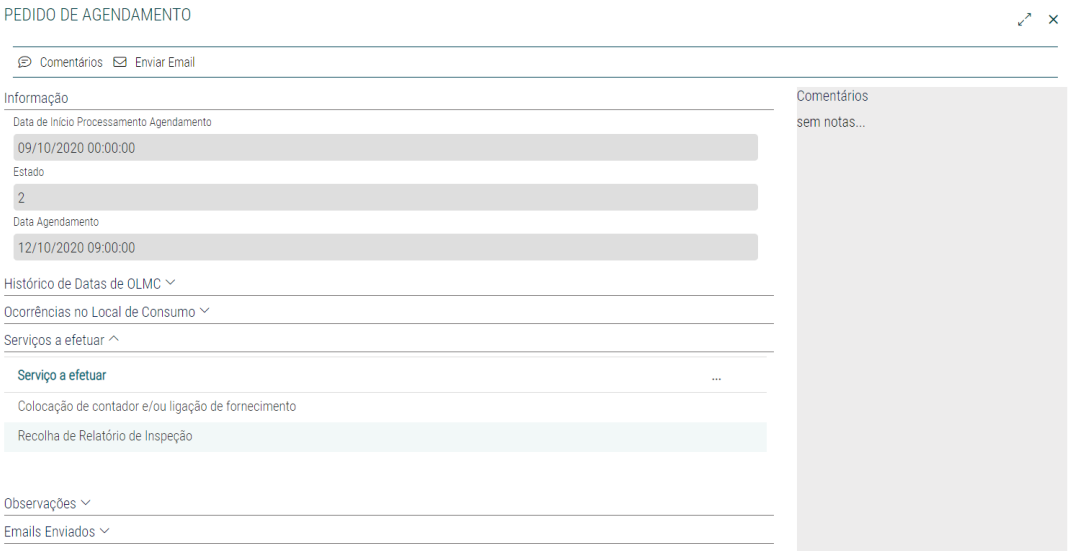

Figura 3.44: Interface gráfica de uma página PageFullEdit

3.3.5 Páginas Dinâmicas

No desenvolvimento desta aplicação foi necessário o desenvolvimento de páginas dinâmicas, como por exemplo as Dashboards.

Para o desenvolvimento deste tipo de páginas em conjunto com a DF4A existem duas opções, criar a página completamente de raiz ou definir a página com os componentes que a *framework* permite e desenvolver os componentes com o resto da informação que é necessária.

No caso deste projeto foi possível reaproveitar recursos e usar componentes da *framework* e depois desenvolver o que era necessário.

```
01. <div id="pf_body">
02.     <%= cpcit.web.Dashboard.HtmlRender("512003.0", new object[] { "Company Code",
03.         _cuc.Company_Code, "Employee Code", "", "Username", _cuc.Identity_Name  }) %>
04. </div>
```

Figura 3.45: Excerto de código HTML para a criação dos Tiles

Na figura 3.45 podemos ver que a div está a ser criada através da *framework* com recurso a uma página XML.

```
<Pagina IDPagina="513003.0" Title="pt-PT=Dashboard;en-EN=Dashboard, New;es-ES=Dashboard">
    <Table Name="Contract" DataKeyNames="EntryNo" />
    <Dashboard>
      <Tiles ID="1" Parametros="Employee Code,{EmployeeCode},CompanyCode,{CompanyCode}">
        <Tile ID="10" Text="pt-PT=Pronto para Switch;en-EN=Pre Contracts Ready for Switch"
          Background-Color="#90EE90" InPageList="1">
          <Add IDPagina="513003.01" />
        </Tile>
        <Tile ID="20" Text="pt-PT=Erro de Switch;en-EN=Pre Contracts in Switch Error"
          Background-Color="#FFB6C1" InPageList="1">
          <Add IDPagina="513003.02" />
        </Tile>
        <Tile ID="30" Text="pt-PT=Erro de BC;en-EN=Pre Contracts in BC Error"
          Background-Color="#FFDAB9" InPageList="1">
          <Add IDPagina="513003.03" />
        </Tile>
        <Tile ID="40" Text="pt-PT=Aguardar Welcome Call;
          en-EN=Pre Contracts to Wait Welcome Call" Background-Color="#ADD8E6" InPageList="1">
          <Add IDPagina="513003.04" />
        </Tile>
        <Tile ID="50" Text="pt-PT=Pedidos de Agendamento;en-EN=Contract Scheduling Requests"
          Background-Color="#FFFACD" InPageList="1">
          <Add IDPagina="513003.05" />
        </Tile>
        <EmptyTiles UICulture="pt-PT"><![CDATA[ <span>Não existem dados para apresentar</span>
          ]]></EmptyTiles>
      </Tiles>
      <PageList ID="1">
      </PageList>
    </Dashboard>
  </Pagina>
```

Figura 3.46: Página XML para criação de Tiles

A página XML definida na figura 3.46 mostra a implementação dos "Tiles"que irão aparecer ao utilizador, que no caso estão relacionados com o estado dos pré contratos.

Posteriormente, é então definida a parte que irá fazer parte da componente desenvolvida dinamicamente, que são os gráficos. No Anexo C está presente o excerto de código HTML usado para criar os gráficos da DashBoard, que está contido no componente Web Contratação.

Em CodeBehind é então feita a diferenciação do que irá aparecer ao utilizador caso este seja um comercial ou um chefe de equipa.

```
01. protected override void Page_Load(object sender, EventArgs e){
02.   base.Page_Load(sender, e);
03.
04.   _cuc = CustomUserClaimsManager.GetUserClaims();
05.
06.   if (!this.IsPostBack)  {
07.       this.DynamicFormParameters.SetDefaultValues();
08.       DataSet ds = cpcit.services.proxy.CoreDatabase.
09.       Table_GetList(Tenant.Id, "Contract_TeamMember", "Flag Manager",
10.                     new object[] { "Username", _cuc.Identity_Name }, "");
11.
12.       if (ds != null && ds.Tables[0].Rows.Count > 0) {
13.           if (Convert.ToBoolean(ds.Tables[0].Rows[0]["Flag Manager"])) {
14.               AddBars();
15.           }
16.           else {
17.               AddLines();
18.           }
19.       }
20.       if (this.DynamicFormParameters.Controls.Count > 0) {
21.           foreach (string scol in this.QUERYSTRING.Keys) {
22.               this.DynamicFormParameters.SetValue(scol, this.QUERYSTRING[scol]);
23.           }
24.       }
25.   }
26.   else {
27.       RadGridDataMinha.Rebind();
28.
29.   }
30. }
```

Figura 3.47: Excerto de código onde é definido o tipo de gráfico que vai ser criado

Na figura 3.47 é possível ver a diferenciação do tipo de gráfico a ser criado. Na linha 13 é verificado se o valor da coluna "Flag Manager"é *true*, que significa que o utilizador é chefe de equipa ou não, e caso seja então são adicionadas barras ao gráfico, caso contrário, vai ser criado um gráfico de linhas.

Como resultado final da página é possível ver na figura 3.48 que, como é explicito, contém um gráfico de barras, o que significa que o utilizador atual é chefe de equipa.

Figura 3.48: Interface de DashBoard do Portal de um Chefe de Equipa

3.3.6 Eventos de Tabelas

A *framework* antes de fazer qualquer operação CRUD nas tabelas, vai verificar se existe algum evento criado para a tabela em específico.

```
01. public static class BeforeDelete {
02.     public static void Run(string Tenant, object[] Params, SqlCommand SQLCommand) {
03.         try
04.         {
05.             Hashtable ht = PageHelper.ParamsToHashtable(Params);
06.
07.             string companyCode = ht["Original_Company Code"].ToString();
08.             string code = ht["Original_Code"].ToString();
09.
10.             IContractRepository contractRepo = new ContractRepository();
11.             IEnumerable<Contract> contracts = contractRepo.GetContractsByTeamCode(Tenant, code, companyCode);
12.
13.             ITeamMemberRepository teamMemberRepo = new TeamMemberRepository();
14.             IEnumerable<TeamMember> teamMembers = teamMemberRepo.GetTeamMembers(Tenant, code, companyCode);
15.
16.             if(contracts != null)
17.                 throw new Exception("A equipa que está a tentar eliminar já tem contratos criados, não a pode eliminar");
18.
19.             if (teamMembers != null) {
20.                 foreach (TeamMember tm in teamMembers)
21.                 {
22.                     teamMemberRepo.Delete(Tenant, tm.CompanyCode, tm.TeamCode, tm.TeamCode, ht["Identity.Name"].ToString(), null);
23.                 }
24.             }
25.         }
26.         catch (Exception ex)
27.         {
28.             LoggingHelper.LogError($"ERROR: {nameof(BeforeDelete)}.{nameof(Run)}", ex);
29.             throw ex;
30.         }
31.
32.     }
33. }
```

Figura 3.49: Evento BeforeDelete da tabela Contract_Team

A figura 3.49 é um exemplo de um evento da tabela Contract_Team que é feito antes de eliminar um registo desta. Neste caso é verificado se existem pré contratos associados a esta equipa, e caso seja o caso é mandada uma exceção a avisar de tal ocorrência. Se não existir nenhuma excessão todos os utilizadores pertencentes à equipa são eliminados da tabela Contract_TeamMember e a equipa é eliminada da tabela.

O tipo de eventos que podem existir são:

- BeforeInsert;
- AfterInsert;
- BeforeUpdate;
- AfterUpdate;
- BeforeDelete;
- AfterDelete;

3.3.7 Funções de Tabelas

Para além dos eventos, no componente tables podem também existir funções como referido na secção 3.2.5.

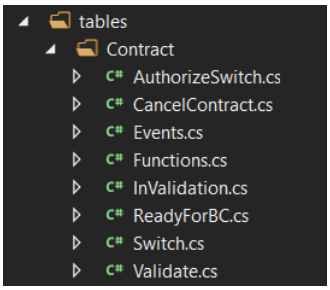

Figura 3.50: Código dinâmico da tabela Contract

A figura 3.50 é um exemplo da aparência do código dinâmico de uma tabela na *framework*.

```
class CancelContract {
  public static string Run(object[] PARAMS) {
      try
      {
          Hashtable _ht = cpcit.framework.Helpers.PageHelper.ParamsToHashtable(PARAMS);
          string tenant = _ht["Tenant"].ToString();

          using (var _conn = DataHelper.Connection.CreateConnection(tenant))
          {
              DataHelper.Connection.HandleOpenConnection(_conn);

              using (var transaction = _conn.BeginTransaction())
              {
                  IContractRepository contractRepo = new ContractRepository();
                  IScheduleRepository scheduleRepo = new ScheduleRepository();
                  Contract contract = contractRepo.GetContractById(tenant, _ht["ProcessID"].ToString());

                  try
                  {
                      ContractStatus contractStatus = contract.Status;

                      if (!(contractStatus == ContractStatus.EmValidacao || contractStatus == ContractStatus.ErroSwitch
                          || contractStatus == ContractStatus.ProntoParaSwitch))
                          throw new Exception($"Contrato encontra-se no estado {contractStatus.Description()} não pode ser anulado");

                      IEnumerable<Schedule> schedules = scheduleRepo.GetSchedulesByContractId(tenant, _ht["ProcessID"].ToString(), transaction);
                      string cancelReason = _ht["Pre Contract Cancel Reason"].ToString();
                      contractRepo.CancelContract(tenant, cancelReason, contract.ProcessId, transaction);

                      if (schedules != null) {
                          scheduleRepo.UpdateStatusByContractIds(tenant, contract.ProcessId, contract.GasContractId.ToString(),
                          contract.ProcessType, ScheduleStatus.Anulado, transaction);
                      }

                      transaction.Commit();
                      return string.Empty;
                  }
                  catch (Exception ex)
                  {
                      transaction.Rollback();
                      throw ex;
                  }
              }
          }
      }
      catch (Exception ex)
      {
          LoggingHelper.LogError($"ERROR: {nameof(CancelContract)}.{nameof(Run)}", ex);
          throw ex;
      }
  }
}
```

Figura 3.51: Função de cancelamento de pré contrato

A função representada na figura 3.51 tem como objetivo cancelar um pré-contrato.

Ao cancelar um contrato, existe a necessidade cancelar tudo o que a ele estivesse associado após a sua criação, isto é, os agendamentos. Por haver a necessidade de intervenção em várias tabelas, e por existir interesse que se um dos processos falhar o outro falhe também, então existe a necessidade do uso de uma *transaction*.

Por esse mesmo motivo é que tudo é feito dentro de um *try-catch*, para caso exista a ocorrência de uma excessão, ser feito *rollback* na *transaction*, e deste modo não fazer alterações na base de dados.

Um pré contrato só pode ser cancelado se o seu estado for "Em Validação", "Erro de Switch"ou "Pronto para Switch". Caso isto não aconteça é enviada uma exceção que irá alertar o utilizador de tal situação.

Por outro lado, se o pré contrato estiver num estado válido para cancelamento, então o contrato é cancelado e caso este tenha agendamentos associados estes são anulados.

```
1  <Pagina IDPagina="512001.21" Title="pt-PT=Anular Pré Contrato;en-EN=Cancel Pre Contract">
2      <Table Name="Contract" DataKeyNames="ProcessID" />
3      <Formulario>
4        <Area Cols="1" Rows="1">
5          <Column ID="1" Name="PreContractCancelReason" Required="true" MultiLine="true"
6            Rows="3" />
7        </Area>
8        <Hidden>
9          <Add Name="Process ID" />
10       </Hidden>
11     </Formulario>
12     <ActionButton>
13       <Add Text="pt-PT=Anular;en-EN=Cancel" Action="wcf"
14         WcfFunction="Contract.CancelContract" />
15     </ActionButton>
16   </Pagina>
```

Figura 3.52: Página XML onde a função CancelContract é usada

Na linha 14 da figura 3.51 é chamada a função referida. Para isso é apenas necessário indicar o nome da tabela, seguido de um ponto e o nome da função que se pretende que seja executada.

Capítulo 4

Avaliação e Experimentação

Considerando a importância da solução sugerida e os sujeitos interessados na sua boa implementação, alguns métodos podem ser utilizados na sua avaliação. Como fatores de avaliação destacam-se:

- Satisfação do utilizador (pessoas envolvidas no processo);
- Verificação do cumprimento dos requisitos não funcionais da solução referidos na secção 3.1 do capítulo 3.

4.1 Satisfação do utilizador

A satisfação do utilizador avalia o uso da nova solução no processo de criação de pré contratos pelas pessoas envolvidas: comerciais, chefes de equipa, gestores da comercializadora.

Esse grupo de pessoas está na primeira linha para lidar com o processo e é aquele que pode dar um *feedback* mais importante e confiável. Para testar esse fator, foi criado um inquérito e fornecido a algumas pessoas desse grupo de forma a que seja dada a sua opinião sobre o novo processo implementado. Esse inquérito pode ser visto no Anexo A.

Tendo em conta que o grupo de entrevistados tem funções diferentes na organização e nem todos eles têm conhecimento técnico sobre cada etapa do processo, é de extrema importância que o inquérito tenha perguntas mais gerais e, ao mesmo tempo, tem de contribuir para uma avaliação confiável. Este inquérito foi organizado com respostas avaliadas de 1 a 5, para permitir uma análise mais objetiva do fator em questão. Os resultados da pesquisa foram analisados e contribuíram para entender o *feedback* das partes interessadas.

Para testar a satisfação do utilizador inclui-se a hipótese de as pessoas envolvidas no processo estarem mais satisfeitas e confiantes com o novo processo implementado. A hipótese nula a ser considerada é a pontuação média aceitável para cada pergunta. As respostas ao inquérito, organizadas por níveis de 1 a 5, como já foi referido, foram analisadas e a média de cada pergunta comparada à pontuação média definida. Essa comparação permitiu concluir se as pessoas envolvidas no processo estão satisfeitas com o seu novo serviço.

Após discussão com os responsáveis, chegou-se à conclusão que a pontuação média aceitável das perguntas por níveis 0 a 5, deveria ser que a soma das percentagens com valores 1 a 3 não deveria ser superior a 25%, ou seja, sendo um total de 12 indivíduos, não poderão existir mais de 3 pessoas a responder com um valor menor que 4 à questão.

Através da análise das respostas do questionário é possível perceber os pontos fulcrais a necessitar de melhorias ou alterações. Este questionário apresentou 12 respostas às questões relativas à satisfação e usabilidade do sistema, e no Apêndice B é possível ver os resultados.

Como resultados a destacar do inquérito realizado tem-se:

- Todas as questões cumpriram o objetivo relativamente ao valor de respostas inferiores a 4 ser menor que 25%
- Durante a execução do guião, todos os utilizadores responderam com não à questão "Durante o guião alguma das operações não correu da forma esperada?";
- A questão "De 0 a 5 indique o grau de confiança que sente ao utilizar este sistema."foi a que teve pontuações mais baixas;
- A questão "Considera que o novo sistema veio diminuir as perdas de tempo durante a criação de contratos?"obteve pontuações mais altas;

Através destes resultados podemos concluir que os objetivos gerais quanto à satisfação e usabilidade do sistema foram cumpridos.

É também relevante mencionar o facto da pergunta quanto à confiança de utilização do sistema ser o ponto que obteve resultados inferiores. Isto deve-se ao facto dos trabalhadores estarem familiarizados com os formulários em papel e terem o controlo total das operações, enquanto que agora há operações automatizadas.

Para além disso, torna-se importante referir que o ponto com pontuação superior foi relativamente à diminuição de perdas de tempo durante o processo de criação de contratos. Sendo isto um dos objetivos principais da dissertação em causa, este *feedback* torna-se muito importante para a equipa de trabalho.

4.2 Verificação do cumprimento dos requisitos não funcionais

Quanto ao segundo ponto, este abrange a hipótese de os requisitos não funcionais serem cumpridos ou não, especificamente:

- **Métrica 1**: O lado do servidor deve ser resiliente garantindo alta disponibilidade, seguindo um SLA de *two nines* (99%);
- **Métrica 2**: Fiabilidade dos dados enviados para OLMC superior a 90%;
- **Métrica 3**: Tempo médio de resposta dos endpoints do serviço REST inferior a 1 segundo;
- **Métrica 4**: Capacidade de processamento até 10 pedidos por segundo nos endpoinst do serviço REST;
- **Métrica 5**: Tempo médio de resposta dos endpoints do serviço SOAP inferior a 1 segundo;

A hipótese nula equivale aos valores definidos nos requisitos que têm de ser cumpridos. Os resultados obtidos serão então comparados com os valores definidos inicialmente. Esta análise permitiu concluir o cumprimento destes requisitos, e perceber se a solução desenvolvida cumpre com o que foi acordado com o cliente.

4.2.1 Métrica 1

De forma a responder a este requisito foi utilizado o Uptime Robot, que monitoriza o *uptime* e *downtime* dos serviços utilizados. Esta monitorização está a ser feita ao ambiente de produção, que está a ser monitorizado de 5 em 5 minutos.

Este serviço é um serviço externo grátis que guarda os logs da monitorização durante 2 meses, e alerta sempre que um serviço se encontrar em baixo.

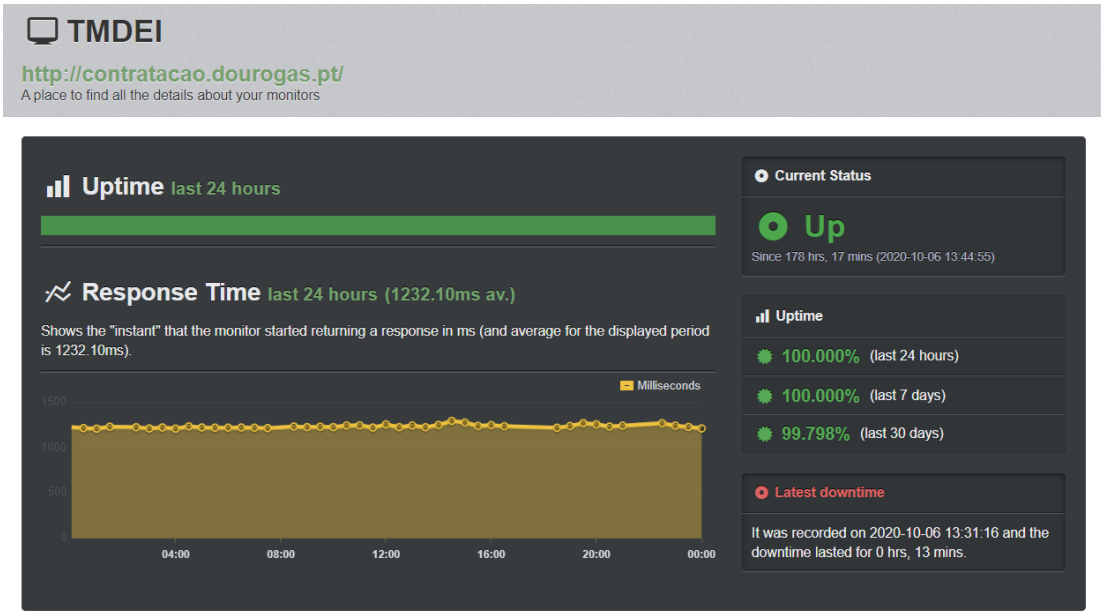

Figura 4.1: Métrica de Disponibilidade

Como é visível na figura 4.1, podemos ver que o resultado obtido nos últimos 30 dias é positivo, sendo então possível considerar este requisito como válido e aceite.

4.2.2 Métrica 2

Para analisar este requisito tirou-se partido das tabelas Contract_Logs e da Contract em ambiente de produção.

Sobre a tabela Contract_Logs fez-se a query:

SELECT COUNT (DISTINCT [ProcessID])

FROM Contract_Logs

WHERE Error = 1 and Type = 0

Esta query retorna a quantidade de linhas que têm o valor da coluna Error a 1 e são do Type 0, isto é, linhas em que ocorreu um erro e que são do tipo Enviado para Switch.

O valor retornado pela query referida foi de 47, ou seja, existiram 47 tentativas de envio para switch de um pré contrato em que ocorreu um erro. Após análise chegou-se à conclusão que realmente estes erros ocorreram por falha nos dados enviados.

Este valor foi dividido pela quantidade de pré contratos que já foram enviados para switch, isto é, que se encontram num estado após a autorização de switch ter sido concretizada.

Estes serão os pré contratos que já passaram pelo processo de envio para switch e são esses que, neste contexto, interessa analisar. Posto isto, sobre a tabela Contract foi executada a seguinte query:

SELECT COUNT (*)

FROM Contract

WHERE Estado = 4 or Estado = 6 or Estado = 7 or Estado = 8 or Estado = 9

Até ao dia, esta query retorna o valor de 1467 pré contratos ilegíveis para esta análise.

O valor obtido pela divisão foi de, aproximadamente, 0,02, que multiplicando por 100 dá 2%. Tendo em conta que o requisito é que a fiabilidade dos dados enviados para OLMC seja superior a 90%, pode-se dizer que este requisito foi cumprido, uma vez que o valor atual da fiabilidade de dados em ambiente de produção é igual a 98%.

4.2.3 Métrica 3

Assim como a métrica anterior, também esta foi obtida analisando os valores de produção. Para isto foram monitorizados os 3 principais serviços Rest, o de Ativação de Contrato, o de Receção de Agendamentos, e o de Confirmação de Agendamentos.

Para conseguir fazer este tipo de análise, foi usado o serviço Statful. Este é uma ferramenta de agregação e visualização de dados, reunindo fluxos de dados para demonstrar todo o espectro do negócio, em tempo real (Statful 2020).

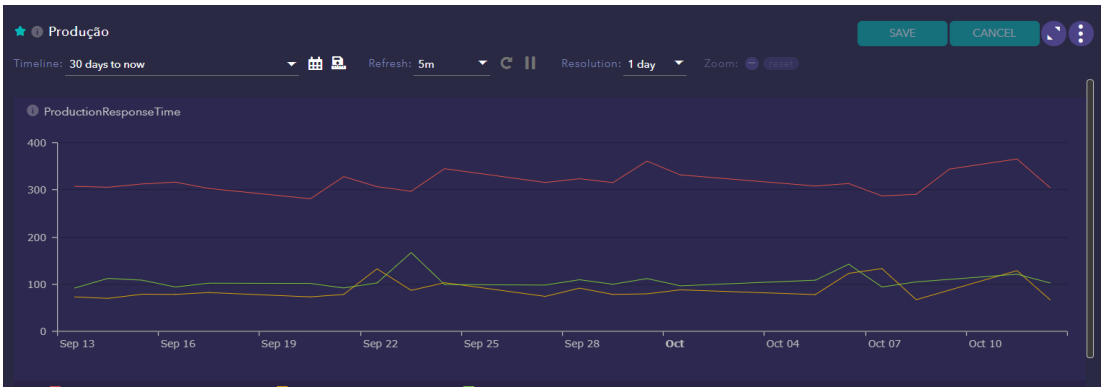

Figura 4.2: Gráfico com tempo médio dos serviços REST em produção durante 30 dias

A figura 4.2 mostra um gráfico com o tempo médio em milissegundos dos serviços REST em produção, durante 30 dias. Após a análise desta, podemos concluir que este requisito foi cumprido com sucesso, e que apesar de algumas variações de valores, o tempo médio de resposta mantém-se constante entre os 300 e os 400 milissegundos.

4.2.4 Métrica 4

Contrariamente à métrica anterior, esta foi desenvolvida apenas no ambiente de qualidade, em que era possível a realização de testes para poder verificar a veracidade deste requisito.

Com recurso aos serviços do Statful, foi também feito uma análise ao sistema, como mostra a figura 4.3.

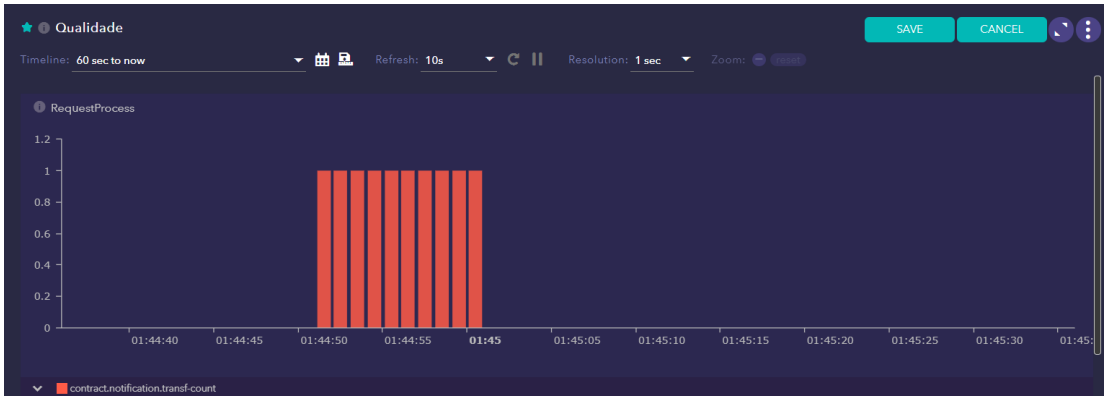

Figura 4.3: Gráfico com 10 pedidos de uma notificação num segundo

Este teste foi feito com recurso a multithreads que enviaram simultaneamente pedidos para o endpoint REST de receção de notificações do sistema no ambiente de qualidade. Sendo que a versão grátis do Statful apenas permite 10 pedidos por segundo, e sendo este o requisito que é necessário verificar se é cumprido, apenas 10 pedidos foram feitos em simultâneo.

Analisando o gráfico é possível perceber que o serviço conseguiu suportar e processar todos os pedidos, cumprindo a métrica solicitada pelo cliente.

4.2.5 Métrica 5

Para fazer a análise desta métrica, foi usado também o ambiente de produção, com o serviço Statful. Neste foi analisado o serviço SOAP que comunica com o BC e envia o contrato para este passar a ser faturado.

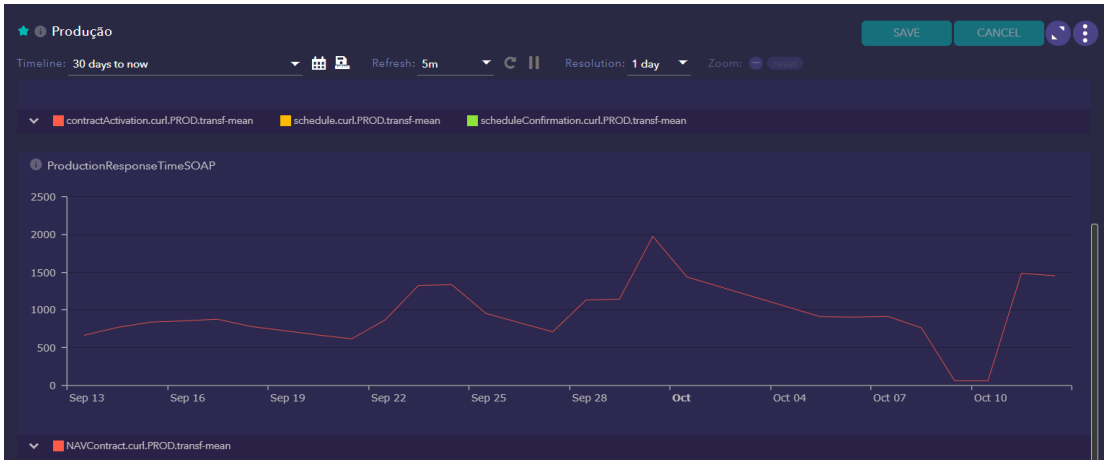

Figura 4.4: Gráfico com tempo médio dos serviços SOAP em produção durante 30 dias

Através da análise do gráfico, é possível perceber que este serviço, apesar de cumprir com o requisito do tempo médio ser inferior a 1 segundo (1000 milissegundos), é muito instável, demonstrando um comportamento um pouco errático. Apesar de este serviço estar a ser desenvolvido pela CPCIT4ALL, as razões pelas quais este comportamento acontece são alheias à equipa de desenvolvimento do portal.

Apesar disto, o serviço cumpre com o requisito estipulado pelo cliente, e deste modo pode-se afirmar que foi cumprido com sucesso.

Capítulo 5

Conclusão

O projeto documentado teve como objetivo a exploração de metodologias inovadoras para o desenvolvimento de software. Neste capítulo, todo o trabalho desenvolvido é analisado, abordando os objetivos alcançados e as dificuldades/limitações que foram sentidas ao longo do projeto.

5.1 Objetivos Concluídos

A solução implementada ao longo deste projeto, permitiu responder ao objetivo principal de criar um módulo de criação de contratos, que fizesse o que o cliente pretendia e que fosse *user friendly*.

Por ser um projeto muito ambicioso, os resultados alcançados permitem afirmar que uma sólida base de trabalho foi construída, para poder integrar outras aplicações no futuro, com uma margem de desenvolvimento e melhoria contínua.

Considera-se que a arquitetura adotada apoia princípios de reutilização e flexibilidade, que permitem o suporte de novas funcionalidades com impacto mínimo às já implementadas. Abaixo expõem-se os principais objetivos alcançados até à data, isto é, que já se encontram em produção e em utilização por parte do cliente:

- Gestão de Pré-Contratos (todos os requisitos implementados);
- API de Switching (todos os requisitos implementados);
- API de BC (todos os requisitos implementados);
- Gestão de comunicação ao cliente (GCC01, GCC02 e GCC04 implementados);
- Controlo de Qualidade (CQ01 implementado);
- Controlo de Acessos (todos os requisitos implementados)

Como é visível, todos os requisitos marcados como prioritários pelo cliente foram cumpridos e estão atualmente em produção e para além destes outros requisitos de menor importância já se encontram implementados.

5.2 Limitações e trabalho futuro

Como o contexto deste projeto é algo muito específico, um dos desafios foi a falta de informação quanto aos aspectos inerentes à usabilidade, e tudo o que tivesse a ver com problemas de negócio. Essa limitação, forçou a exploração e a auto-aprendizagem e influenciou o tempo de implementação das várias funcionalidades.

Para além disso, o facto de ser necessário usar a *framework* desenvolvida pela empresa, uma vez que era completamente desconhecida e nunca antes usada, levou a um período inicial de habituação, porém após esse período mostrou-se como sendo uma grande mais valia, pois diminuiu consideravelmente o período de implementação de front end.

Relativamente a trabalho a desenvolver no futuro, prevê-se que nos próximos meses, sejam desenvolvidos os requisitos em falta, sendo que existe uma grande possibilidade de o número de requisitos aumentar consoante a utilização dos utilizadores, devido ao aumento de interesse pelo serviço.

Bibliografia

.NET (set. de 2020). *.NET*. url: https://dotnet.microsoft.com/.

Atlassian (2020). *What is version control | Atlassian Git Tutorial*. url: https://www.atlassian.com/git/tutorials/what-is-version-control.

Bo, Tai (2019). *Build and deploy an ASP.NET core app running on IIS using Azure pipelines*. url: https://medium.com/@taithienbo/build-and-deploy-an-asp-net-core-app-running-on-iis-using-azure-pipelines-e675041f62d4.

Datadog (2020a). *IIS monitoring with Datadog*. url: https://www.datadoghq.com/blog/iis-metrics/#resource-metrics.

– (2020b). *See it all in one place*. url: https://www.datadoghq.com/product/.

ESCHBERGER, Tanja (abr. de 2018). «The fuzzy front-end of the innovation process». Em: url: https://www.lead-innovation.com/english-blog/fuzzy-front-end.

Fernández, Raquel S. e M. Ángeles Iniesta Bonillo (dez. de 2017). «The concept of perceived value: A systematic review of the research». Em: url: https://www.researchgate.net/publication/237937875_The_concept_of_perceived_value_A_systematic_review_of_the_research.

Foundation, The jQuery (2020). *What is jQuery?* url: https://jquery.com/.

GALP (2020). *Mercado de Gás Natural*. url: https://galpgasnaturaldistribuicao.pt/Gas-Natural/Mercado-de-Gas-Natural.

Galp (2020). «O que é o Gás Natural?» Em: url: https://galpgasnaturaldistribuicao.pt/gas-natural/o-que-e.

GIT (set. de 2020). *Git–distributed-is-the-new-centralized*. url: https://git-scm.com/.

IAN (jun. de 2016). *What does ACID mean in Database Systems?* url: https://database.guide/what-is-acid-in-databases/.

KendoUI (set. de 2020). *Build Better JavaScript Apps Faster*. url: https://www.telerik.com/kendo-ui.

Koen, P et al. (mar. de 2001). «Providing clarity and a common language to the fuzzy front end». Em: *Research Technology Management* 3, pp. 46–55.

Krishna (out. de 2020). *SQL vs NoSQL: What's the difference?* url: https://www.guru99.com/sql-vs-nosql.html.

Kruntchen, P (1995). «Architectural blueprints–the" 4+ 1" view model of software architecture». Em: 12, pp. 42–50. issn: 07407459. url: http://scholar.google.com/scholar?hl=en%7B%5C&%7DbtnG=Search%7B%5C&%7Dq=intitle:Architectural+Blueprints+?+The+?+4+++1+?+View+Model+of+Software+Architecture%7B%5C#%7D0.

Kurt, G. e R. Michael (2014). *Management of the Fuzzy Front End of Innovation*. Place of publication not identified: Springer.

Liberty, Jesse (2005). *Programming C#: Building .NET Applications with C#*. Place of publication not identified: O'Reilly Media, Inc. isbn: 9780596006990.

Lines, Mark (2012). *Disciplined Agile Delivery : a practitioner's guide to agile software delivery in the enterprise*. Place of publication not identified: IBM Press. isbn: 0132810131.

Mahemoff, Michael (2006). *Ajax Design Patterns*. Place of publication not identified: O'Reilly Media, Inc. isbn: 9780596101800.

Microsoft (ago. de 2018). *Design the infrastructure persistence layer*. url: https://medium.com/@priyalwalpita/software-architecture-patterns-layered-architecture-a3b89b71a057.

– (nov. de 2019a). *What is Azure DevOps?* url: https://docs.microsoft.com/pt-pt/azure/devops/user-guide/what-is-azure-devops?toc=%2Fazure%2Fdevops%2Fserver%2Ftoc.json&bc=%2Fazure%2Fdevops%2Fserver%2Fbreadcrumb%2Ftoc.json&view=azure-devops-2020.

– (nov. de 2019b). *What is SQL Server Management Studio (SSMS)?* url: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15.

– (abr. de 2020). *NET Core vs. .NET Framework for server apps."* url: https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server.

Motlik, Florian (2017). «Why You Should Use Continuous Integration and Continuous Deployment». Em: url: https://css-tricks.com/continuous-integration-continuous-deployment/.

Murray, Chuck (set. de 2007). «1 SQL Developer Concepts and Usage». Em: url: https://docs.oracle.com/cd/B28359_01/appdev.111/b31695/intro.htm.

Myerson, Judith M. (2013). *Standardize cloud SLA availability with numerical performance data*. url: https://www.ibm.com/developerworks/cloud/library/cl-SLAloadbalance-numanalysis/.

OLMC (2019). *Operador Logístico de Mudança de Comercializador (OLMC)*. url: https://olmc.adene.pt/.

Pereira, Daniel (jul. de 2016). *O que é o Business Model Canvas*. url: https://analistamodelosdenegocios.com.br/o-que-e-o-business-model-canvas/.

Peretti, Juan (jul. de 2020). *Data Layer Using the Repository Pattern*. url: https://medium.com/swlh/data-layer-using-the-repository-pattern-e32b19b04466.

Pérez, Javier Eguíluz (2019). *Introducción a JavaScript*. Place of publication not identified: www.librosweb.es.

Pigneur, Yves e Alexander Osterwalder (2003). «MODELLING VALUE PROPOSITIONS IN E-BUSINESS». Em:

PSPlus (2020). *Think you've got your requirements defined? Think FURPS!* url: http://www.psplus.ca/articles/think-youve-got-your-requirements-defined-think-furps/.

Reis, Carlos (2016). *Cadeia de Valor*.

REN (dez. de 2017). *CONSUMO DE GÁS NATURAL 11% ACIMA DO MÁXIMO HISTÓRICO*. url: https://www.ren.pt/pt-PT/media/comunicados/detalhe/consumo_de_gas_natural_11__acima_do_maximo_historico.

– (jan. de 2020). *NOVO MÁXIMO HISTÓRICO NO TRANSPORTE DE GÁS NATURAL*. url: https://www.ren.pt/pt-PT/media/comunicados/detalhe/novo_maximo_historico_no_transporte_de_gas_natural.

Rouse, Margaret (2019). *front end and back end*. url: https://whatis.techtarget.com/definition/front-end.

Schaeffer, Chuck (2020). *Agile versus Waterfall for CRM Software Success*. url: http://www.crmsearch.com/agile-versus-waterfall-crm.php.

Silva, Bárbara (2019). «Dourogás investe 58 milhões em redes de gás natural em Bragança e Vila Real». Em: *Medium*. url: https://www.dinheirovivo.pt/economia/dourogas-investe-58-milhoes-em-redes-de-gas-natural-em-braganca-e-vila-real/.

Sketchbubble (2017). *Concept Development*. url: https://www.sketchbubble.com/en/presentation-concept-development.html.

Sonarqube (2020). *Code Quality and Security*. url: https://www.sonarqube.org/.

Statful (2020). *THE ART AND SCIENCE OF DATA VISUALISATION*. url: https://www.statful.com/.

Thangarathinam, Thiru (2020). *IIS and ASP.NET: The Application Pool*. url: https://www.developer.com/net/asp/article.php/2245511/IIS-and-ASPNET-The-Application-Pool.htm.

UptimeRobot (2020). *About Uptime Robot*. url: https://uptimerobot.com/.

w3techs (set. de 2020). *Usage statistics of JavaScript libraries for websites*. url: https://w3techs.com/technologies/overview/javascript_library.

Walpita, Priyal (jul. de 2019). *Software Architecture Patterns — Layered Architecture*. url: https://medium.com/@priyalwalpita/software-architecture-patterns-layered-architecture-a3b89b71a057.

Wickramarachchi, Anuradha (ago. de 2017). *Software Architecture Patterns*. url: https://towardsdatascience.com/software-architecture-patterns-98043af8028.

Woodall, Tony (2003). «Conceptualising 'Value for the Customer': An Attributional, Structural and Dispositional Analysis». Em:

Apêndice A

Inquérito de Satisfação e Usabilidade

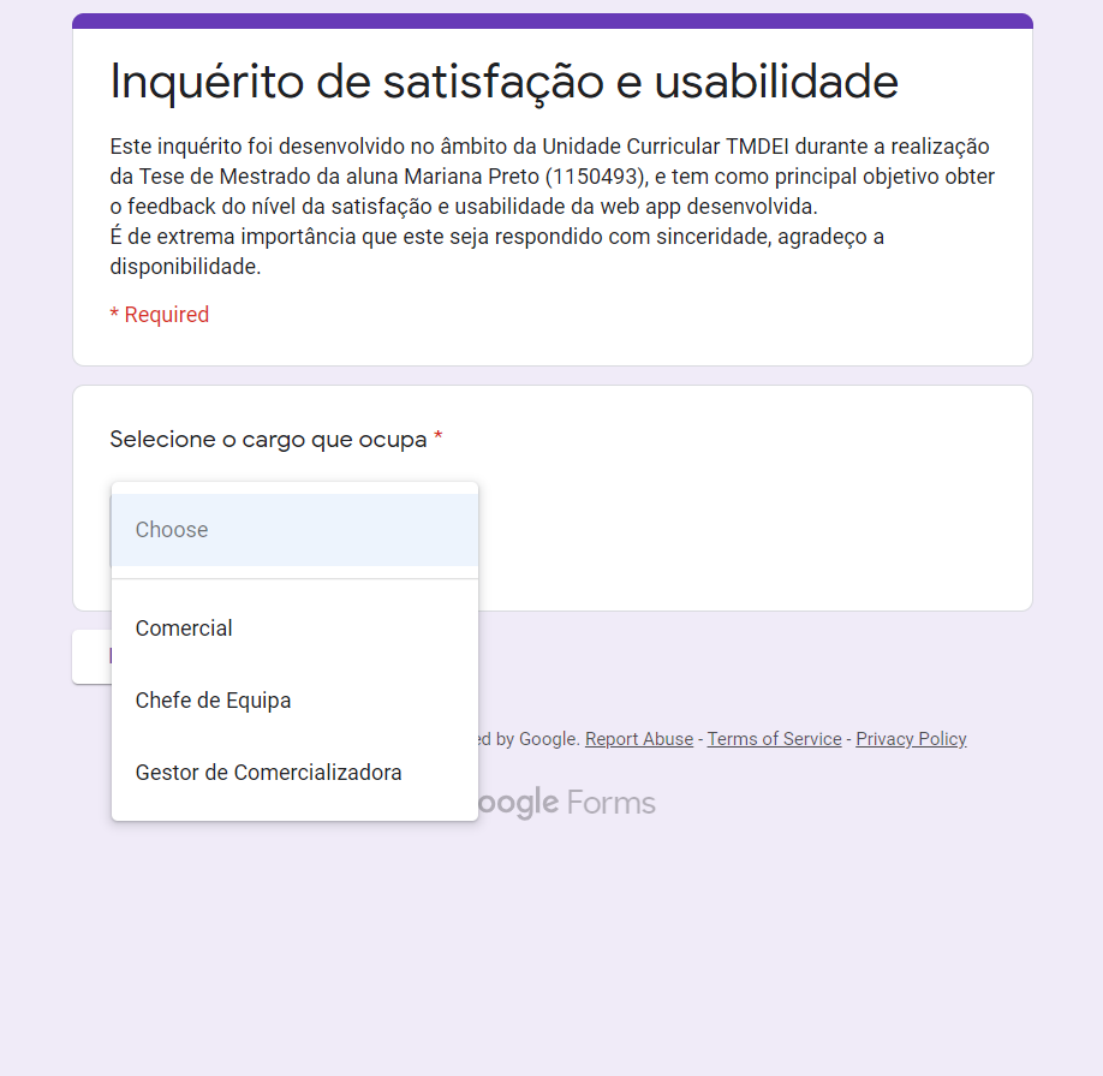

Figura A.1: Página inicial de inquérito

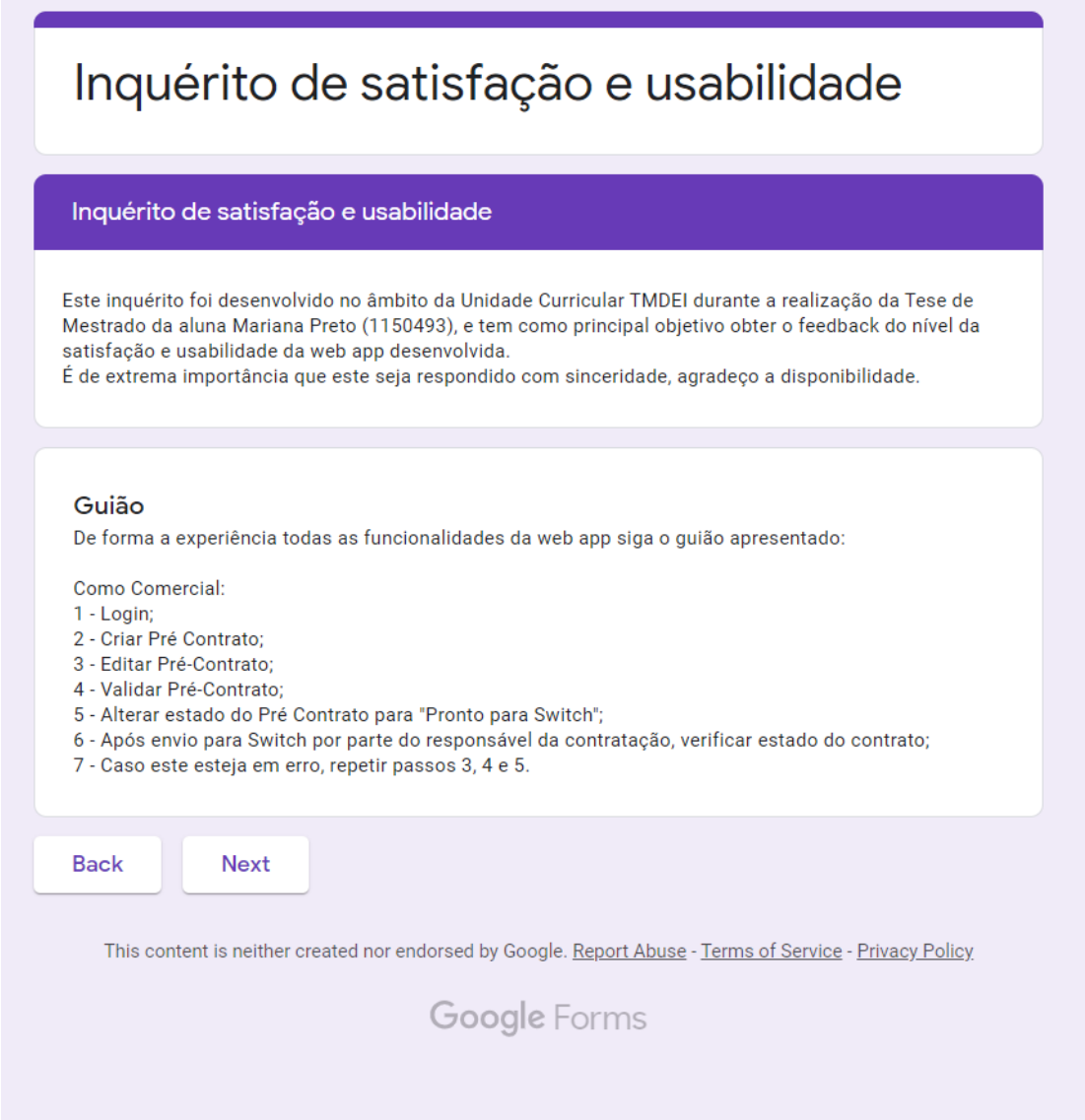

Figura A.2: Guião caso a escolha seja Comercial

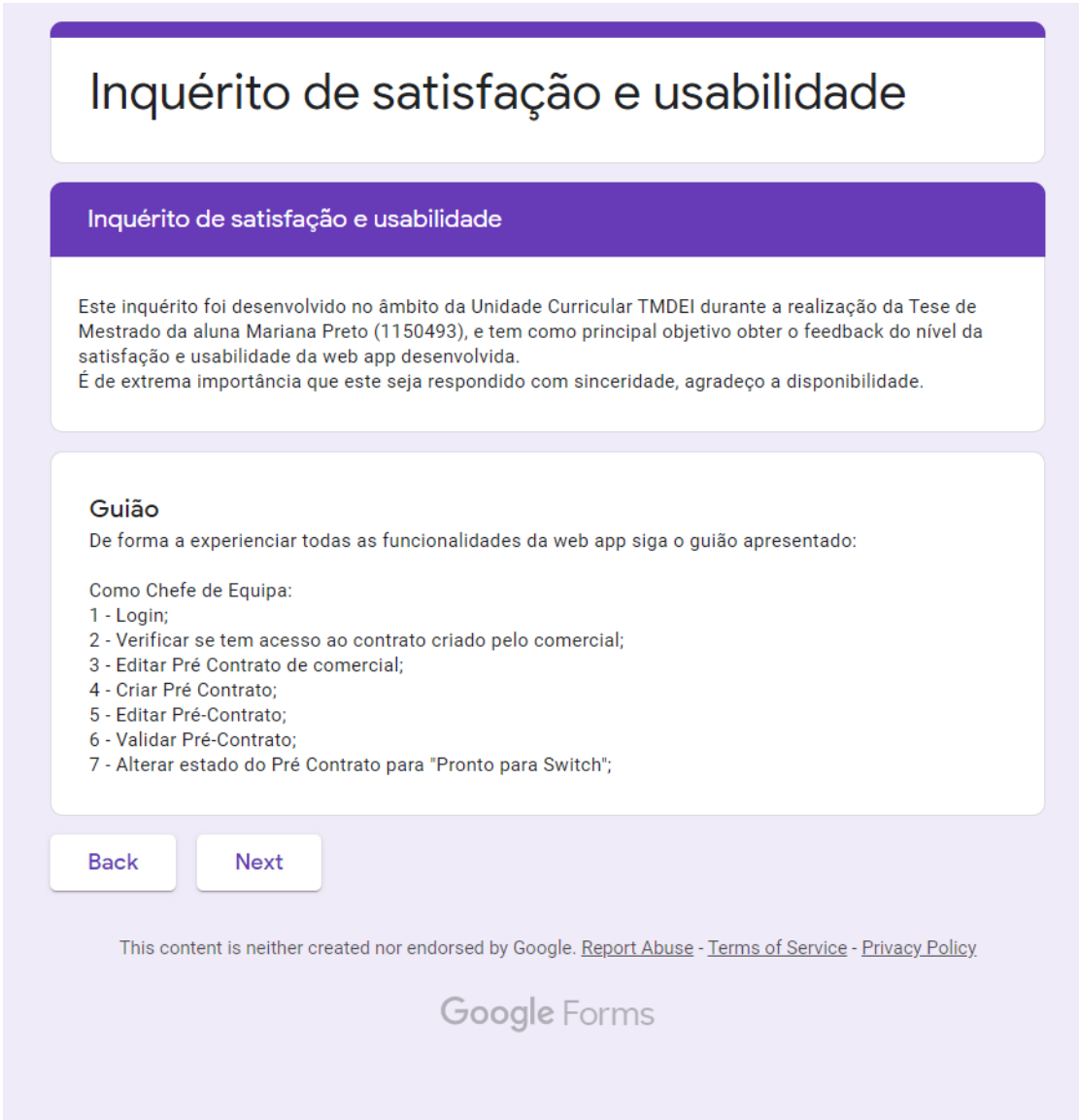

Figura A.3: Guião caso a escolha seja Chefe de Equipa

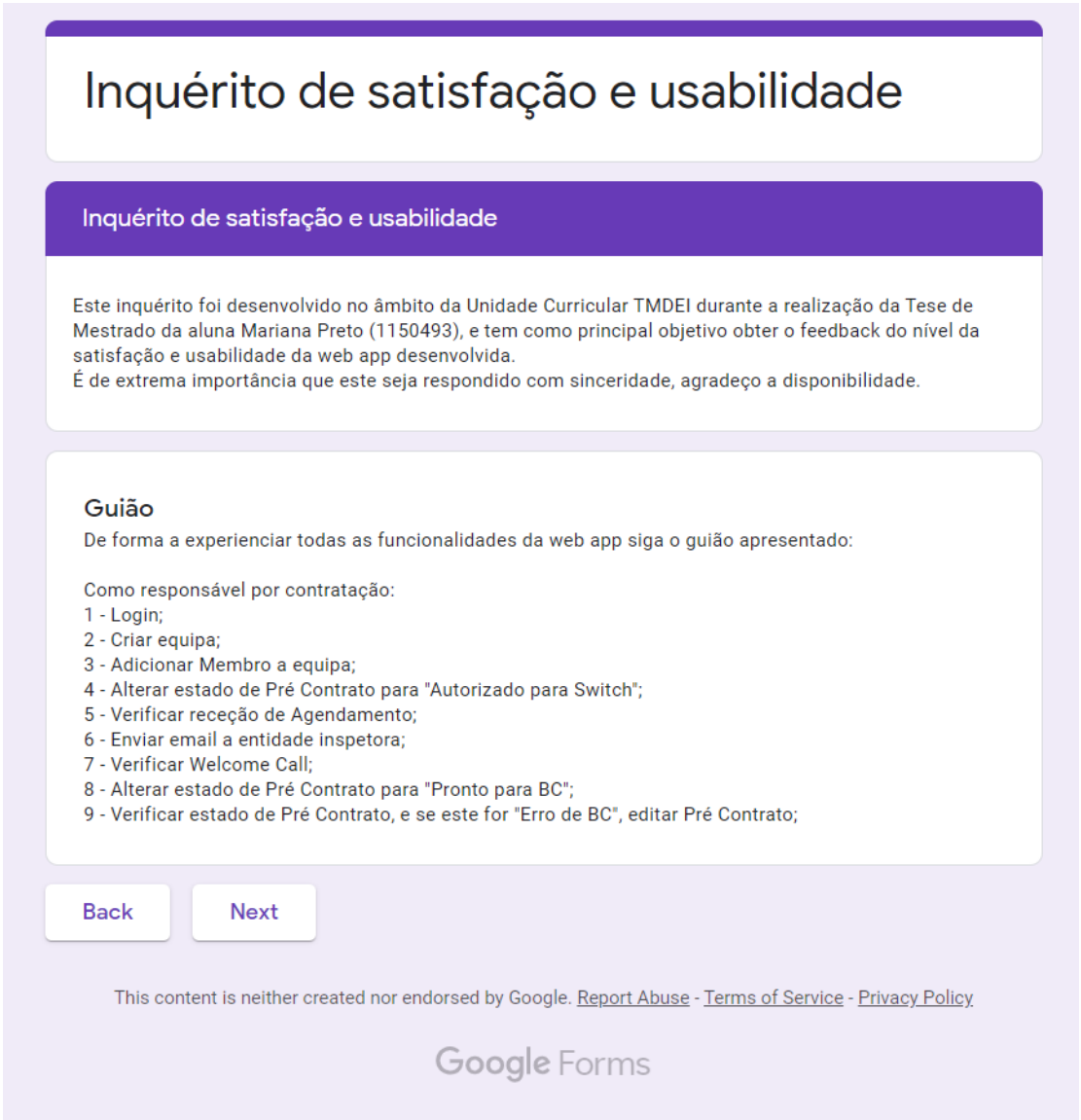

Figura A.4: Guião caso a escolha seja Gestor de Comercializadora

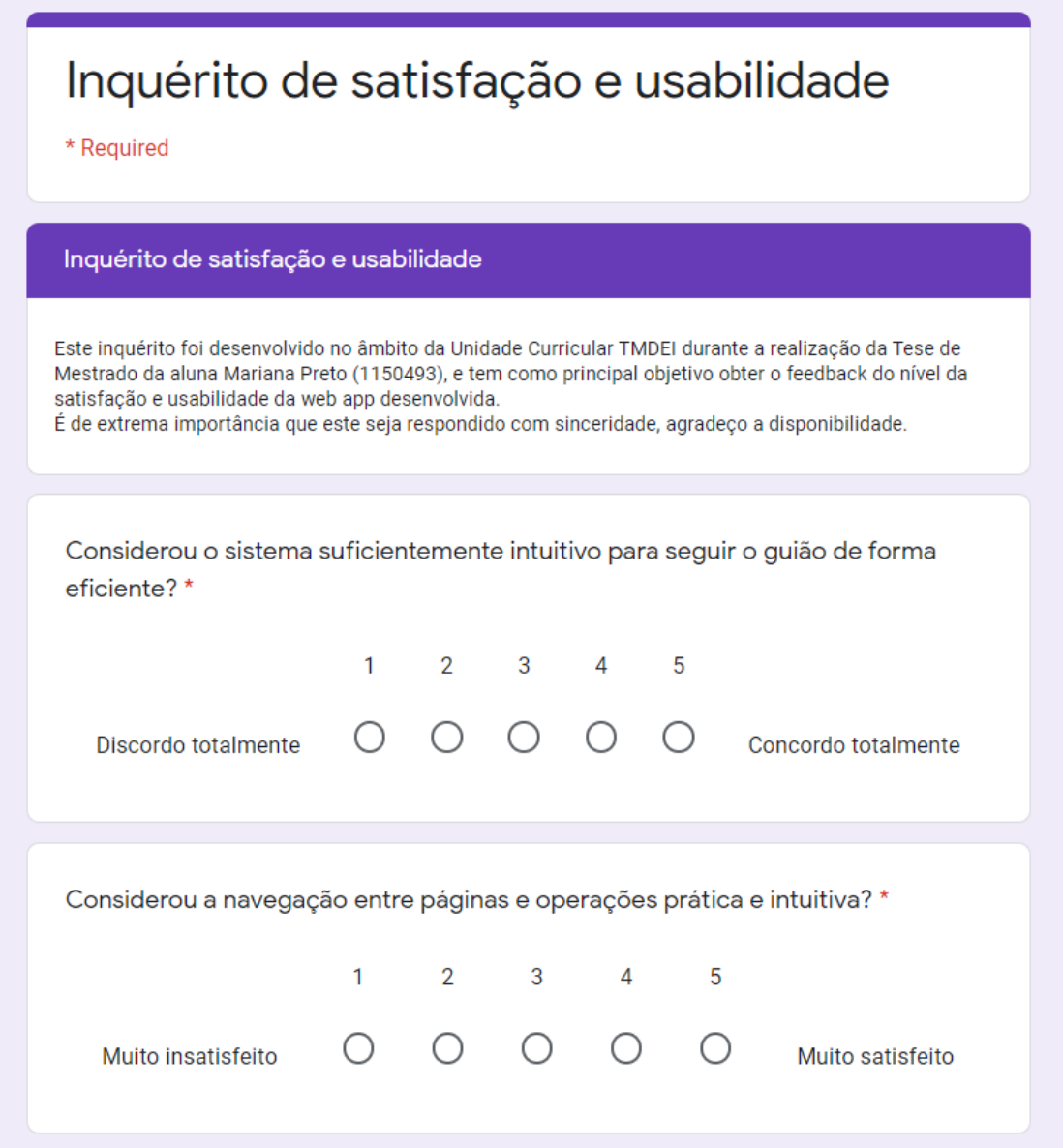

Figura A.5: Início de inquérito

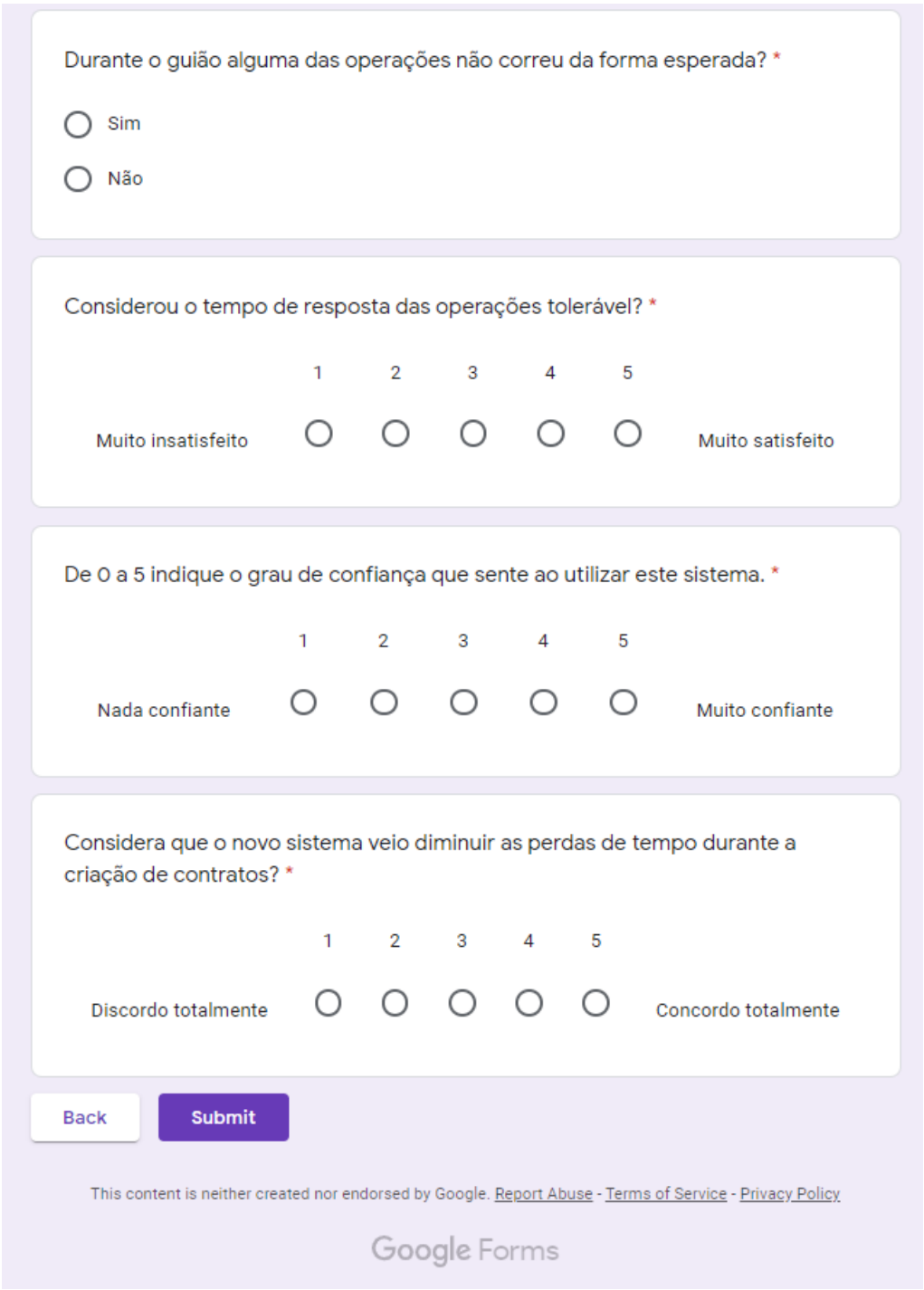

Figura A.6: Fim do inquérito

Apêndice B

Resultados do Inquérito de Satisfação e Usabilidade

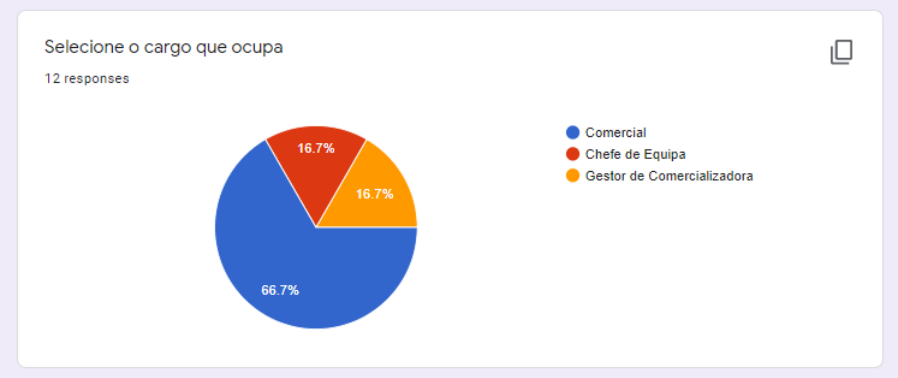

Figura B.1: Gráfico circuclar de variação de tipo de utilizador

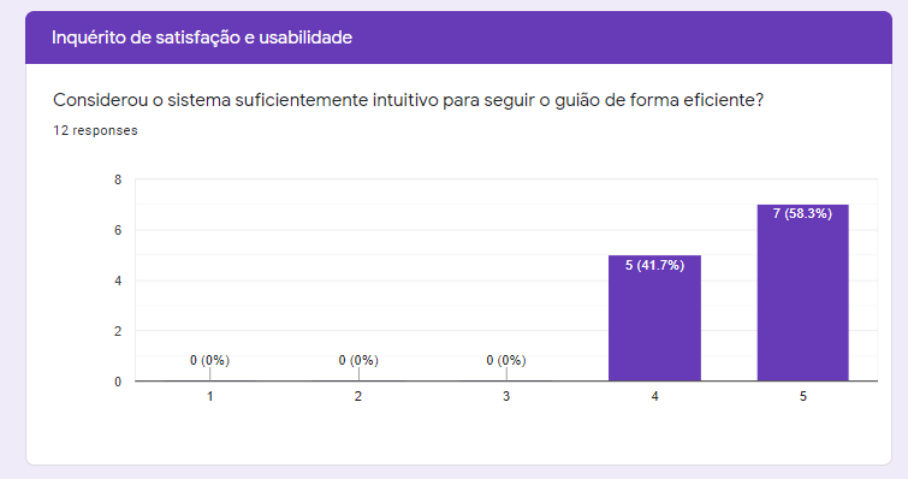

Figura B.2: Gráfico de barras com percentagem de resposta à pergunta 1

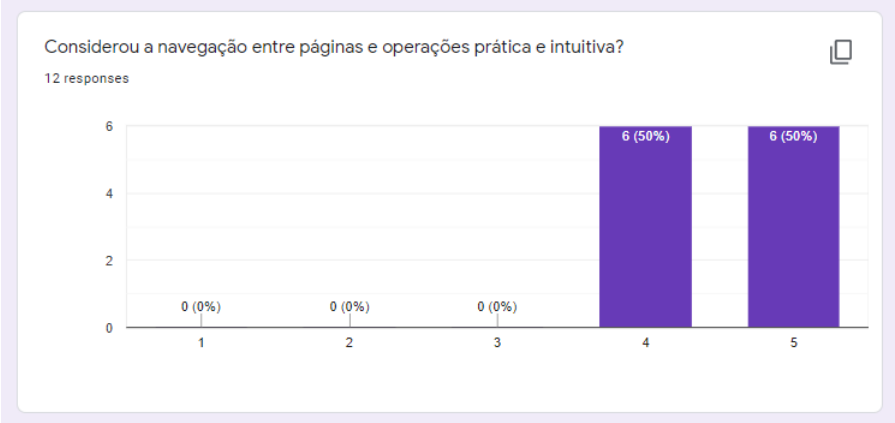

Figura B.3: Gráfico de barras com percentagem de resposta à pergunta 2

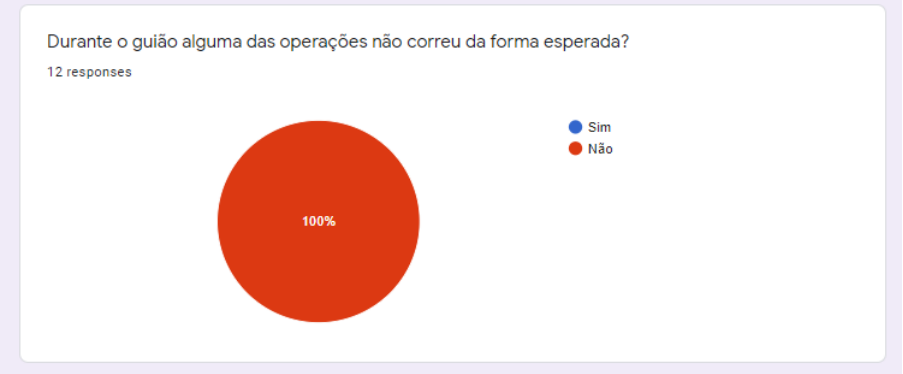

Figura B.4: Gráfico de barras com percentagem de resposta à pergunta 3

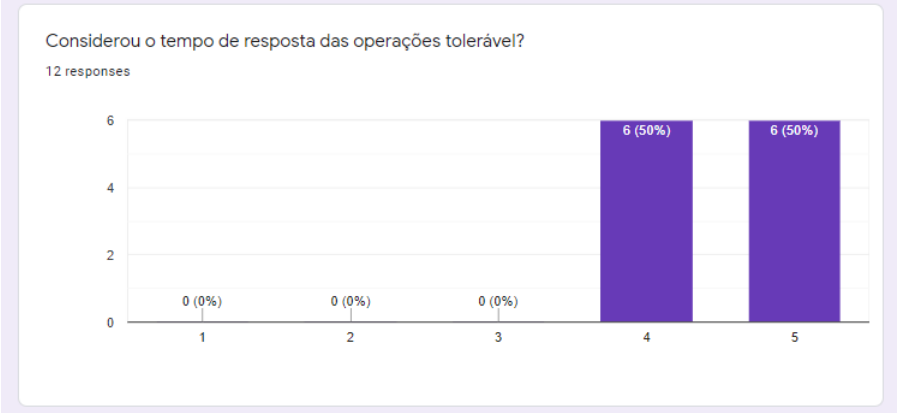

Figura B.5: Gráfico de barras com percentagem de resposta à pergunta 4

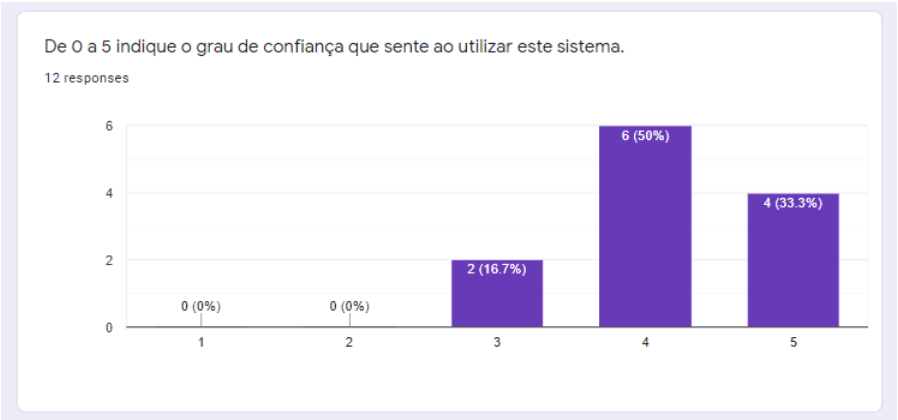

Figura B.6: Gráfico de barras com percentagem de resposta à pergunta 5

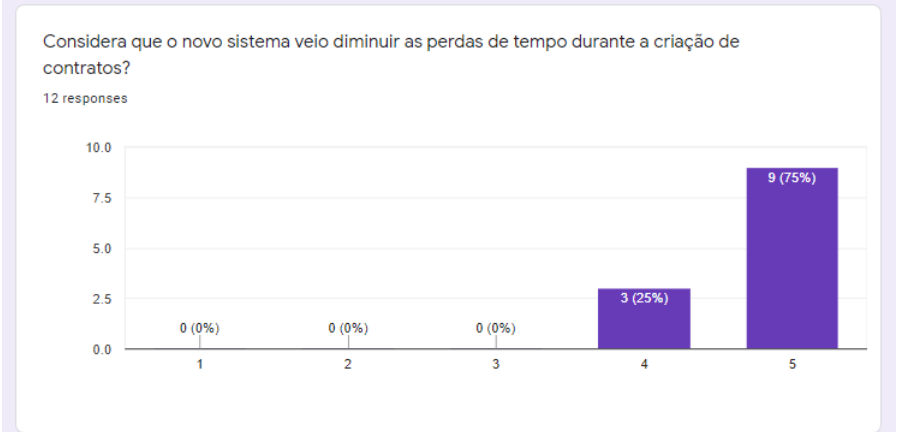

Figura B.7: Gráfico de barras com percentagem de resposta à pergunta 6

Apêndice C

Definição HTML dos Gráficos da DashBoard

```
<div id="cont_columns">
  <div id="grid" style="width: 50%; float: left">
      <cpcit:DynamicForm runat="server" ID="DynamicFormParameters" />

      <telerik:RadGrid runat="server" ID="RadGridDataMinha"
          OnNeedDataSource="RadGridData_NeedDataSource"
          OnItemDataBound="RadGridData_ItemDataBound"
          OnItemCommand="RadGridData_ItemCommand">
          <MasterTableView TableLayout="Fixed"></MasterTableView>
      </telerik:RadGrid>
  </div>
  <div id="pf_graph_container_right" style="width: 50%; float: right">
      <div id="pf_graph">
          <div>
              <telerik:RadHtmlChart runat="server" ID="Chart" Transitions="true">
                  <Pan Enabled="true" Lock="Y" />
                  <Zoom Enabled="false">
                      <MouseWheel Enabled="true" />
                      <Selection Enabled="true" ModifierKey="Shift" />
                  </Zoom>
                  <Appearance>
                      <FillStyle BackgroundColor="Transparent"></FillStyle>
                  </Appearance>
                  <ChartTitle>
                      <Appearance Align="Center" BackgroundColor="Transparent" Position="Top">
                      </Appearance>
                  </ChartTitle>
                  <Legend>
                      <Appearance BackgroundColor="Transparent" Position="Bottom">
                      </Appearance>
                  </Legend>
                  <PlotArea>
                      <Appearance>
                          <FillStyle BackgroundColor="Transparent"></FillStyle>
                      </Appearance>
                      <XAxis AxisCrossingValue="0" Color="black" MajorTickType="Outside" MinorTickType="Outside"
                          Reversed="false" DataLabelsField="label">
                          <MajorGridLines Visible="false" />
                          <MinorGridLines Visible="false" />
                          <LabelsAppearance DataFormatString="{0}" RotationAngle="0" Skip="0" Step="1">
                          </LabelsAppearance>
                          <TitleAppearance Position="Center" RotationAngle="0">
                          </TitleAppearance>
                      </XAxis>
                      <YAxis AxisCrossingValue="0" Color="black" MajorTickSize="1" MajorTickType="Outside"
                          MinorTickSize="0" MinorTickType="Outside" MinValue="0" Reversed="false"
                          >
                          <MajorGridLines Visible="false" />
                          <MinorGridLines Visible="false" />
                          <LabelsAppearance RotationAngle="0" Skip="0" Step="1">
                          </LabelsAppearance>
                          <TitleAppearance Position="Center" RotationAngle="0" Text="Nº de criações">
                          </TitleAppearance>
                      </YAxis>
                  </PlotArea>
              </telerik:RadHtmlChart>
          </div>
      </div>
  </div>
  <br style="clear: both;" />
</div>
```

Figura C.1: Excerto de código de definição HTML dos gráficos