UNIVERSIDADE
BEIRA INTERIOR

Sistema IoT para gestão inteligente de *Smart Lockers* em ambientes remotos ou rurais

Daniel Pereira Reis

Relatório do Projeto de Estágio
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutor Bruno Miguel Correia da Silva
Orientador Altice Portugal: Engenheiro Valter Guerreiro do Vale

Junho de 2022

Agradecimentos

Agradeço às entidades que proporcionaram esta oportunidade de estágio muito enriquecedora da minha experiência pessoal e profissional. Nomeadamente a Universidade da Beira Interior e a Altice Portugal.
Quero endereçar também os meus agradecimentos ao Professor Doutor Bruno Miguel Correia da Silva que fez a orientação do projeto e esteve sempre disponível para colaborar na concretização do mesmo. Agradeço também ao Engenheiro Valter Guerreiro do Vale que supervisionou a realização do projeto e deu assistência sempre que necessário, tendo um papel fulcral e facilitador do desenvolvimento do protótipo dentro da empresa.
Finalmente agradeço a família e amigos pelo apoio e suporte na duração deste estágio curricular.

Sistema IoT para gestão inteligente de *Smart Lockers* em ambientes remotos ou rurais

Resumo

O presente relatório resume o projeto e o trabalho desenvolvido no decorrer do estágio referente à disciplina de Dissertação ou Estágio em Engenharia Informática. O estágio foi proporcionado pela Altice Portugal e decorreu no Data Center desta empresa na Covilhã de Fevereiro a Março, mudando para regime remoto desde Março até Junho, inclusive, de 2022. O objetivo deste estágio foi criar uma melhoria de Logística pela utilização de *smart lockers*. Este documento possui uma breve introdução ao tema e objetivo, incluindo descrição do que o tema engloba e o estado da arte da tecnologia que se pretende desenvolver. Após esta introdução é feita então uma apresentação do projeto a elaborar durante o período de estágio, desde casos de uso a uma projeção do trabalho a executar para se tornar a implementação possível. Depois desta exposição é apresentado um resumo do sistema programado pelo estagiário e um relatório do que foi feito em cada semana do estágio. Inclui-se, ainda, alguns obstáculos a ultrapassar para beneficio e evolução positiva deste projeto.

Palavras-chave

Cacifos inteligentes
Internet das coisas
Logística
Aplicações móveis

Sistema IoT para gestão inteligente de *Smart Lockers* em ambientes remotos ou rurais

Abstract

This paper summarizes the project and the work that was carried out during the internship referring to the discipline of Dissertação OU Estágio em Engenharia Informática. The internship took place at Altice's Data Center in Covilhã from February to March, changing to remote from this month until June 2022, inclusive. Its objective was to improve Logistics through the use of *smart lockers*. This document has a brief introduction to the topic and objective of the internship, including a description of what this topic encompasses and state of the art of the technology developed. After this introduction, a presentation of the project that was made during the internship, from use cases to a projection of the work that was performed to make this implementation possible. After this exposition, a summary of the system programmed by the intern and a report of what was done weekly is presented.

Keywords

Smart-Lockers
Internet of Things
Logistics
Mobile Applications

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

API	Application Programming Interface
AWS	Amazon Web Services
BD	Base de Dados
CPU	Central Processing Unit
DTO	Data Transfer Object
FOB	Free on Board
GB	Gigabyte
GUID	Globally unique identifier
ID	Identifier
IDE	Integrated Development Environment
IOT	Internet Of Things
JSON	JavaScript Object Notation
JWT	JSON Web Token
PHP	Personal Home Page: Hypertext Preprocessor
PIN	Personal Identification Number
QR	Quick Response Code
RAM	Random Access Memory
REST	Representational State Transfer
RFID	Radio Frequency Identification
SMS	Short Message Service
SSMS	SQL Server Management Studio
UBI	Universidade da Beira Interior
URI	Uniform Resource Identifier
VPN	Virtual Private Network

Capítulo 1

Introdução

1.1 Enquadramento

Este trabalho encontra-se inserido no contexto do estágio proposto pela Altice Protugal. Este, é um estágio curricular que serve como avaliação da disciplina Dissertação ou Estágio em Engenharia Informática de 2ºciclo da Universidade da Beira Interior (UBI) com o tema de Sistema IoT para gestão inteligente de *smart lockers* em ambientes remotos ou rurais. Este relatório engloba a projeção de trabalho e pesquisa para a realização do mesmo, bem como o percurso de desenvolvimento e conclusão do projeto.

1.2 Foco e motivação

Internet das coisas é um conceito de interligação de dispositivos do dia a dia de forma que estes superem objetos tradicionais através da comunicação entre si por uma *intranet* ou *internet*. Estas soluções podem integrar análise de dados para que se tornem inteligentes de certa forma. Esta análise depende de plataformas de cloud devido a este tipo de processamento normalmente ser difícil de integrar num produto que já tem constrangimentos de espaço ainda integrar um processador de dados além de aumentar o custo de um produto também adiciona complexidade ao mesmo. Um exemplo é o caso de uma cafeteira que com este tipo de integração será muito mais cara que uma cafeteira simples, mas tornando-a num produto especial, apesar de aumentar a pegada que ocupa.
Com isto integram-se dispositivos com capacidades de comunicação pela rede de forma que seja possível programar comportamentos e análise de dados de dispositivos do nosso dia a dia. Com este paradigma é possível ter um maior conforto do individuo pela capacidade que a tecnologia nos oferece.
Aquilo a que se refere esta tecnologia é uma melhoria de um cacifo normal. Habitualmente estes têm a possibilidade de ser trancados, ou com um cadeado, ou integram uma maneira de os trancar, sendo sempre necessária uma chave para os abrir. Nestas soluções além de haver a necessidade de existir um objeto físico para dar acesso a um individuo ao conteúdo do cacifo, têm o problema de já existirem vários métodos para abrir estas soluções, causando um grande problema de segurança ao utilizador do cacifo caso este não possua uma solução para todas as contra medidas existentes para a abertura destas trancas dos cacifos.
Assim, a solução criada por esta tecnologia elimina a necessidade de existir uma chave ou tranca física que tenha de acompanhar o utilizador do cacifo, sendo estas substituídas por chaves virtuais, como códigos, gestos ou algo semelhante que será inserido no sistema via teclado ou câmara respetivamente. Apesar disto, este tipo de acesso apresenta ainda outras

vantagens para o utilizador como a possibilidade de abrir um cacifo remotamente caso este possua uma ligação à rede da internet, revolucionando a partilha de acesso a estes cacifos.
Esta solução também tem os seus problemas que substituem os da solução física, por exemplo o memorizar ou anotar destas chaves que podem ser perdidas e a possibilidade de ciberataques. Estes problemas comparativamente são mais improváveis que os da solução física. Isto deve-se à necessidade de informações do cliente, como passwords comuns do utilizador, ou por um ataque especializado ao sistema. Apesar de serem mais improváveis que problemas com um simples cadeado, estas lacunas deste tipo de sistemas também têm de ser tidas em conta.
A ideia de *smart lockers* tem surgido também devido ao crescimento das grandes cidades, sendo a logística de comércio digital um dos fatores motivadores desta tecnologia. Isto deve-se ao constante ritmo de inovação pretendido pelos clientes e empresas. Hoje em dia, existem vários problemas logísticos, como, por exemplo, a dificuldade na conclusão de uma entrega por constrangimentos de horários dos clientes ou dos distribuidores, bem como, o facto de haver limitações nas estruturas do setor de transporte existentes, seja em quantidade de viaturas ou de disponibilidade de tempo para todas as solicitações. Estes limites ocorrem também em termos de capacidade de acomodação de veículos como também da sua expansão, sendo estas limitadas pela utilização e pelo espaço físico respetivamente. A expansão destas estruturas é muitas vezes impossível devido ao facto de serem limitadas pelos edifícios das cidades.
Com a IoT, ou internet das coisas, temos um meio de criação de uma nova perspetiva de logística para os problemas apresentados. Com este conceito de IoT temos um grande potencial de acessos remotos, mudanças de chave em caso de problemas de segurança, chaves de uso único. Ou seja, a dinâmica de acessibilidade, segurança e partilha, é modificada pelo que se pode fazer com a tecnologia ao nosso dispor.
Introduzidas estas tecnologias, a Altice, empresa que propõe o tema de estágio que é aqui relatado, pretende criar uma solução de *smart lockers* de forma a revolucionar internamente a logística da empresa. Isto inclui clientes da própria empresa, bem como os funcionários que mantêm as atividades da empresa e interagem com produtos que estes necessitam. Esta mudança de paradigma sugere a introdução de um novo nicho de mercado à empresa e uma distribuição de produtos tanto da própria empresa como de terceiros.

1.3 Objetivos do Estágio

Ao longo do estágio, pretendeu-se colaborar com a empresa Altice no sentido de agilizar a logística interna com a criação e desenvolvimento do conceito de smart lockers. Entende-se que dividir produtos por vários pontos, em vez de manter a sua centralização num único ponto de recolha, permite a sua recolha quando sejam necessários ou quando exista a possibilidade de os recolher. Esta flexibilidade dá resposta ao problema que, por vezes, se apresenta devido a incompatibilidades de horário ou imprevistos que surjam, impossibilitando o indivíduo a que se destina um produto de estar presente para recolhê-lo no horário agendado.
A solução proposta visa não serem necessárias pessoas a monitorizar os produtos para a

sua distribuição. Estes serviços não estão sujeitos a um horário de abertura, facultando o acesso a estes sistemas vinte e quatro horas por dia. A única necessidade encontrada por esta solução é a manutenção física dos dispositivos, sendo necessária a presença pontual de alguém apenas para assegurar o bom funcionamento e integridade física da solução.
Para alguns casos de manutenção são necessários às funções dos técnicos e administradores de hardware físico da empresa algumas ferramentas e aparelhos. Os *smart lockers* também aqui poderão constituir uma mais valia, pois o facto de estes materiais serem distribuídos por vários pontos numa dada área em vez de estarem num ponto central pode ter alguns benefícios. Nomeadamente minimiza as deslocações que, para além de criar um acréscimo de custos de manutenção e de distribuição de ferramentas vitais a certos trabalhos, também cria problemas referentes à área ambiental que surge como consequência da queima de combustíveis.

1.4 Resumo e Estrutura do Trabalho

Esta secção demonstra a estrutura e resume cada capítulo e objetivos da inserção destes na projeção do estágio. A seguinte lista ilustra os vários capítulos sugeridos para estrutura do trabalho:

1. Introdução, noções base para entendimento do contexto do projeto e introdução de temas base que suportam o tema de *smart lockers*.
2. Estado de arte, resume o estado de investigação do tema presente e a aplicação prática da tecnologia em contexto empresarial com as suas diferenças.
3. Engenharia de software, ilustra aquilo que se entende como pertencente à realização e planeamento das soluções, incluindo uma descrição do problema a resolver e as diversas soluções necessárias para a sua resolução.
4. Planeamento de estágio, descreve as soluções necessárias por forma a ilustrar a ordem de prioridade de implementação no espaço temporal e previsão de alguns problemas que possam surgir e afetar estas projeções.
5. Desenvolvimento e Implementação, descreve a solução final que foi desenvolvida no decorrer do estágio.
6. Execução do Estágio, é um resumo semanal do trabalho que foi realizado nas 17 semanas de estágio, sendo incluídos relatórios de reuniões e alguns pormenores de factos que levaram a certos desenvolvimentos.
7. Conclusões e sugestões de Melhoria, faz o resumo do que foi aplicado pelo estagiário no estágio e melhorias que podiam ser aplicadas ao prototipo final do projeto.

Capítulo 2

Estado da Arte

2.1 Introdução

Introduzida a tecnologia e termos necessários para a compreensão do tema do estágio, segue-se a análise do estado da arte no momento de escrita deste relatório. Para seleção dos artigos e aplicações da tecnologia que será utilizada como base do projeto de estágio tem-se como importantes as necessidades do público em geral e possibilidades de avanços tecnológicos que tenham promovido a evolução da tecnologia nos últimos 3 anos. Assim os artigos e aplicações práticas da tecnologia por parte de investigadores e empresas até ao dia de escrita do relatório encontrados pela pesquisa, e relevantes para o tópico de *smart lockers* serão incluídos neste capítulo.

2.2 Estudos e artigos no tópico de *smart lockers*

Nos últimos anos a necessidade logística de grandes urbanizações é cada vez mais complexa, especialmente o processo logístico de contacto final de entrega das mercadorias ao consumidor. Esta pressão por parte do consumidor tem sido cada vez mais sentida pelas das empresas com o crescimento do comércio *online* [1] [2].
Com este crescimento verifica-se também que, tanto em ambientes de acesso público, no caso de universidades [3], ginásios a ambientes privados, como empresas [4] existe a necessidade da guarda de bens dos indivíduos que frequentam estes espaços, sejam estes clientes ou colaboradores, respetivamente.
Indústrias, como a indústria farmacêutica, revelam especial interesse neste tipo de tecnologia. Segundo o estudo [5] realizado no contexto de África do Sul, isto deve-se à comodidade oferecida pelo serviço, que possibilita uma melhor adesão ao levantamento de medicamentos prescritos, levando a um melhoramento da saúde pública. Este serviço é também extremamente útil a pessoas com doenças crónicas, devido ao facto de estas, muitas vezes, requererem medicação regular. Também se constatou no mesmo estudo que esta modalidade melhora o custo para os clientes, não sendo necessário um colaborador da farmácia no local, sendo apenas necessária a receita médica para levantamento de medicamentos nestes postos.
Existe também um potencial de melhoria de partilha de artigos entre indivíduos. Uma plataforma, como *smart lockers*, cria um ponto seguro para o depósito de bens sem necessidade de vigia destes. Segundo a referência [3], estes cacifos foram utilizados desta mesma maneira na Tailândia, para a partilha de artigos entre alunos.
Assim, a tecnologia de cacifos inteligentes é uma possível forma de resolver vários problemas logísticos, como apresentado, pela sua integração tanto em grandes urbanizações e estendendo-

se a zonas rurais [6]. Isto permite uma maior flexibilidade de acesso do cidadão a produtos, sem o constrangimento temporal da presença do recetor, além de permitir a uma redução económica por parte das empresas em recursos humanos para a organização da logística a nível fino. O que se pretende com os *smart lockers* é que um produto seja acessível de forma fácil e com alta disponibilidade, por um serviço ubíquo e acessível com a mesma ubiquidade na guarda de bens como serviços já oferecidos [7], no caso de cacifos de locais de serviços públicos com chaves físicas.

Para comprovar a necessidade de aplicação da tecnologia referida, realizaram-se estudos que levam a inquéritos feitos em pesquisa da opinião pública do consumidor sobre este tipo de serviços. Segundo os artigos [8][9][10][11][12][13], um utilizador com a utilização deste tipo de serviço, valoriza e compreende imediatamente o valor desta tecnologia, em termos de conveniência, privacidade, segurança e fiabilidade do serviço. Estes estudos também têm como foco o facto de esta ser uma nova forma de interação de indivíduos com tecnologia, está também incluída a área de pesquisa de interação pessoa máquina, além do conceito logístico já referido.

Smart lockers, entendem-se então como úteis mas a questão de não serem utilizados de maneira mais comum hoje em dia mantém-se. Assim devem tomar-se vários fatores. Deduz-se que este tipo de serviço estar muito dependente da sua localização para a sua utilização por indivíduos é, assim um dos principais. Numa tentativa de melhorar a aceitação e uso de *smart lockers*, artigos como [14] abordam este requerimento. Este, em específico, foca-se no prever, através de simulações de modelos organizacionais com inteligência artificial, as melhores localizações para criação de eventuais cacifos numa zona por minimização de custos com a maior disponibilidade possível. Para melhorar esta acessibilidade de outras formas, além de localizações fixas, os estudos [15][16], defendem a integração de *smart lockers* com transportes públicos e estruturas associadas, nomeadamente o metro. Desta forma, a acessibilidade destes serviços na rotina dos cidadãos, aumenta pela sua disponibilidade nas suas viagens diárias.

Os benefícios de *smart lockers* não se limitam apenas a acessibilidade de produtos ao cliente final, uma fator que melhoraria no nosso dia a dia seria na área de emissão de gases. Segundo artigos publicados [15][16][17][18] as emissões seriam reduzidas devido à simplificação que um sistema de cacifos proporciona a empresas logísticas. Isto não se deve só à melhoria da rota dos transportes, mas também a entregas serem possíveis em horários não de ponta, evitando que veículos se mantenham em trânsito urbano nestas horas, prolongando os seus gastos de combustíveis fósseis. Além disto, o melhor aproveitamento de tempo de cada veículo, também cria uma melhoria de utilização destes. Isto reduz o número total de veículos que circula para o propósito de distribuição de produtos, e reduz a capacidade necessária das infraestruturas.

A reutilização de recursos também seria possível com esta tecnologia, pelo aproveitamento de locais e estruturas já existentes para a aplicação de cacifos nestes. No caso do artigo [19] menciona-se a possibilidade de reutilizar armários de escritórios para alojar os *smart lockers*. Neste caso de estudo, justifica-se que o armazenamento de arquivos tem a tendência de se tornar um processo maioritariamente digital, o que torna estes armários dispensáveis.

Estes, assim podem ser reutilizados para o alojamento dos *smart lockers*, reaproveitando, desta forma, o espaço e gastos feitos na aquisição dos mesmos. Recursos como estes que se tornam obsoletos por uma razão ou outra, devem ser aproveitados para uma mesma consciência ambiental de redução de gastos de combustíveis e outros.

Mencionadas as vantagens de *smart lockers*, segundo o estado da arte, resta mencionar como a tecnologia tem sido implementada até ao momento. Várias abordagens a *smart lockers* podem ser realizadas, cada uma tendo as suas vantagens e desvantagens. Num estudo de configurações destes, o artigo [20], menciona um cacifo de dimensão fixa, torres modulares, cacifos modulares e grelhas de cacifos, chamadas no artigo como *π-Boxes as Mobile Modular Lockers*. Cacifos de dimensão fixa sendo cacifos que desde a sua conceção até ao fabrico, são feitos para as dimensões e número de cacifos ser fixo. Torres modulares no sentido de poderem ser acrescentados cacifos de forma modular adjacentes a um modulo de controlo, mas sendo as adições em termos de cacifos fixas pelo tipo de módulo. Cacifos modulares é o passo de adicionar-se cacifo a cacifo ao sistema consoante a necessidade, com restrições ao tipo de cacifos serem fixos, sendo estes obrigatoriamente de uma dada dimensão ou do mesmo fabricante. E finalmente a última solução proposta, sendo qualquer tipo de cacifo poder ser inserido desde que tenha capacidade de comunicação com o sistema. Existem já estudos para a otimização da solução de cacifos modulares, o artigo [21] tanto em termos económicos, fabrico e uso por parte do utilizador, isto na perspetiva de otimização do serviço, para todas as partes envolvidas.

No caso de cacifos físicos que são a implementação mais simples de todas as que foram apresentadas, os principais benefícios de implementação são o menor gasto de materiais e engenharia para eventuais expansões do cacifo. A pegada deste também se manterá fixa, não existindo a necessidade de considerar a possibilidade de expansão em mente. A principal desvantagem é que para a adição de novos cacifos será necessário utilizar a solução fechada e adicionar um novo modulo de controlo, que podia ser evitado se fosse usado outro tipo de cacifos mais adaptáveis.

A utilização de uma solução com torres modulares como mencionado proporciona a oportunidade de adicionar novos cacifos a posteriori. Este tipo de implantação tem como principal vantagem a expansão ser indefinida, sendo a sua capacidade máxima a pegada que pode ser ocupada, e a capacidade do módulo de controlo. Também apresenta uma melhoria em termos económicos, sendo que a expansão deste tipo é tipicamente mais barata que a compra de módulos físicos. A principal desvantagem é que a expansão é sempre fixa, sendo que se for necessário apenas mais um cacifo terá de ser adicionado um módulo inteiro. Será de ter em conta a localização do cacifo para o caso de vir a ser necessária a expansão, para evitar o realojar da solução quando se chega ao limite da capacidade.

A solução de cacifos modulares é mais ambiciosa, sendo a sua complexidade maior em termos de implementação. Como será de esperar, este tem como principal necessidade acompanhar a expansão. Ou seja, esta solução permite que o número de cacifos seja igual ao número máximo de artigos ou conjuntos de artigos a guardar. Também não necessita de adição de módulos de controlo adicionais em relação à solução fixa. O lado negativo é que este tipo de solução pode ser mais dispendioso para poucos cacifos, pela engenharia da solução ter de

assumir, desde o início, uma possível expansão. Outro ponto negativo é que estes cacifos têm de ter as mesmas interfaces de comunicação entre eles, assumindo-se que os cacifos sejam provenientes do mesmo fabricante.

Finalmente a solução de π-Boxes as Mobile Modular Lockers, são a solução mais complexa que se encontra em estado de arte. Esta solução tem como principal beneficio a sua modularidade. Esta modularidade permite a adição de qualquer tipo de cacifo ao sistema desde que o módulo de controlo consiga comunicar com este. Isto também permite que a adição de novos cacifos sejam de diferentes fabricantes, apresentando uma possibilidade de ganho económico para o comprador pela diversidade de oferta e a possível melhoria de segurança e materiais de cada um dos cacifos. Estes também poderão ser adicionados da forma que seja mais conveniente. A maior conveniência deste tipo de solução é este ser um posto de deslocamento do cacifo em si, sendo o produto protegido durante o transporte pelo cacifo e depois apenas a porta do cacifo abrir do lado do cliente e posteriormente reutilizado pelo distribuidor. A principal desvantagem é que será o mais caro por cacifo para poucos cacifos.Poderá haver alguma relutância por parte de alguns fabricantes em aderirem a este tipo de modalidade, pela possibilidade de perda de clientes para a concorrência e, também, por este tipo de modalidade ser mais difícil de implementar e, como tal, mais dispendiosa que a solução de cacifos e torres de cacifos modulares.

Para facilitar a visualização da informação apresentada, a tabela 2.1 resume as informações das tecnologias utilizadas para organização dos *smart lockers*.

Para inovação dos *smart lockers* existentes, existe a pressão de integração de tecnologias emergentes no seu contexto. No caso da referência [22] existe uma tentativa teórica de integrar tecnologias de *block-chain* mas este tipo de tecnologia tem, aos olhos do consumidor final, pouco valor por não adicionar muito ao serviço, devido a este ter a maioria do seu valor na conveniência e não na segurança. Também com a mesma tecnologia, a referência [23] criou um estudo na relevância para entregas duplicadas neste tipo de serviço e chegou-se à conclusão de que nem o custo-benefício de integração de tal tecnóloga teria sido benéfico para as empresas e *respetivos* clientes. Em teoria poderia também ser útil no caso de utilização de *smart contracts* que esta tecnologia de *blockchain* oferece, segundo o artigo [24] que menciona a utilização para regulamento de mercado livre entre cidadãos comuns com a mediação por parte de um terceiro que existe confiança por ambas as partes.

Resumo de Abordagens a *Smart Lockers*		
Tipo de tecnologia	Vantagens	Desvantagens
Cacifos Fixos	- Custo reduzido para poucos cacifos - Baixa complexidade - Pegada fixa	- Expansibilidade reduzida - dispendioso para grande quantidade de cacifos - Módulos de controlo possivelmente desnecessários
Cacifos com Torres Modulares	- Possibilidade de expansão - Associação de cacifos facilitada - Um módulo de controlo para todos os cacifos associados	- Associação de cacifos com número fixo - Ter em conta localização para expansão - Sem aumento da capacidade é mais dispendioso que cacifos fíxos - Aquisição de cacifos ao mesmo fabricante
Cacifos Modulares	- Possibilidade de expansão - Associação de cacifos por pilhas - Um módulo de controlo para todos os cacifos associados - Associação de cacifos consoante necessidade	- Ter em conta localização para expansão - Sem aumento da capacidade é mais dispendioso que cacifos fíxos ou torres modulares - Aquisição de cacifos ao mesmo fabricante
π-Boxes as Mobile Modular Lockers	- Possibilidade de expansão - Associação de cacifos sem necessidade do tamanho ser uniforme - Um módulo de controlo para todos os cacifos associados - Associação de cacifos consoante necessidade - Aquisição de cacifos a diferentes fabricante	- Ter em conta localização para expansão - Sem aumento da capacidade é mais dispendioso que outras soluções - Maior complexidade de manutenção em caso de erro

Tabela 2.1: Tabela de resumo de tipos de tecnologias utilizadas para implementação de *smart lockers*.

2.3 Projetos com implementação da tecnologia

Já existem várias soluções em vigor nesta área de logística, sendo que em Portugal, esta tecnologia, ainda não foi adotado de forma significativa. Assim sendo e com o aumento do comércio online e a comodidade que será oferecida ao cliente, a adesão a este tipo de tecnologia é apenas uma questão de tempo. Por aquilo que se pode verificar do estado de arte, qualquer sociedade vê os benefícios desta solução aquando apresentada a sua potencialidade e quando é adotada, mesmo que seja um serviço que adiciona custos ao preço de uma entrega.
Projetos encontrados em pesquisa para análise atual revelam que já existe muita diversidade, desde montagem por módulos, cacifos fixos, tipos de chaves para conceder acesso a um utilizador, etc. Empresas mais convencionais como a *FNAC* já implementaram protótipos para utilização por parte dos consumidores. Isto como uma melhoria do serviço prestado e com resultados que propõe escalar o serviço. Para referência desta informação, temos a apresentação de serviços de 2019 da *FNAC*, referenciado em [25] e a solução na figura 2.1.
Uma das primeiras aplicações desta tecnologia com sucesso de forma pública foi feita pelo

Figura 2.1: Solução de cacifos inteligentes pela *FNAC*.

grupo empresarial *CTT* que tem como principal negócio a logística de correios de Portugal. Esta inovação criou 267 pontos de recolha com *smart lockers* de norte a sul do país. O litoral e os centros urbanos de Lisboa e Porto possuem maior concentração destes cacifos. Os CTT já têm parcerias de colaboração com outras empresas, como a *Decathlon*, *Nespresso* e *Springfield*, a título de exemplo, pela inclusão de levantamento dos seus produtos em compras online estarem diretamente ligados a estes serviços. Para visualização desta informação, consultar o *website* dos *CTT*, na referência [26] e um exemplo da solução na figura 2.2.

Figura 2.2: Solução de cacifos inteligentes pela *CTT*.

Uma outra empresa portuguesa *Lokk*, referenciada em [27] e o produto representado pela figura 2.3, também está a explorar este mercado, mas neste caso opta pelo aluguer ou venda dos seus cacifos. Estes cacifos têm como objetivo estar dentro de uma empresa, ginásio, estabelecimento de ensino sendo a sua finalidade para uso exclusivo de clientes ou colaboradores destes locais. A interação com os cacifos é feita através de dispositivos móveis e outros métodos sem contacto direto do utilizador.

Numa perspetiva mais internacional já existem projetos por parte de grandes empresas de logística como a *UPS* para melhoria dos seus serviços e eficiência das suas entregas sendo que ainda não foi implementado em Portugal. Oficialmente em uso, mas indisponível nesta

Figura 2.3: Solução de cacifos inteligentes pela *Lokk*.

região, a situação é mais um estudo de aplicabilidade do que um produto totalmente realizado e eventual aplicação com a evolução de mercado. Esta está referida em [28] para referência a demonstração de produto, sendo estes cacifos representados na figura 2.4.

Figura 2.4: Solução de cacifos inteligentes pela *UPS*.

À semelhança do caso da *UPS*, a *Amazon* já tem implementada internacionalmente uma solução de *smart lockers* que chamou *Amazon Locker*. Estes já estão disponíveis em vários países europeus incluindo a Espanha, França, Alemanha, etc. Este serviço também está disponibilizado para utilização por outras empresas ou gestão de encomendas pelo uso do conjunto das suas soluções, através do *Amazon Hub*. Este último modelo de negócio inclui dois tipos de cacifo, *Locker* e *Apartment Locker*, sendo estes correspondentes aos tipos de aplicação mencionados respetivamente. *Apartment Lockers* são mais indicados para gestores de propriedades como condomínios fechados e universidades. Para referência desta informação, consultar o *website* dos cacifos *Amazon*, referenciado em [29] e um exemplo da solução na figura 2.5.

Numa breve pesquisa verifica-se que a grande maioria dos produtos disponíveis se destina ao mesmo mercado que a *Lokk* está a explorar. As empresas *Ricoh*[30], *Locktec*[31], *Bradford Systems*[32], *Velocity Smart*[33] e *Meridian Kiosks*[34] são alguns exemplos deste tipo de empresas. Estas têm todas o mesmo modelo de negócio e a distinção entre estas é na sua grande maioria a disponibilidade do serviço e a interface de como é feita a entrega e receção da entrega a nível de plataforma.

Figura 2.5: Solução de cacifos inteligentes pela *Amazon*.

Algumas empresas optam por códigos PIN outras por códigos de barras, outras por códigos QR entre outros métodos, sendo o seu propósito: servir de chave única para os cacifos. Estas soluções variam em termos de segurança, sendo umas mais resilientes que outras.
Estas diferenças de implementação serem pequenas, revela uma grande maturação da tecnologia. Sendo que o principal fator de escolha entre as várias opções é, em maioria, o preço e a interação do cacifo e o utilizador, serviços complementares e não a funcionalidade em si.

2.3.1 Soluções Comerciais Disponíveis

2.3.1.1 Ricoh

A empresa *Ricoh* desenvolveu várias soluções, consoante cada aplicação a interação do utilizador com o cacifo difere. Isto deve-se ao ambiente em que estes se integram poder não ser viável ou fazer mais sentido para o tipo de negócio em que se aplica. Para ambientes de escritório, é oferecida a solução *Smart Day Lockers*, que são *smart lockers* para uso pessoal, sistemas em que é comum os colaboradores não estarem associados a um posto fixo, alternando entre vários *open-spaces* ou lugares nos mesmos e em consequência coisas como cacifos têm de ser flexíveis na sua associação ao colaborador. A interação é feita através de dispositivos móveis, via ecrã tátil ou mesmo por leitura de cartões. Para contexto de entregas e correio a empresa criou a solução *Smart Mail and Parcel Lockers*, que se destina à sua gestão interna, para que seja extinta a necessidade de um colaborador a tempo inteiro para tal. Isto faz com que avisos de receção e registo de recolha sejam automáticos. A interação é feita através de um ecrã tátil para inserção de um código ou por cartão de colaborador. Para ambientes em que seja necessário manter registo de inventário por ser equipamento especializado, por exemplo, foi desenvolvida a solução *Smart Inventory Lockers*, que têm as mesmas funcionalidades que a solução anterior, sendo a interação semelhante, apresentando este o benefício de manter totais em stock do equipamento que está associado a um diferente colaborador. Isto pode ser estendido a um simples controlo de uso de stock para melhoria de serviços internos. A interação é feita através de ecrãs táteis ou cartões de identidade na empresa. A última solução disponível é para a área de venda de retalho, sendo esta a solução *Smart Lockers for Retail*. Esta solução tem como objetivo disponibilizar ao negócio a possibilidade de levantamentos a qualquer hora, estando os cacifos preparados para serem instalados no exterior. A interação com o sistema é feita pela inserção de um código enviado

ao cliente por *short message service* (SMS), email ou algo similar. As soluções são não expansíveis, mas é possível utilizar uma destas versões de produto para a criação do número de cacifos desejados. A figura 2.6 ilustra um exemplo das soluções propostas por esta empresa.

Figura 2.6: Solução de cacifos inteligentes pela *Ricoh*.

2.3.1.2 Locktec

A *Locktec* tem muitas opções de produtos consoante a sua aplicação, mas esta diferença é apenas ao nível da capacidade do cacifo em si. Estas opções são coisas como resistência a impactos, refrigeração, sendo as soluções diferenciadas pelo tipo de tecnologia utilizada. A expansão do sistema é não modular, ou seja, não é possível adicionar novos cacifos a posteriori ao sistema, sendo necessária uma nova aquisição da totalidade do sistema se houver necessidade de ampliação. Também é possível contactar diretamente a empresa para customizar o número de cacifos. A interação com os cacifos é opcional entre eletrónico, *radio frequency identification* (RFID), impressão digital, código *personal identification number* (PIN) com ecrã tátil, ou leitor de código de barras. A figura 2.7 ilustra um exemplo das soluções propostas por esta empresa.

Figura 2.7: Solução de cacifos inteligentes pela *Locktec*.

2.3.1.3 Bradford Systems

As soluções providenciadas pela *Bradford Systems* são mais guiadas para a gestão de cacifos em ambientes como escritórios e universidades. Isto pela integração de IOT com os cacifos criando uma maneira alternativa para a sua gestão. Isto pode variar desde controlo de acessos, disponibilidade, mudança de chave, etc. As soluções têm várias tecnologias para a certificação do acesso, sendo opcional entre RFID, impressão digital, código PIN, comando de infravermelhos, *free on board* (FOB) e Bluetooth. Também tem soluções semelhantes às mencionadas anteriormente em que o controlo é feito por um módulo para acesso a vários cacifos. Esta solução é a resposta às necessidades de venda de retalho, dando o acesso através de um código PIN ao cacifo adequado. A figura 2.8 ilustra um exemplo das soluções propostas por esta empresa.

Figura 2.8: Solução de cacifos inteligentes pela *Bradford Systems*.

2.3.1.4 Velocity Smart

Soluções mais limitadas, em relação ao que já foi apresentado, são providenciadas pela *Velocity Smart*. Isto deve-se ao facto de serem soluções fechadas sem customização do número de cacifos que são inseridos na solução. Existem 3 tipos de soluções, séries 1000, 1800 e Element. A primeira serve apenas para uso em interiores e as outras duas também podem ser utilizadas no exterior. Em termos de interação, as duas primeiras utilizam códigos PIN e códigos de barra para acesso, a Element é programável e tem acesso a uma câmara para prova de acesso, mas pode ser também um input válido. A figura 2.9 ilustra a série 1000 proposta por esta empresa.

2.3.1.5 Meridian Kiosks

A solução mais inovadora é proveniente da *Meridian Kiosks*, sendo esta uma solução de torres modulares. Os módulos oferecidos são diferenciáveis pelo tamanho dos cacifos de cada um, apresentando 4 tamanhos possíveis. Assim esta será a solução mais indicada para expansão no futuro. Apesar disto será limitado aos tamanhos existentes, e sendo que cada torre só tem um tamanho, não existindo misturas de tamanhos por módulo, tem de ser planeada a expansão e compra consoante a necessidade. Esta solução está direcionada a ser um ponto de recolha de mercadorias, tendo assim o propósito de gestão de correio e encomendas pelo

Figura 2.9: Solução de cacifos inteligentes pela *Velocity Smart*.

cliente. Apesar de este ser o principal caso de uso, a solução pode também ser utilizada em ambiente de escritório e uso pessoal, sendo retiradas muitas das capacidades do sistema que não são úteis para estes ambientes. Ou seja, a solução também é adaptável ao fim que é pretendido. Para interação com o sistema, são utilizados ecrã táctil, câmara para leitura de códigos QR, leitor de códigos de barras e possivelmente um simples painel numérico para inserção de códigos PIN, para levantamento rápido de mercadorias. A figura 2.10 ilustra uma possivél configuração deste produto.

Figura 2.10: Solução de cacifos inteligentes pela *Meridian Kiosks*.

2.4 Conclusões

Com a breve análise do estado da tecnologia, é percetível que existe já muita informação que facilita as perspetivas de implementação, com o complemento de análise de expectativas do público em relação a esta tecnologia. A tecnologia tem um grande nível de maturação devido à evolução do mercado de comércio *online*, sendo este uma grande contribuição para o aumento do número de encomendas que têm de ser tratadas por empresas de logística, o que leva a uma pesquisa contínua de melhoria das soluções destas para os problemas que vão sendo criados. Alguns destes problemas são:

- Limitações horárias na entrega, comprometendo recursos da empresa e em especial do cliente, sendo difícil a coordenação em horários de trabalho e de disponibilidade.
- Tentativa de menor alocação de trabalhadores e recursos como veículos para melhoria de resultados económicos para os investidores.
- Otimização de tempos de entrega desde comunicações com o cliente para acordo de horários de entrega, a entrega em si e a burocracia associada para registo da entrega.
- Gastos agravados por congestionamento de vias de trânsito, em que veículos que fazem as entregas em horários de ponta, também contribuem para o aumento de gases de efeito de estufa.
- Fraude ou roubo de mercadorias por entregas serem deixadas sem vigia em ambientes públicos.

Os benefícios de resolução deste tipo de problemas e outros menos relevantes, como melhoria de disponibilidade de acesso do utilizador a bens 24 horas por dia no caso de *smart lockers* para venda de produtos sob demanda, a partilha de bens por indivíduos através do uso de um meio que controla acesso sem necessidade de vigia, aumento da qualidade do serviço nas grandes áreas importantes do negócio como fiabilidade e privacidade, e em especial, a conveniência para todas as partes envolvidas. Além disto, existem subprodutos da implementação de *smart lockers* como o benefício de não contacto, para pessoas com doença auto-imune, ou em situações em que contacto entre indivíduos pode levar a problemas de saúde pública, e também o facto de manter informação privada como a morada do indivíduo.
A potencialidade de escalamento de soluções apenas por uso de IOT também revela o potencial de que *smart lockers* sejam uma solução a longo prazo. A distribuição de recursos irá evoluir, mas a última fase de entrega de um produto ao indivíduo será difícil de superar os benefícios oferecidos por este tipo de serviço, especialmente em termos de manutenção, consumo de recursos e disponibilidade.
Esta tecnologia também pode revolucionar áreas como a indústria farmacêutica, adicionando a tecnologia a uma simples máquina automática de vendas, criando a hipótese de farmácias estarem disponíveis 24 horas por dia, por exemplo, para levantamento de receitas com prescrição. Isto aumenta a rentabilidade destes estabelecimentos e a rapidez do processo de transação e pagamento, dependendo da implementação. Este método é especialmente útil quando a pessoa tem medicação regular.

A integração de novas tecnologias como *block chain* para garantir o registo de entregas e outras aplicações, podem melhorar em muito a solução. Estas pequenas inovações podem apresentar-se muito relevantes para questões de administração. Outras tecnologias ainda indisponíveis decorrentes de novos estudos que integrarem a solução serão sempre uma mais valia, tanto em áreas de redes, interação pessoa máquina, serviço ao utilizador, etc.

Capítulo 3

Engenharia de Software

3.1 Introdução

Este capítulo destina-se ao planeamento do trabalho idealizado no período do estágio proposto pela Altice. Tendo os casos de uso de software que foram propostos como casos de utilização destes *smart lockers*, sendo a parte de uso para clientes a prioridade de realização do projeto.

3.2 Casos de Uso

Como já foi mencionado neste relatório as intenções de implementação do projeto deverão ter várias aplicações para a empresa. O caso que será a prioridade deste estágio é a interação cliente empresa, para facilidade de recuperação de serviço por parte dos clientes, caso existam danos de dispositivos fornecidos, por exemplo. Esta oferta de serviço possui várias motivações, entre as quais, danos provocados por picos de corrente elétrica em casos de mau tempo inutilizando os produtos de *hardware*, ou falhas de *hardware* que levam a que estes se tornem inutilizáveis por desgaste.
Assim o primeiro caso de uso e prioridade do que será realizado para este estágio, será a utilização destes cacifos para recolha de novos dispositivos como *routers*, *box* televisivo, cabos de alimentação, RJ45, etc. Assim, pretende-se que a empresa possa oferecer um novo tipo de serviço que estará disponível 24 horas por dia, sem contacto com outros indivíduos e com a conveniência de potencialmente estar mais perto do cliente do que um dos postos que estão disponíveis hoje em dia, que tipicamente, estão em centros e áreas comerciais. Estes postos podem ser distantes especialmente para zonas rurais que muito beneficiarão com a implementação proposta, mas existe a possibilidade de ser benéfico, também, em grandes cidades em que existem áreas designadas de habitação e comércio, implicando uma deslocação para um destes centros. Mais uma vez, ocupando estruturas de transporte desnecessariamente.
Em situações de fragilidade de saúde ou imunodeficiência do indivíduo, a solução evita que exista proximidade entre indivíduos, minimizando contacto e em consequência a possível transmissão de doenças. Isto pode ser um grande benefício no que toca à saúde destas pessoas que, por vezes, combater o mais simples vírus ou bactéria se torna muito difícil. A acessibilidade a produtos seria então melhorada, muito devido à segurança que seria criada por esta não interação. Em situação de epidemia e pandemia será ainda mais relevante este não contacto, por estes poderem ser um ponto neutro de recolha com a redução de possibilidade de transmissão no caso de um bom controlo higiénico por parte de quem utiliza os cacifos.
Para ilustrar os casos de uso, as figuras 3.6 e 3.2, ilustram os diagramas de caso de uso para clientes e *call center* da empresa. Este pode estar incompleto pelo facto deste relatório ser

realizado antes de qualquer trabalho em estágio.
O diagrama de caso de uso 3.1 ilustra o cliente no requisitar de *hardware*, sendo que isto poderá ser executado pela aplicação que será desenvolvida ou por telefone e registado no sistema pelo *call center*, no caso do mesmo não conseguir ter acesso a esta. Os restantes usos são os processos de levantamento de produto dos *smart lockers*, isto será, como ilustrado, realizado ou pela aplicação sendo mostrado o código para abrir o cacifo na mesma, ou este ser enviado por SMS ou email para o cliente. Chegando assim ao levantamento que inclui todo o processo de registo de uma transação concluída como o fecho do cacifo.

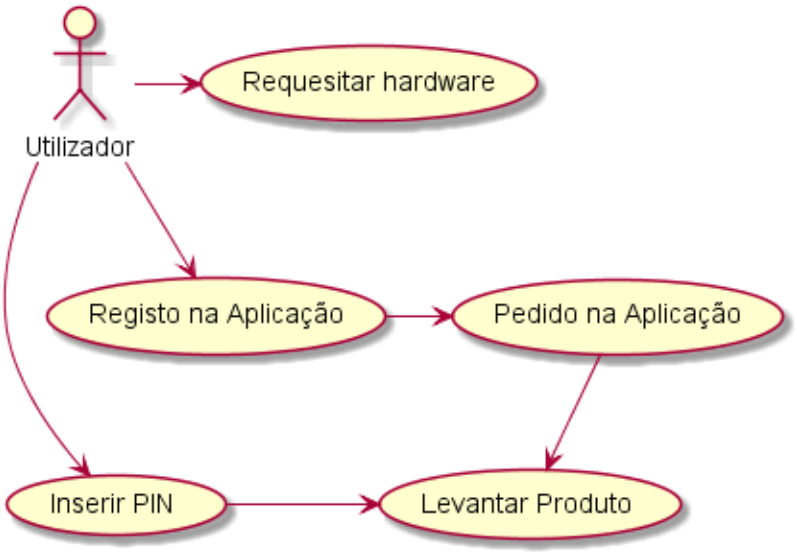

Figura 3.1: Diagrama de casos de uso de fluxo de ações do utilizador.

A figura 3.2 é um caso de uso para especificação de como será o processo de registo interno para clientes que tomam a opção de ligar à empresa em caso de ocorrência de mudança de contrato que envolve mudança de *hardware*, ou em caso de avarias. Estas interações são uma boa maneira de facilitar o utilizador e também criar um meio que permite utilizadores que não têm a possibilidade de utilizar a aplicação.

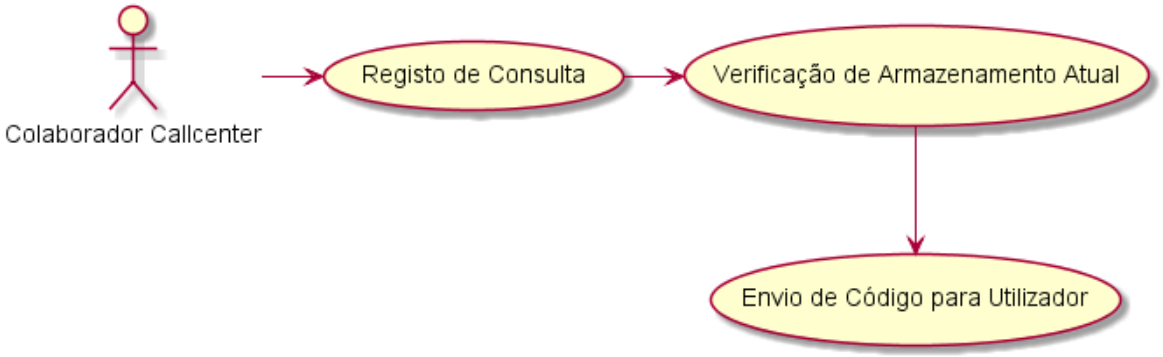

Figura 3.2: Diagrama de casos de uso de fluxo de ações por parte da empresa para atribuição de *Hardware*.

Para completar o que é comunicado pelos casos de uso, as figuras 3.3, 3.4 e 3.5 , incluem diagramas de atividades. Estes são uma das muitas maneiras como se poderá desenvolver o sistema a um nível básico, sendo esta então a base que guia o desenvolvimento inicial.

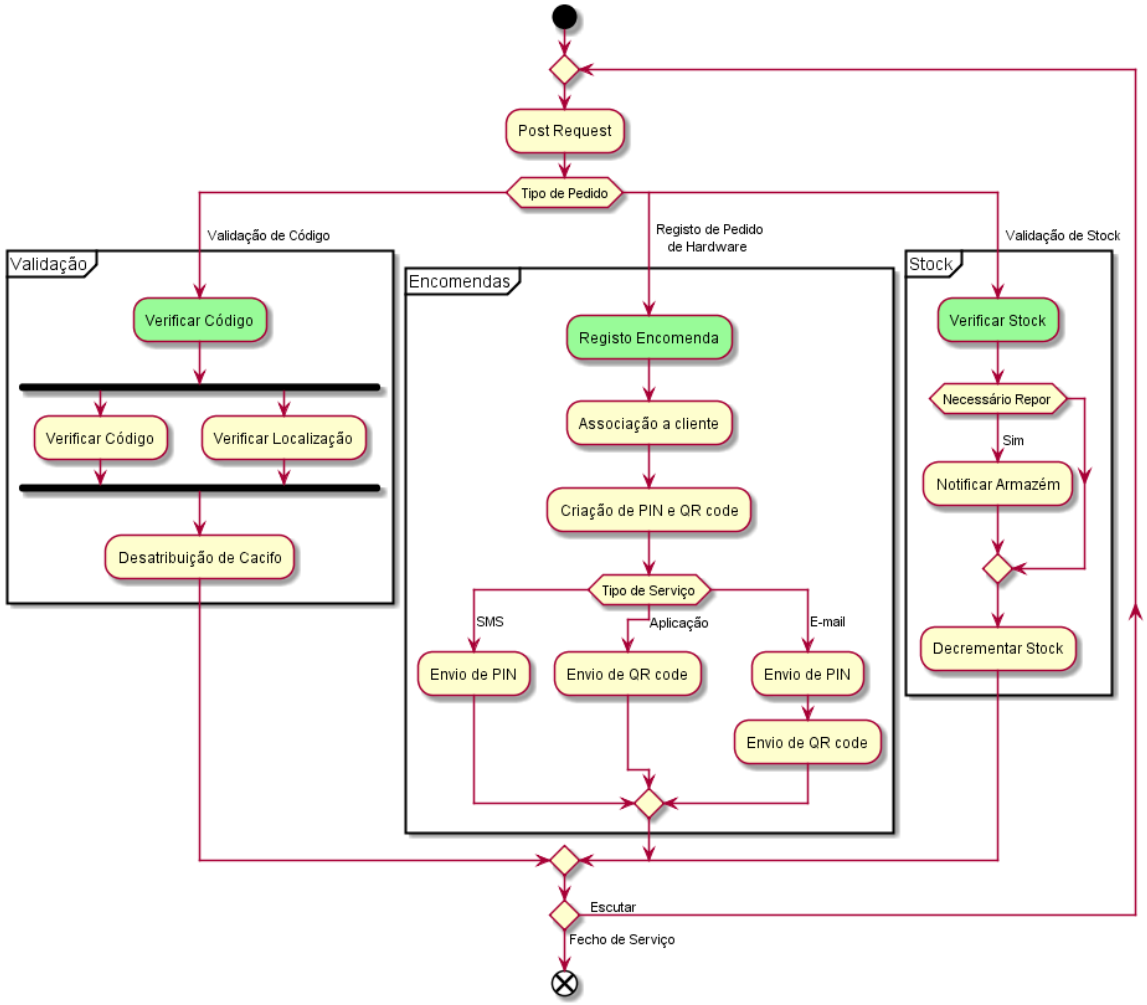

Figura 3.3: Diagrama de atividades de backend.

Em explicação da figura 3.3, esta contém o diagrama de atividades que descreve a *application programming interface* (API) que comunica com a base de dados e os *endpoints* que esta inclui. Este serviço terá de estar disponível 24 horas por dia de forma a manter o serviço dos cacifos. Como se pode observar pelo menos 3 *endpoints* serão incluídos:

- A validação de códigos para acesso a cacifos;
- O registo do *hardware* que é para ser levantado e lógica para associação a um cliente;
- Verificação de *stock* existente numa dada localização para fins administrativos.

A figura 3.4, contém o diagrama de atividades que aborda o pedido de *hardware* do cliente. Este terá de comunicar com o *backend* para algumas das suas funções e terminar o seu registo através do envio das informações de levantamento para o cliente.
O diagrama de atividades, contido na figura 3.5, expõe como o *smart locker* irá proceder no levantamento de produtos. Assim, este começa por uma aquisição do código de acesso, por um dos métodos oferecidos. Neste caso demonstra-se com a possibilidade de PIN ou QR *code*. O código é então processado pelo *backend* e o cacifo correspondente ao pretendido pelo cliente é aberto e registada a receção por parte do cliente, de seguida é fechado o cacifo.

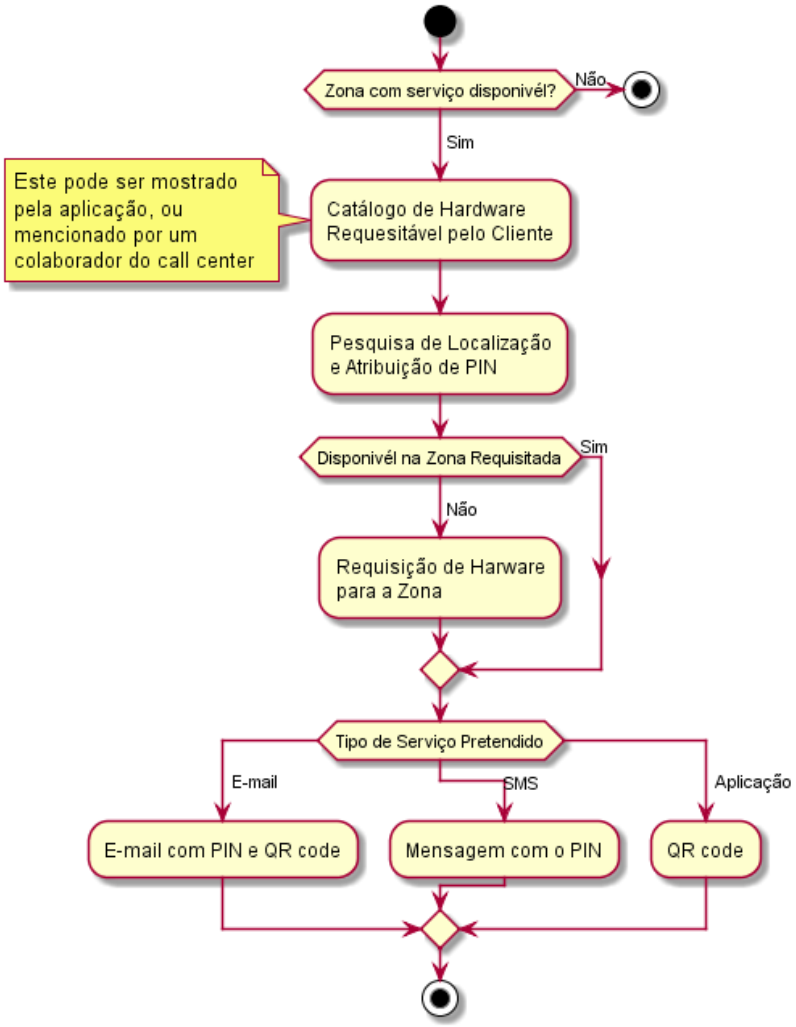

Figura 3.4: Diagrama de atividades de pedidos do cliente para *hardware*.

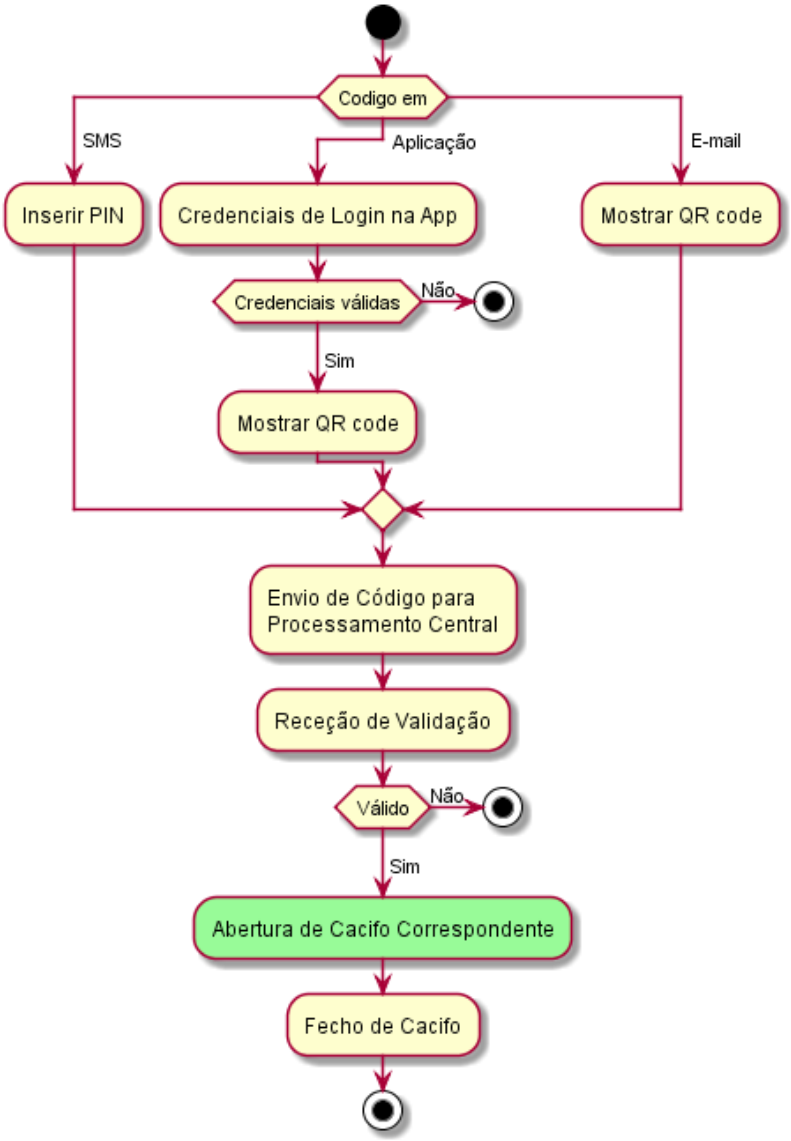

Figura 3.5: Diagrama de atividades de levantamento de *hardware*.

3.3 Integração

Como será de esperar, as soluções criadas pelo aluno têm de ser integradas com as soluções já existentes na empresa. Tendo isto em conta existe a necessidade de comunicação com várias áreas de negócio da empresa para acesso de dados dos clientes para aprovação dos serviços prestados.

Com isto será necessário criar integração com as bases de dados existentes pela empresa. Este será um constrangimento que poderá criar entraves no progresso do projeto devido a projetos herdados para integrar com a solução pretendida.

Outro constrangimento é a disponibilização destes dados. Sendo estes, dados pessoais de clientes da Altice, poderá ser um conflito de interesse para o desenvolvimento destes cacifos, por parte das entidades envolvidas, empresa e cliente. Na integração e aceitação do produto, no caso dos clientes, e disponibilização dos dados para uso pelo aluno, no caso da empresa.

Desta forma são ilustrados alguns possíveis problemas de integração da tecnologia que podem ocorrer na duração do período de estágio. Estes desafios terão de ser respeitados por parte do aluno e como tal os resultados do estágio podem ser diferentes dos projetados para os casos de aplicação deste sistema de *smart lockers* e adaptados à melhor solução encontrada em conjunto com a empresa. Uma possível resolução para o caso dos dados uma resolução poderá passar pela utilização de dados fictícios com a mesma estrutura para aplicação da tecnologia do futuro, isto porque não seria necessário modificar a lógica intrínseca do programa.

3.4 Conjunto de soluções técnicas

Para tornar possível esta tecnologia ser integrada de forma simples na vida do cliente e, possivelmente na de colaboradores da empresa, várias soluções distintas deverão ser criadas para este fim. Após alguma interação com o orientador designado pela empresa chegou-se ao entendimento que o projeto terá de envolver várias tecnologias e envolver vários ramos de informática.

Primeiramente será necessário criar a parte eletrónica do dispositivo a que chamamos de *smart locker*. Não tendo ainda uma informação concreta daquilo que é fornecido por colaboradores da Altice em termos de *hardware* assume-se que seja de raiz excetuando o cacifo em si, que será a plataforma, em que será feita esta componente do estágio curricular. Aquilo que é garantido é que serão utilizadas soluções de *hardware* de baixo custo e de baixo rendimento, como Arduino e adições a esta plataforma com a possibilidade de utilização de circuitos integrados para prototipagem do produto, para prova de conceito. Esta solução deverá ser integrada em antigas cabines telefónicas para a sua substituição e assim aproveitamento do espaço existente que estas oferecem. Isto deve-se ao facto de estas possuírem conexões à rede da empresa, devido à comunicação requerida pelas linhas telefónicas, eletricidade para a alimentação do sistema e o espaço, já está dinamizado e autorizado a ser utilizado pela empresa por todo o país.

Em segundo lugar existe a necessidade da comunicação com os dados dos clientes. Sem esta informação será impossível saber se um indivíduo é ou não cliente da empresa. Dado que isto é um requisito para fornecimento de *hardware* para os serviços prestados pela empresa, sendo inclusive necessário saber se a pessoa deve ter acesso a certo tipo de *hardware*. Um exemplo seria um indivíduo que tem contratado à empresa serviços televisivos, mas, neste caso, tenta levantar um *router*. Como é possível entender com este exemplo, esta interação é não trivial, exigindo alguma lógica de acesso, consoante o tipo de serviços contratados. Outro dado a registar será o que foi fornecido ao cliente, isto pela necessidade de reaver o *hardware* no caso de cessação de um contrato. Com este sistema será possível monitorizar com quem está o *hardware* e que *hardware* está por ser reavido de forma a evitar a perda destes produtos e manter o *stock* de equipamentos atualizado.

Por fim existe a necessidade de criar uma forma de interação com o sistema por parte de um indivíduo. Assim, para tornar o sistema o mais ubíquo possível teorizou-se que seria benéfico ter uma aplicação móvel para esta interação. Dito isto terá de ser desenhada a interação,

desde autenticação à função de interação do sistema consoante o contrato que está associado à pessoa. A aplicação terá de comunicar com o sistema referido anteriormente a fim de inserir e referir intenções do cliente para a recolha de um produto.
No caso de não ser possível utilizar esta tecnologia de aplicação móvel, seria de valor criar uma integração com o sistema telefónico. Este sistema enviaria uma mensagem para o cliente com uma palavra-passe a fim de fornecer o acesso aos produtos no cacifo e além disso serve como registo e prova de levantamento de um dado produto. Esta solução pode ser mais dispendiosa para a empresa, mas seria benéfico para clientes que não possuam um *smartphone*, ou não tenham capacidade para o fazer.

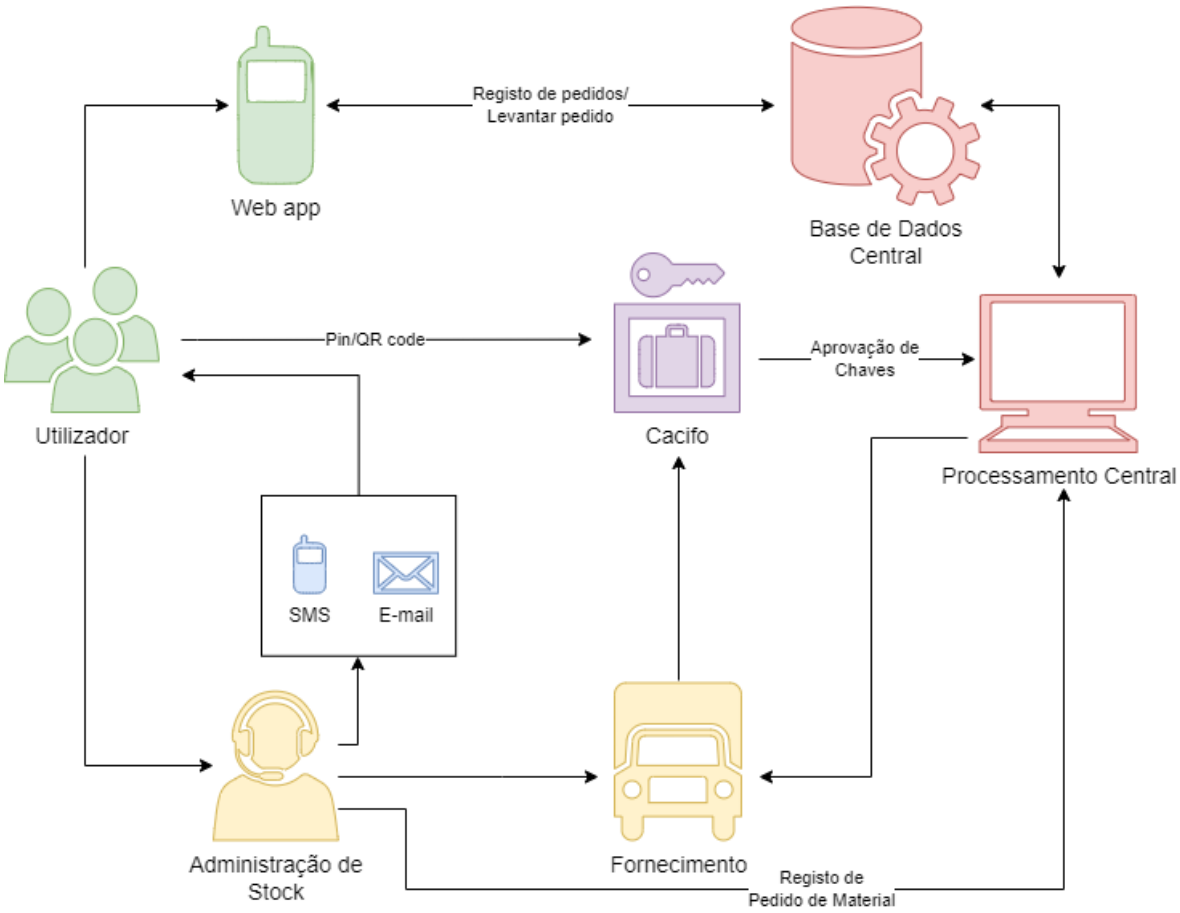

Figura 3.6: Diagrama de funcionamento a alto nível de arquitetura do projeto final.

3.4.1 Integração com *backend* existente

Para a criação da solução pretendida no final do estágio com a Altice terão de ser integrados alguns dos serviços que estão em produção pela empresa atualmente. Por motivos de continuidade e manutenção do produto, num futuro em que a solução desenvolvida neste estágio se toma como totalmente viável pela empresa e se avança com o projeto como uma área de investimento, a integração com serviços existentes é uma prioridade.
Assim, para integrar estes serviços e tornar possível que os serviços que são acrescentados pelo projeto de *smart locker* se tornem profícuos foi mencionado que esta integração tem como requisito a utilização de PHP sendo que C# também seria uma possibilidade. Isto deve-

se ao facto de que a maioria da capacidade técnica da empresa está mais familiarizada com PHP e a minoria C#. Estas tecnologias são testadas já há algum tempo, tendo assim um maior suporte pela comunidade e tempo de maturar os seus erros e falhas. PHP é maioritariamente utilizado para desenvolvimento web, principalmente a interação de sistemas pela *internet*. Ou seja, comunicação de informação entre máquinas. C# é uma linguagem mais generalizada sendo utilizada em todo o tipo de aplicações.

3.4.2 Aplicação para *Smartphone*

Para que os clientes tenham a melhor experiência possível com a potencial comercialização desta tecnologia temos de tornar o seu acesso o mais ubíquo possível e o mais acessível para quem não possui tanta experiência com tecnologia. Isto deve-se ao facto de a empresa ter todo o tipo de bases tecnológicas da pessoa que contrata os seus serviços.
Em discussão com o responsável pelo projeto na empresa foi mencionada a criação de uma *web app* para eventualmente ser integrada numa aplicação para *smartphones*. Assim, tendo um ponto de interação com o sistema como o telefone, prevê-se uma maior taxa de adesão a estes serviços. Terá de ser visto o âmbito do projeto para a exequibilidade de uma aplicação híbrida para não ser apenas um *website* como ponto de acesso. Outra opção seria a criação de uma aplicação nativa, mas os recursos necessários para tal são mais exigentes e foi mencionado que este em princípio não será o objetivo para o estágio.

Capítulo 4

Planeamento de estágio

4.1 Introdução

Neste capítulo é feita uma breve explicação do que será realizado ao longo do estágio e como será organizada, em princípio, a ordem de tarefas. Assim serve como um guia para a fase de execução do projeto e uma projeção para análise nesta fase intermédia do planeamento de estágio do mestrado.

4.2 Descrição de Planeamento de Estágio por Tarefas

Como já mencionado, será necessário criar pelo menos 3 soluções e talvez seja necessário subdividir ou integrar mais projetos para um sistema completo. Estas 3 soluções são: a criação da solução de hardware, criação de comunicação de dados para operação do hardware e finalmente a *web app* para uso do cliente em *smartphones* ou então a utilização de código enviado por SMS para um telefone, ou mesmo até as duas soluções dependendo de requisitos e tempo para implementação de soluções.
Assim como podemos ver, existe um tema central que deverá ser tratado com prioridade: a comunicação de dados. Assim a comunicação com a base de dados deverá ser a primeira prioridade do estágio. Isto envolve o registo de pedidos de hardware, a criação de um código de acesso para um acesso que pode ser na forma de carateres para inserir num teclado físico ou tátil, código de barras ou código *quick response* (QR) ou de resposta rápida em português. Pesquisa no stock existente de cada produto, ou seja, que produtos estarão em cada ponto de cacifos. A criação de correlação por área ou concelho para evitar a deslocação de colaboradores a áreas que não possuem stock de um dado produto.
Depois disto será possível completar a fase de criação do hardware e o software que o suporta, a fim de que este tenha comunicação com os dados referidos acima. Para o funcionamento destes cacifos será de notar o seguinte, a chave tem de ser anulada sempre que é utilizada, para evitar o retirar de mais que um produto indevidamente, por exemplo vários *routers* quando é suposto este ter apenas direito a um pelo contrato. Assim esta dinâmica será tratada a nível do servidor que detém a base de dados que autentica o acesso uma única vez por chave. Com a aprovação feita será necessário a abertura do cacifo e assim concluído pelo fecho deste. Com esta explicação evidencia-se a necessidade de fecho de comunicação da base de dados.
Quando estas duas tarefas estiverem concluídas o último passo é a criação da *web app* que facilita a comunicação do utilizador com o cacifo. Esta é a fase menos crítica do desenvolvimento pela comunicação a implementar serão só chamadas ao *backend* referido da primeira fase, simplificando esta. O principal foco desta fase de desenvolvimento será tornar o serviço o mais acessível e intuitivo possível. Para tal, se houver a possibilidade, a introdução

de utilizadores finais para testes alfa seria desejável para completar o *feedback* deste. Caso esta fase também possibilite, será de notar que a implantação de envio de códigos por SMS também poderá ser de valor.

A duração de cada uma das tarefas terá a previsão de 1 mês, o que deixa um mês alocado para testes e análise de bugs que possam existir. Neste último mês também será possível, no caso de criação de atrasos compensar algum imprevisto. Para o desenvolvimento do projeto também será necessária a aprendizagem de PHP e de políticas de implementação da empresa, podendo estes fatores influenciar as previsões apresentadas e alongar o tempo necessário para a primeira fase de implementação do projeto. Este planeamento está referenciado na tabela 4.1.

Planeamento de Desenvolvimento de Estágio				
Tarefas	Fevereiro	Março	Abril	Maio
Onboarding na empresa	X			
Interação com Backend existente	X	X		
Plataforma de Hardware		X		
Web app Para interação de Cliente			X	
Testagem e integração				X

Tabela 4.1: Tabela de projeção de tarefas do projeto a executar em cada mês, assinalado com um X o projeto a executar.

Capítulo 5

Desenvolvimento e Implementação

5.1 Introdução

Este capítulo faz a representação da arquitetura do sistema final do estágio. Inclui o funcionamento do sistema e como este está estruturado. A descrição de funcionamento também refere a execução do código de cada um dos componentes que integram o sistema.
Isto é feito através de uma breve apresentação de forma geral descrevendo a estrutura e interações entre sistemas e, de seguida, uma narração mais detalhada de cada um dos tópicos. Isto inclui a interface do utilizador, *Web App*, a base de dados, as API's que controlam o sistema e finalmente os micro serviços.

5.2 Visão Geral do Sistema

O sistema é composto por uma *Web App*, que é o ponto de interação não administrativa, esta interage com duas APIs que fazem a interação entre o processamento e a base de dados do sistema, finalmente o restante sistema é composto por 6 micro serviços em que 5 são de processamento de dados e um para o registo de erros, *logging*.
A *Web App* tem vários modos de operação consoante o grau de autorização do sistema. Por exemplo, o sistema dá ao cliente acesso aos seus dados pessoais, localizações possíveis e que tipos de equipamento pode levantar num dado ponto, mas um colaborador pode consultar o registo de encomendas para uma dada localização e consultar requisições de equipamentos de utilizadores do produto. Apesar de dar mais acessos continuam a existir vários dados que não são acessíveis a não ser por criação de registos na base de dados diretamente, por exemplo que papel administrativo uma pessoa está associada no sistema, sendo que ao registar-se tem apenas nível de cliente e este terá de ser alterado manualmente na base de dados. Isto aumenta a segurança dos dados por não existir qualquer ponto de entrada, inclusive da API que interage com a base de dados, apesar de existir alguma sobrecarga sobre quem for administrador dos dados do sistema. Em princípio será algo que não é um problema porque adições ou mudanças de papéis na base de dados não serão comuns a não ser que existam alterações grandes de quem está incumbido de interagir com o sistema na Altice.
As duas APIs são os dois pontos que devem ser síncronos no sistema. A API de base de dados tem pontos de acesso a uma base de dados relacional baseada na tecnologia de SQL da Microsoft. Existem 9 tabelas nesta, mas cada uma tem vários pontos de acesso para diferenciar pelo menos as operações de criar e consultar um registo, listar registos consoante uma restrição. Algumas destas tabelas, como mencionado, não possuem acessos como a tabela de "*Role*". A API "*Orchestrator*" que lança o processamento por parte dos micro serviços do sistema em dois grupos, o grupo de criação de pedido e o grupo de autorização de acesso.

Esta última API também comunica com a API de base de dados para utilização de registos para o processamento dos micro serviços de um dos grupos, sendo estes registados na tabela "*Verification*" e "*Assign*". Outras tabelas como "*Stock*" e "*StockRequest*" também são afetadas por este processamento, mas de forma indireta, são apenas o resultado da utilização do sistema de produtos disponíveis.

Os grupos de micro serviços mencionados, "*Assign*" e "*Verification*" são a parte do sistema que faz todo o processamento dos dados do sistema. Estes grupos de micro serviços são compostos respetivamente de 3 e 2. O grupo "*Assign*" como o nome indica é o ramo de processamento relativo ao assignar ao cliente de um produto por meio de um pedido, sendo que este trata de assegurar e criar um PIN para a abertura de um cacifo. O grupo de "*Verification*" é o grupo de micro serviços que faz a aprovação do PIN e outras funções como a modificação do número de produtos de cada tipo que o cliente tem na sua posse.

Para melhor ilustrar o serviço a figura 5.1 pode ser utilizada como referência destes sistemas.

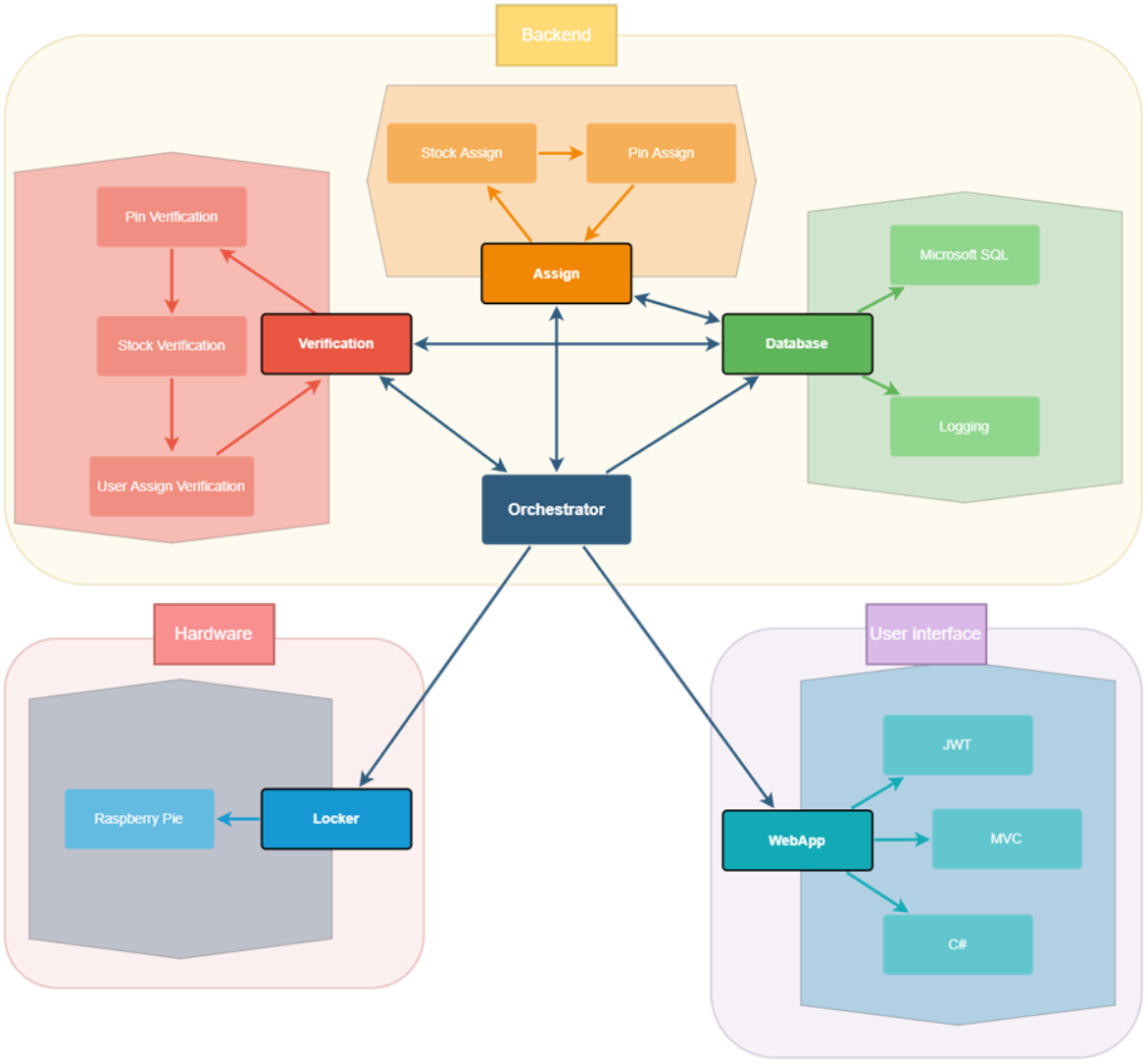

Figura 5.1: Diagrama a alto nível de arquitetura do projeto final.

Para suportar a comunicação entre os vários componentes do sistema, três bibliotecas foram criadas. Estas limitam a quantidade de código duplicado, fazem o sistema ser mais fácil de

manter e tendem a fazer com que o processo de implementação de cada micro serviço seja mais rápido por apenas ter de ser feita uma configuração de cada uma destas.

Uma destas é referente à comunicação com a Base de Dados (BD) que tem acessos a todos os *endpoints* que estão implementados na API de Base de Dados para que limite código duplicado. Sempre que é mencionado comunicação com a base de dados esta biblioteca é sempre utilizada. Esta é uma implementação de um cliente HTTP que modifica o caminho dinamicamente consoante o método chamado.

Outra biblioteca é referente à comunicação de micro serviços e também entre estes e o "*Orchestrator*". Esta abstrai a configuração da comunicação dos micro serviços e o servidor da *Queue*. A única configuração necessária pela utilização da biblioteca é o fornecer do nome da *Queue*, o IP do servidor e utilizador e password para autenticação das mensagens. Para facilitar a configuração foram implementadas duas formas de ler a *Queue*, de forma ativa ou por eventos. A forma ativa pode ser utilizada para ler a fila num dado momento caso seja necessário no futuro e por eventos é a forma que será utilizada para melhorar a performance do servidor. Sempre que uma mensagem está disponível numa *Queue* um evento é enviado e é feito um processamento da mensagem.

A última biblioteca que foi criada foi para a manipulação das mensagens que são enviadas entre serviços. Isto inclui a estrutura de uma mensagem, conversões de texto para objeto e vice-versa para facilitar a conversão das mensagens para serem enviadas ou para serem interpretadas pelo código, assinatura da mensagem, etc.

Estas mensagens têm uma estrutura semelhante ao que é utilizado na realidade por serviços de correios, com algumas liberdades para ajuste ao sistema. Estas são o endereço, que tem os nomes de remetente e identificador universal da mensagem, chamado *globally unique identifier* (GUID) no contexto de programação, campo de tempo de criação da mensagem como identificadores e como informação contida, os campos de história e processamento, *History* e *Processing* respetivamente. Esta organização cria claramente uma espécie de cabeçalho e um corpo de processamento de informação. A figura 5.2 ilustra esta informação.

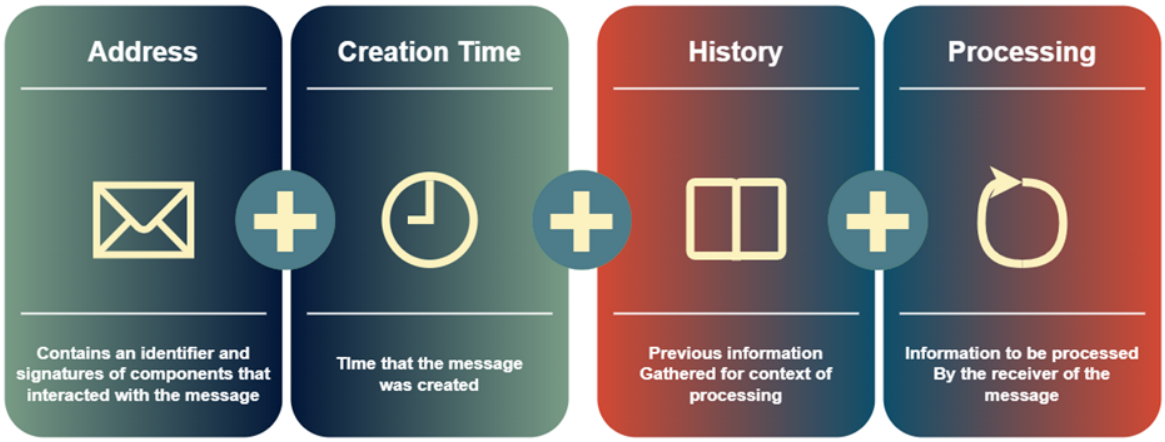

Figura 5.2: Diagrama a alto nível de organização de uma mensagem entre Queues do sistema de micro serviços.

5.3 Interface de utilizador

Para tentar preservar a solução de como interagir com o produto por parte do utilizador, uma *Web App* foi sugerida tanto por parte da Altice como do professor orientador. Assim esta tornou-se um objetivo claro a manter. Esta tem como objetivo:

- Utilizador
  - Pedidos
    - Criar pedidos
    - Visualizar pedidos
  - Visualizar *stock* existente
  - Locais com Cacifos
  - Perfil
- Colaborador
  - Pesquisa de utilizador
  - Visualização de *stock*
  - Visualização de pedidos de *stock*

As tecnologias que estão envolvidas são C#, *framework* de *.Net 6*, para a criação da API que está de acordo com a arquitetura de MVC. Existem pequenas diferenças na estrutura convencional desta arquitetura estas são derivadas de uma abordagem de manter as classes curtas e de fácil interpretação e manutenção. Assim tudo o que será processamento que seria feito em *controllers* está numa biblioteca de classes que é chamada de *business* e esta mantém todo o processamento de dados que vêm da camada de dados. Esta camada como indica o nome é a camada que faz chamadas à API de dados consoante o que é necessário por parte da página que o utilizador quer visualizar.
Para visualizar melhor a arquitetura a figura 5.3 pode ser tomada como referência.

5.4 Base de dados

A base de dados utilizada neste projeto é uma base de dados relacional de *SQL server*. Como guia visual desta base de dados a figura 5.4 pode ser tomada como referência. Desta imagem pode ser concluído que a arquitetura dos dados consiste em 9 tabelas. Estas estão organizadas em tabelas de registos e tabelas de informação.
As tabelas de informação são as que registam informações que são para consulta. Estas tabelas são as tabelas *Role*, *Equipment*, *Location* e *Locker*. Estas terão vários graus de acesso, por exemplo, para criar um novo tipo de papel para um utilizador do sistema terá de ser feita a adição por edição da base de dados diretamente, enquanto para as restantes deste grupo, para criar uma nova localização, ou modificá-la, existe um *endpoint* na API que governa a tabela neste sentido.

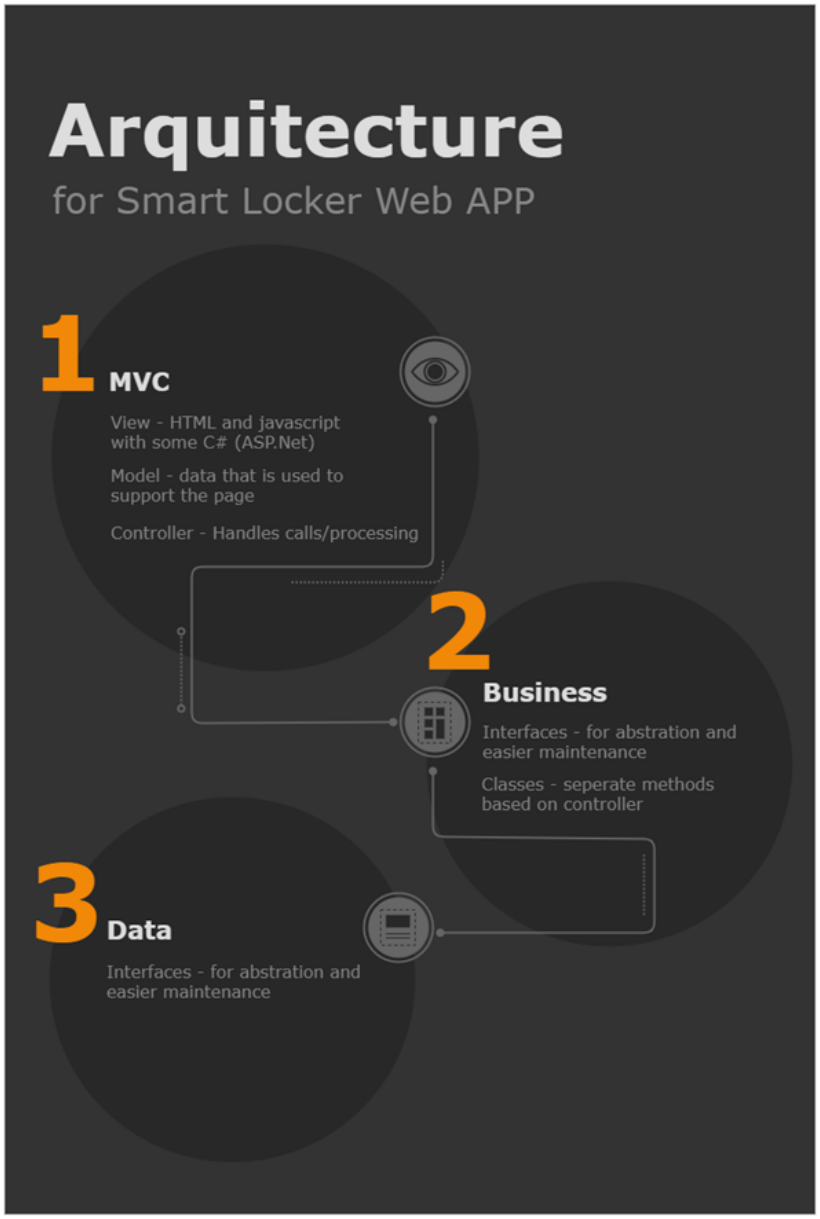

Figura 5.3: Diagrama a alto nível de organização de código da WebApp.

As tabelas de registos são as tabelas de *Assigned*, *User*, *Request*, *Stock* e *Stock Request*. Estas serão as maiores devido à injeção de registos com a utilização por parte dos clientes. Como exemplo, a tabela *Request*, terá um registo de cada vez que um cliente faça um novo pedido de equipamento.

Existem algumas opções que não são óbvias à primeira vista da organização das tabelas que se justificam neste parágrafo. Uma destas opções é a separação dos cacifos da localização. Isto deve-se ao facto de os cacifos individuais estão marcados um a um na base de dados para que esta possa ter informação de qual o cacifo que é suposto ser disponibilizado no caso do levantar de um pedido. Os Roles estão separados dos utilizadores para que esta informação seja mais restrita, sendo que os registos desta tabela deverão ser apenas acessíveis aos administradores da base de dados. Também o controlo de qual o tipo de acesso que um utilizador tem, sendo que todos os utilizadores são registados como “cliente” e só modificado

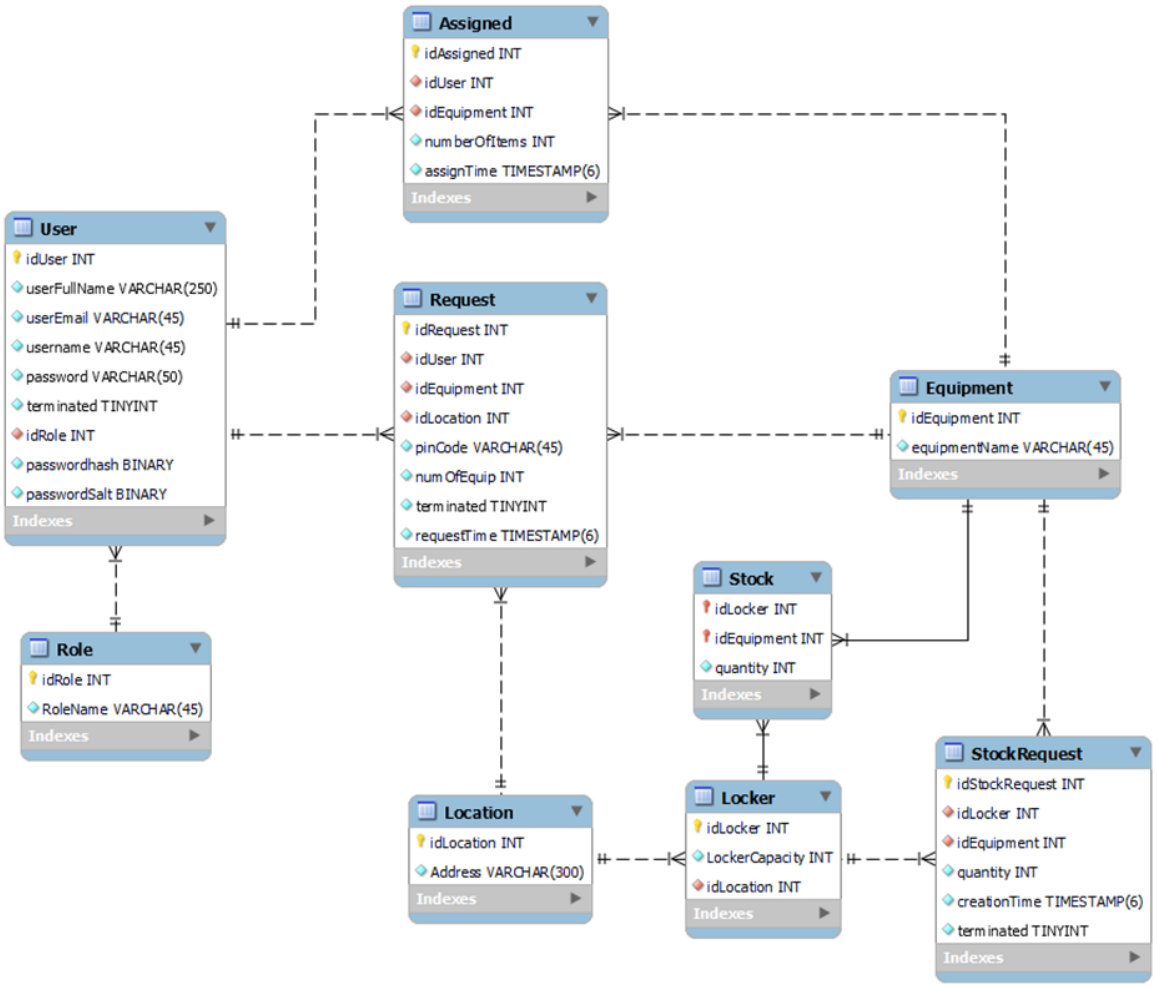

Figura 5.4: Diagrama de organização da base de dados do sistema.

posteriormente.

A base de dados também tem o registo de *stocks*, não só para ter o registo do que está em cada localização, mas também para a utilização desta informação para a gestão de produtos e o seu fluxo, sendo que este fluxo fica gravado nas requisições. O sistema está feito para que as requisições de *stock* para uma localização mantenham a mesma distribuição de uma configuração inicial, sendo que se um cliente levantar um produto, como um *router* do cacifo, será pedido um novo *router* para este cacifo. A tabela de *stock* vai sempre representar que equipamentos estão num cacifo e por *proxy* quantos equipamentos estão em falta numa localização e quantos estão nesta, pela acumulação dos registos de cada cacifo de uma localização. De forma semelhante o *stock* em utilização por clientes está registado na tabela *Assigned* sendo cada registo o acumular de produtos que estão ocupados ou consumidos pelo cliente.

Os identificadores de utilizadores, localizações, papéis no sistema, pedidos de *stock*, etc. poderão no futuro, por questões de segurança, ser modificados para algo diferente. Isto porque será uma fraqueza ter estes identificadores seguidos, por exemplo, utilizador um, dois, três, etc. no caso de ganho de acesso a estes registos. Isto deve-se a que se esta informação for acedida por *query* seria fácil de saber a informação dos utilizadores registados em especí-

fico. A possibilidade de acontecer é reduzida por estar por de trás de código que protege estes registos, mas há que considerar a hipótese de ocorrer.

5.5 API's

As API's que integram o sistema são baseadas na noção da *framework* REST já aceite no mercado para este tipo de operações. Esta *framework* tem como qualquer API a interpretação de um protocolo para comunicação, sendo que este é maioritariamente HTTP, que depois usam a informação que chega aos *endpoints*, neste caso de um controlador, destas para um processamento ou armazenamento de dados.
A evolução que é feita neste sentido é a utilização de uma camada de negócio entre controlador e camada de dados. Isto permite que o controlador seja *lean* e o processamento e validação do pedido à API seja separado de um processamento inteligente da informação. Isto também permite que a manutenção da solução seja mais fácil pela separação das diversas capacidades do sistema em blocos de código diferentes. Além disto, com a utilização de injeção de dependências é transparente a mudança do processamento do sistema para o nível da API, sendo que os nomes de métodos e utilização se manterão, podendo até ser utilizadas tecnologias que não estavam disponíveis ou estavam em falta na execução do projeto, isto sem que seja necessário modificar o código da camada da API.
Para ilustrar melhor a arquitetura que é feita nas API's do sistema, a figura 5.5 deve ser tomada como referência.

5.6 *Orchestrator*

Esta API tem como objetivo fazer a gestão de fluxo de dados, no sentido de que dados devem ser processados e quais são as *pipelines* que devem ser lançadas. Esta também tem como objetivo ser a coordenação de processamento de dados de forma assíncrona para síncrona.
Apesar disto, o sistema é relativamente simples pois é apenas coordenação de informação sem o processamento direto desta. Esta API tem de comunicar tanto com os micro serviços como com a Base de Dados. Por este motivo, esta faz uso na sua camada de dados das três bibliotecas que foram criadas para este projeto: as duas bibliotecas de mensagens, do envio destas e da sua estrutura, e a de comunicação com a API de base de dados.
Existem dois *endpoints* que podem ser utilizados pelo utilizador através da *Web App*. Cada um destes aciona a execução de uma das *pipelines*, pela criação de uma mensagem originada pela biblioteca de mensagens com a informação inicial obrigatória para essa pipeline. A comunicação com a base de dados é a standard com a utilização da biblioteca mencionada anteriormente.
A camada de dados também comunica com o cacifo, depois de receber mensagens da mesma forma que um micro serviço, em caso de sucesso do processamento dos micro serviços do sistema. Esta comunicação é processada e validada na camada de negócio da API. Esta validação é importante para garantir a segurança deste sistema, que assume que o cacifo tem um mecanismo de autenticação da comunicação por parte da API.

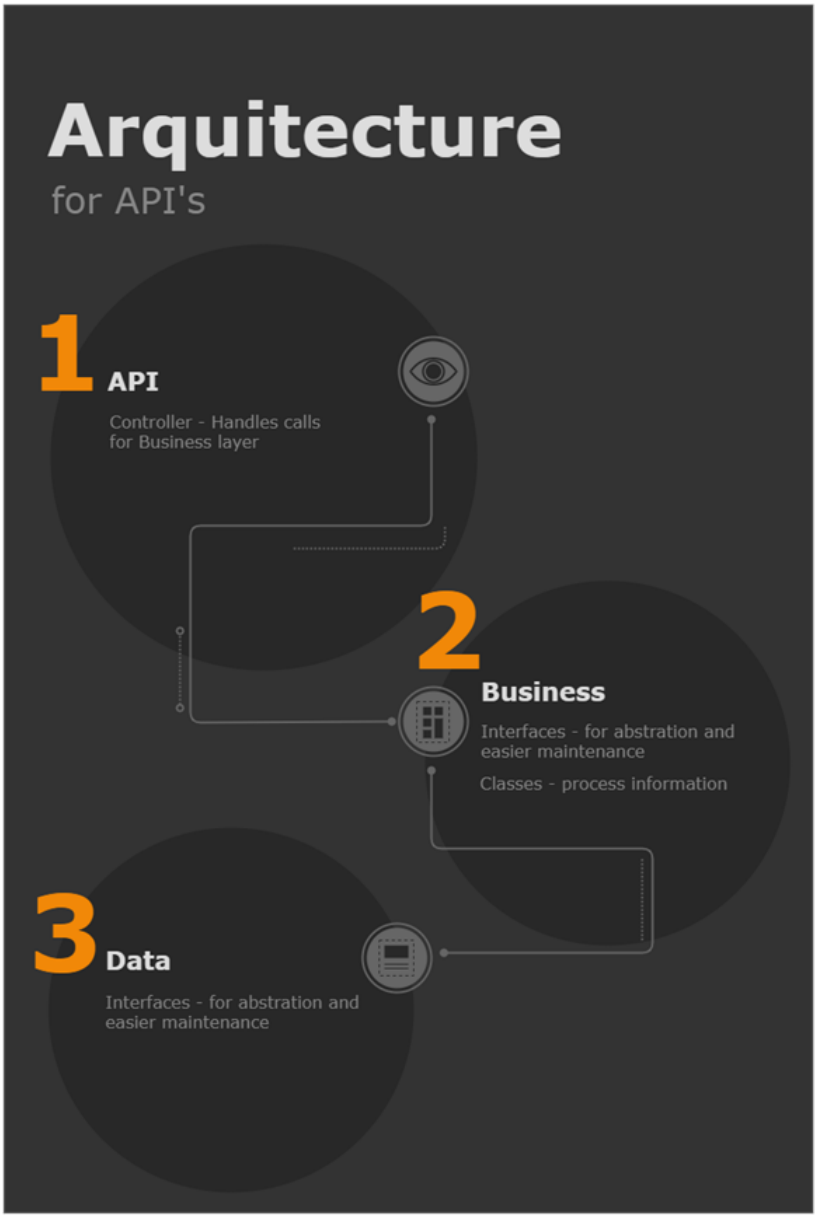

Figura 5.5: Diagrama a alto nível de organização de código de Web API's neste sistema.

5.7 Base de Dados

À semelhança do sistema desenvolvido para o *Orchestrator*, a API tem o código que a compõe organizado da mesma forma. Diferindo na execução principalmente na camada de Dados desta. A camada de dados desta API tem como meio de comunicação com a base de dados um *Nuget Package* chamado *Entity framework Core*. Este *Nuget* abstrai a complexidade de comunicação com a base de dados diretamente e evita o uso de *queries* criadas em SQL. Isto mantém a manipulação de dados na mesma linguagem que tem sido utilizada até esta camada. Esta pode ser uma vantagem mais óbvia do sistema para que a manutenção desta seja mais fácil e que os engenheiros/programadores apenas necessitem utilizar as mesmas ferramentas e otimizações de uma linguagem em vez de duas, neste caso.

Uma grande vantagem deste *Nuget* é a separação clara de versões da base de dados, em

termos de conceção. Isto porque esta tem um sistema de migrações que ficam guardadas. Isto pode ser útil para saber exatamente o que mudou de uma versão para outra das bases de dados feitas com este *Nuget*.
A principal desvantagem da utilização deste *Nuget* é que a construção das *queries* deste são feitas durante a sua execução, sendo que isto atrasa ligeiramente a resposta. Este atraso é mínimo, sendo no máximo na ordem de milissegundos e normalmente em microssegundos, tendo em conta as otimizações que já foram feitas durante os anos em que este existiu.

5.8 Micro Serviços

O sistema foi concebido com o conceito de software em micro serviços. Isto deve-se ao facto deste tipo de arquitetura já ter demonstrado vantagens sobre sistemas monolíticos que se construíam antes desta arquitetura. Apesar de ter o problema de ter uma comunicação do sistema internamente que pode atrasar o sistema ligeiramente em relação a um sistema monolítico, já é minimizado pela utilização de certas tecnologias que melhoram estas transferências e inclusive garantem segurança de transição de mensagens entre elas.
Este tipo de arquitetura é muito flexível e escala melhor em termos de disponibilidade de serviço. Isto deve-se à separação inerente que é feita das operações. Cada um dos serviços deve ser simples tendo uma ou duas operações de processamento de dados, isto é para ter o mínimo de impacto no sistema em caso de modificação do código devido a erros, um *upgrade* ou alteração de regras de negócio.
A arquitetura geral de cada um dos micro serviços é comum a todos eles, sendo que existe a componente de *worker*, que faz o *setup* inicial do programa, a parte de *business* que é inicializada no *worker* e mantém a lógica do sistema, inclusive o tratamento do conteúdo de uma mensagem recebida, finalmente, a última camada é a camada de comunicação que faz o uso das bibliotecas de mensagens e de interação com a base de dados para o funcionamento da camada de *business*. Esta arquitetura está ilustrada na figura 5.6.
Para que seja transparente para o leitor, menciona-se que as mensagens que passam pelas *Queues* do sistema são em JSON. Isto deve-se a que este tipo de formato de dados é de fácil interpretação humana e bibliotecas como o *NewtonSoft* facilitam o seu uso com código, neste caso este é um *Nuget* de C#. Assim esta escolha foi feita por forma a que se mantenha uma fácil interpretação dos dados e que exista flexibilidade entre que tipo de tecnologias são utilizadas em cada um dos micro serviços. Uma possibilidade desta arquitetura é que o sistema seja convertido lentamente numa outra tecnologia sem comprometer o seu funcionamento.

5.8.1 Pipelines

Como anteriormente mencionado na secção de visão geral do sistema, foram criadas duas *pipelines* para processamento dos dados do sistema. Estas funcionam de forma assíncrona para que seja possível a paralelização do sistema. Cada uma destas *pipelines* está dividida em vários micro serviços para que seja possível escalar este paralelismo mesmo para processos mais demorados e adaptabilidade à solicitação do sistema de forma dinâmica. De momento nenhum serviço da nuvem é utilizado como o *Azure* da Microsoft nem o *amazon web services*

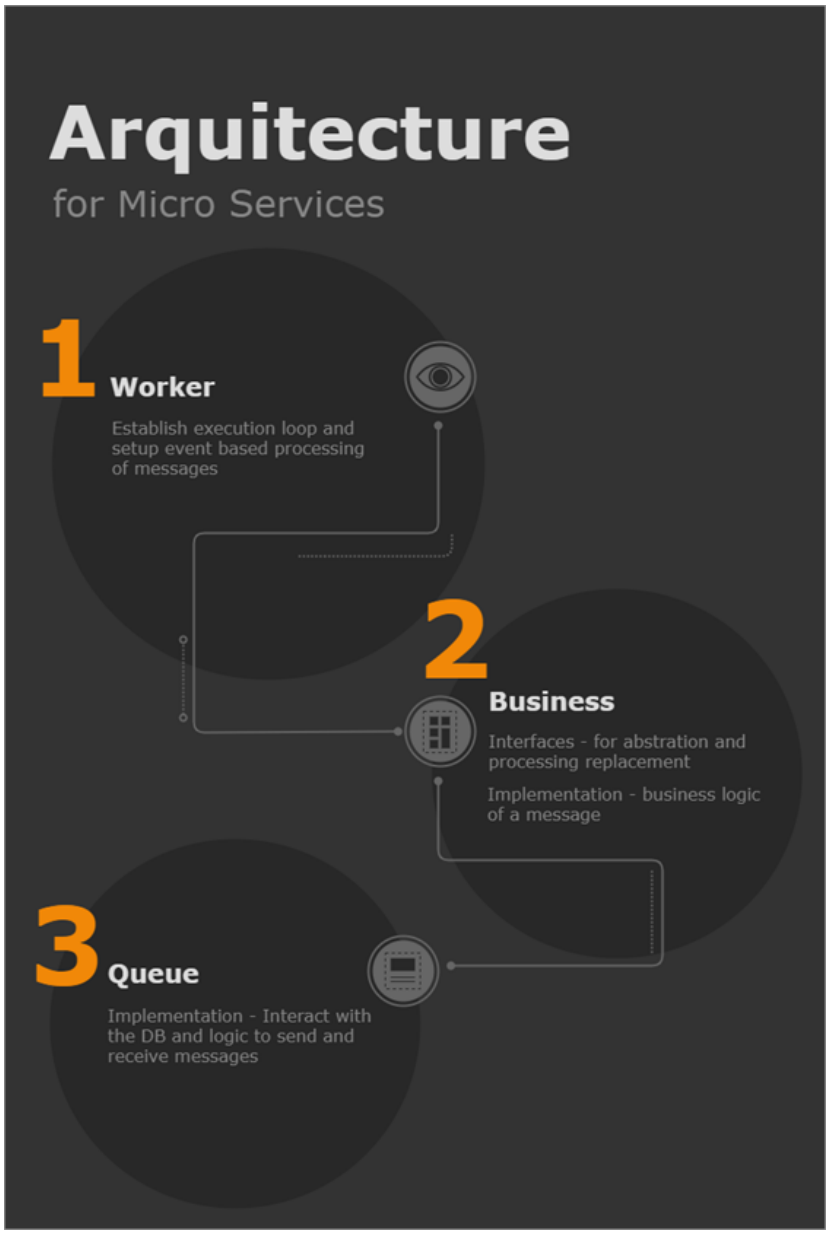

Figura 5.6: Diagrama a alto nível de organização de código de micro serviços neste sistema.

(AWS) da *Amazon*, mas caso se verifique benéfico para a empresa, este passo já permite uma melhor adaptação a estes sistemas.

Cada um dos micro serviços poderá ser inserido em *containers Docker* e coordenado o número de instâncias por *Kubernets*. Estas tecnologias estão cada vez mais a ser utilizadas pelos benefícios que oferecem em termos de escalabilidade em relação a uma solução monolítica.

Para além do que já foi mencionado, as manutenções de pontos específicos do sistema também se tornam mais fáceis. Isto porque cada um dos processos envolvidos num micro serviço é inerentemente simples. Esta simplicidade de cada componente significa que caso a manutenção seja um erro de código, ou um problema de negócio ou uma simples mudança de operação, o sistema pode continuar a executar mesmo que um módulo seja retirado. Sendo que tudo o que está dependente de um dado módulo em manutenção terá um corte na sua execução, mas assim que este é adicionado, poderá processar todas os pedidos que foram

acumulados nas filas de comunicação que o precedem.
A figura 5.7 representa as ligações dos micro serviços que estão incluídos nesta arquitetura. Como é possível observar todos estes comunicam tanto com o micro serviço de *logging* como com a API de base de dados. Esta comunicação é para o registo de erros no caso do sistema de *logging* e acesso a informação da base de dados para a utilização do sistema e processamento de pedidos por clientes.

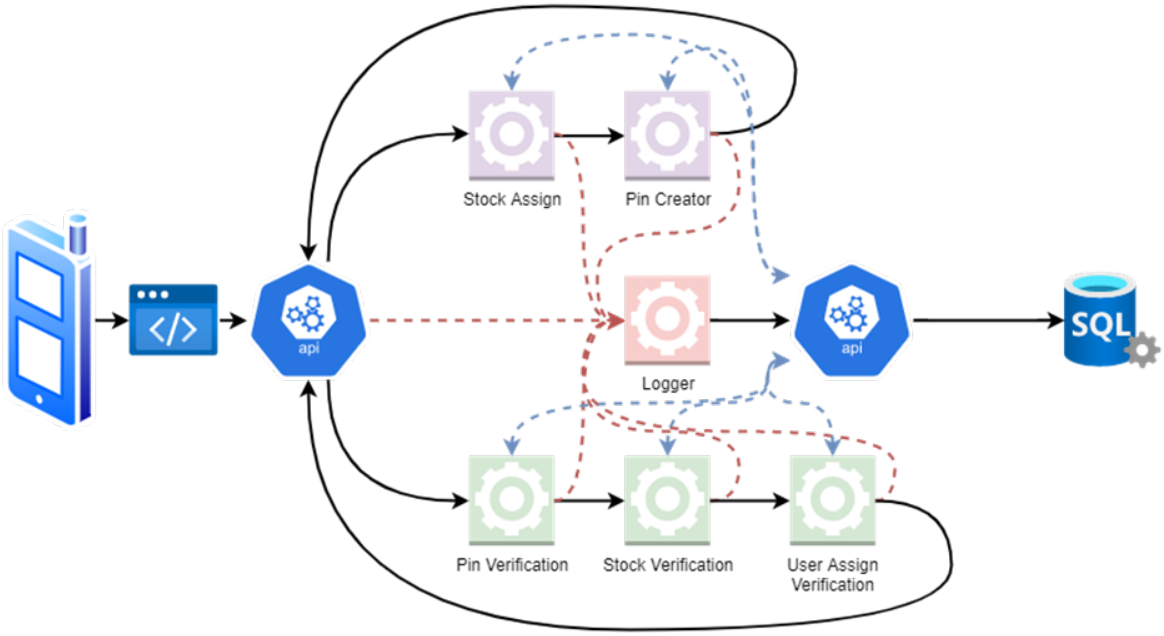

Figura 5.7: Diagrama de organização de pipelines de processamento no sistema.

A lilás está representada a pipeline de *Assign* e a verde a pipeline de *Verification*. A vermelho está representado o micro serviço de *logging*. Este último, consoante a perspetiva, tanto faz parte das duas *pipelines* como de nenhuma, pois pode ser o término de uma pipeline por nesta ter ocorrido um erro, ou nunca executar em caso desta ter um processamento previsto. Sendo que este micro serviço é uma interface à API de base de dados para registo de erros.
O processo de *Logging* inicialmente estava idealizado para o registo de todas as mensagens que tinham sido enviadas ao longo do tempo pelo sistema mas este sistema não foi a implementação final. Da ideia inicial ficaram apenas as mensagens que dariam em erros, isto deve-se ao facto de que o sistema em caso de não haver erros já regista informação sobre transações de produtos nas tabelas da base de dados. Tendo isto em conta, o registo de mensagens seria supérfluo pela duplicação da informação da comunicação para além de tornar mais difícil a procura de erros que tenham ocorrido no sistema.

5.8.2 *Assign Pipeline*

Nesta subsecção será apresentado o objetivo dos micro serviços da pipeline *Assign*. Esta explicação tem como objetivo mostrar o que se propõe com cada um dos micro serviços e não uma explicação de organização de código ou implementação técnica, sendo que isto já foi feito na introdução ao capítulo. Além disto, para cada um dos serviços é feita a apresentação das mensagens que se esperam receber e enviar para a sua execução.

5.8.2.1 *Stock Assign*

O início de um ciclo de processamento de um produto que tem como fim estar na posse de um cliente começa neste micro serviço. Como tal, este micro serviço recebe uma mensagem com as informações básicas de um pedido de um produto na sua secção de processamento. Estas informações são o identificador do utilizador, neste caso um número, um identificador da localização onde pretende levantar o produto, os produtos que pretende levantar, inclusive o número de cada um e a data do pedido deste. A figura 5.8 ilustra esta mensagem.

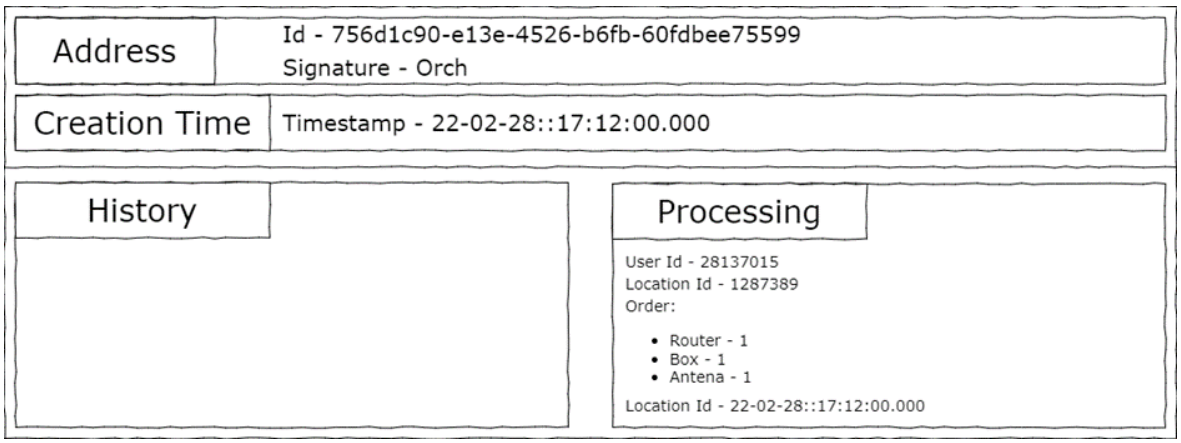

Figura 5.8: Ilustração de uma mensagem do *Orchestrator* para o micro serviço *Stock Assign*.

Estas informações são utilizadas para reserva dos equipamentos para um registo de pedido de produtos. Cada um dos produtos pedidos será um desses registo. E como otimização para limitar o número de acessos à base de dados, a informação de *stock* de uma localização é guardada na mensagem. Esta informação é acompanhada por uma variável "*StockAvailable*" que é o contexto da existência de stock no momento em que o cliente fez o pedido, para evitar que este se desloque ao local sem que estejam os produtos desejados na localização. A figura 5.9 ilustra esta mensagem.

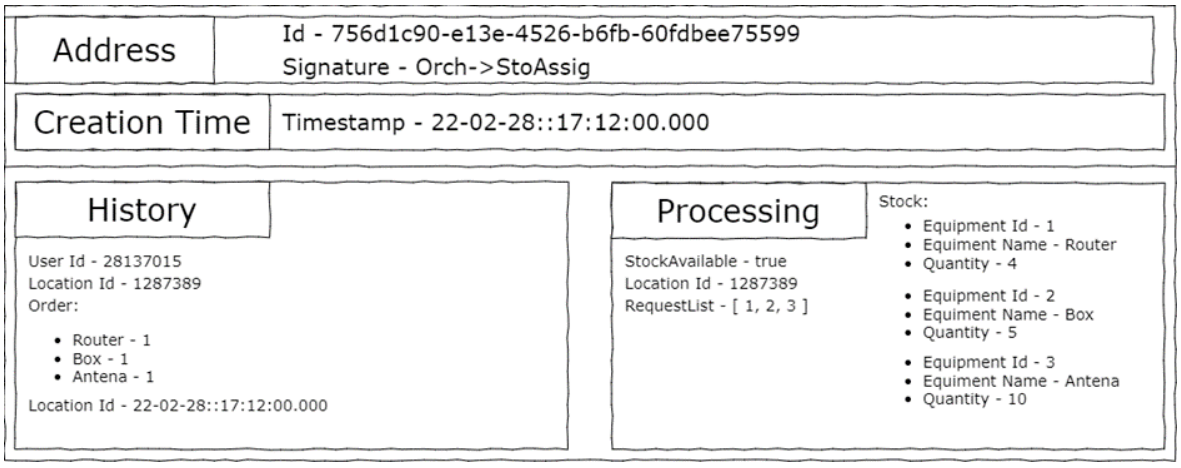

Figura 5.9: Ilustração de uma mensagem do *Stock Assign* para o micro serviço *PIN Creator*.

5.8.2.2 *PIN Creator*

No continuação deste processamento inicial de um pedido de um produto por um cliente, tem de ser solidificado o resto do registo, além de ser verificado que o *stock* está ou não disponível, como foi feito na mensagem anterior. Assim, este micro serviço cria o PIN de acesso ao cacifo e para manter o *stock* disponível faz uma redução do número de artigos que estão presentes no local onde o cliente quer levantar um produto, para emular o processo reserva deste.
A mensagem que sai deste serviço de momento não tem qualquer uso sendo apenas o manter da arquitetura coesa. No futuro esta informação poderá ser utilizada para confirmar em base de dados que a reserva foi feita e que esta está válida no momento que se encontra. A figura 5.10 ilustra esta mensagem.

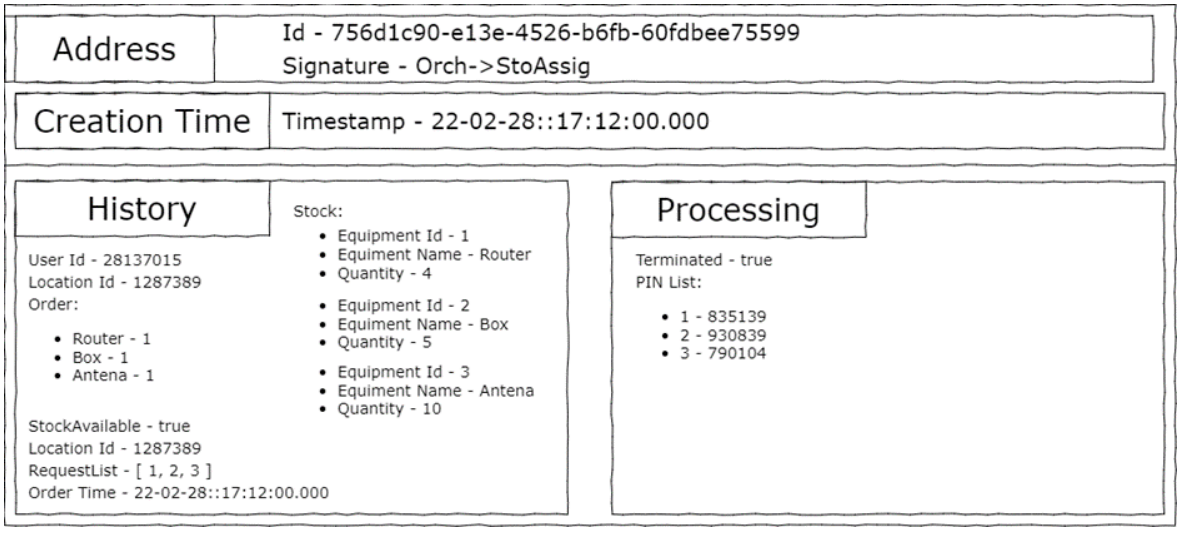

Figura 5.10: Ilustração de uma mensagem do *PIN Creator* para o *Orchestrator*.

Esta função não se encontra implementada por o processamento desta informação ser instantâneo para demonstração do protótipo. Também não é possível prever o que a Altice quer partilhar com o utilizador em termos de reserva de itens e assegurar a segurança interna de o utilizador não perceber o processamento nos bastidores do sistema.

5.8.3 *Verification Pipeline*

Nesta subsecção será apresentado o objetivo dos micro serviços da pipeline *Verification*. Esta explicação tem como objetivo mostrar o que se propõe com cada um dos micro serviços e não uma explicação de organização de código ou implementação técnica, sendo que isto já foi feito na introdução ao capítulo. Além disto, para cada um dos serviços é feita a apresentação das mensagens que se esperam receber e enviar para a sua execução.

5.8.3.1 *Pin Verification*

No começo de processamento que é feito neste pipeline é verificada a informação que é dada pelo utilizador para a finalização de um pedido de um utilizador na sua perspetiva. Ou seja, abertura de um cacifo para levantar um produto.

Para isto é necessário que o utilizador forneça a sua identidade, neste caso o seu ID, o número do pedido que foi feito por este, que também é um ID e finalmente o código que é definido pelo pipeline anterior, para a abertura do cacifo. Com estas informações será possível ter um alto nível de segurança no acesso do utilizador ao sistema. A figura 5.11 ilustra a mensagem com estas informações.

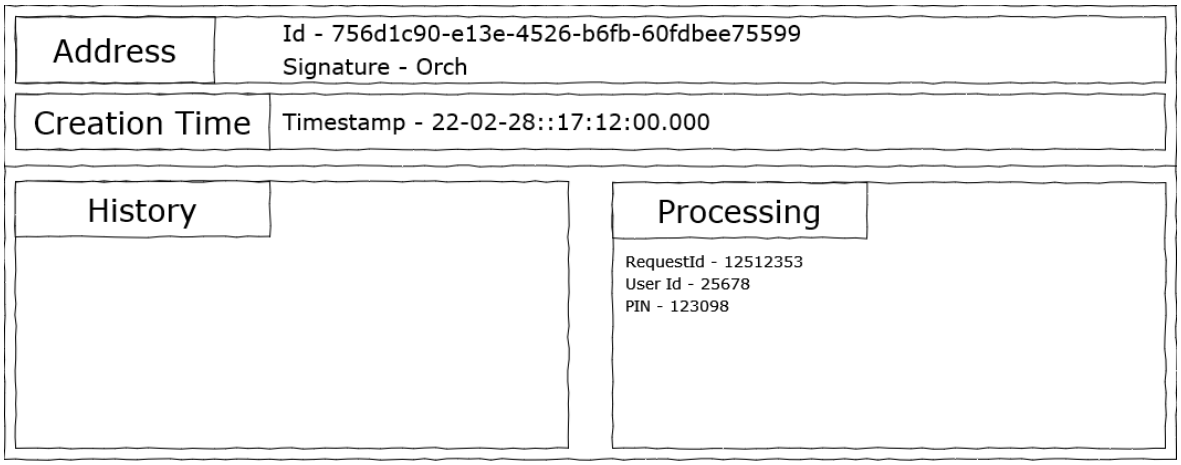

Figura 5.11: Ilustração de uma mensagem do *Orchestrator* para o *Pin Verification*.

Uma possível falha de segurança do sistema é o facto de não ser necessária a presença física do cliente para a abertura do cacifo, uma vez que que o próprio pode abrir o cacifo remotamente. O sistema presente pode ser alterado para não existir esta falha, consoante a implementação do *hardware* do cacifo.
A mensagem que é enviada está ilustrada na figura (inserir figura) para efeitos de visualização. A mensagem que é resultante do processamento desta informação é um resumo da informação que compõe o pedido. Isto inclui, a aprovação do PIN do utilizador, se foi válido ou inválido, a informação específica do pedido como guardado na tabela da base de dados, e informações mais específicas do pedido como o *stock* do local onde é recolhido o produto e uma separação para facilidade de acesso do próximo micro serviço à informação sem pesquisa no objeto da base de dados. A figura 5.12 ilustra a mensagem de resultado do processamento deste micro serviço.

5.8.3.2 *Stock Verification*

Com o processamento e aprovação inicial do pedido, será necessária a reposição do *stock* do cacifo. Com isto o propósito deste micro serviço é a reposição para o manter do serviço. Esta é uma questão que não faz parte da proposta do estágio, de forma direta, mas sim da gestão de *stocks*. Contudo, sem esta verificação o sistema rapidamente seria mais um problema para a empresa do que um complemento a um serviço.
As informações Processadas são incluídas na mensagem sem que seja processada outras informações. Este processamento é o mantém a viabilidade do sistema e para um futuro em que se guardem todas as mensagens do sistema será um bom método de verificar flutuações de *stock* possivelmente indevidas. O resultado de processamento dos dados do sistema de stock ficam representados na mensagem como ilustrado na figura 5.13.

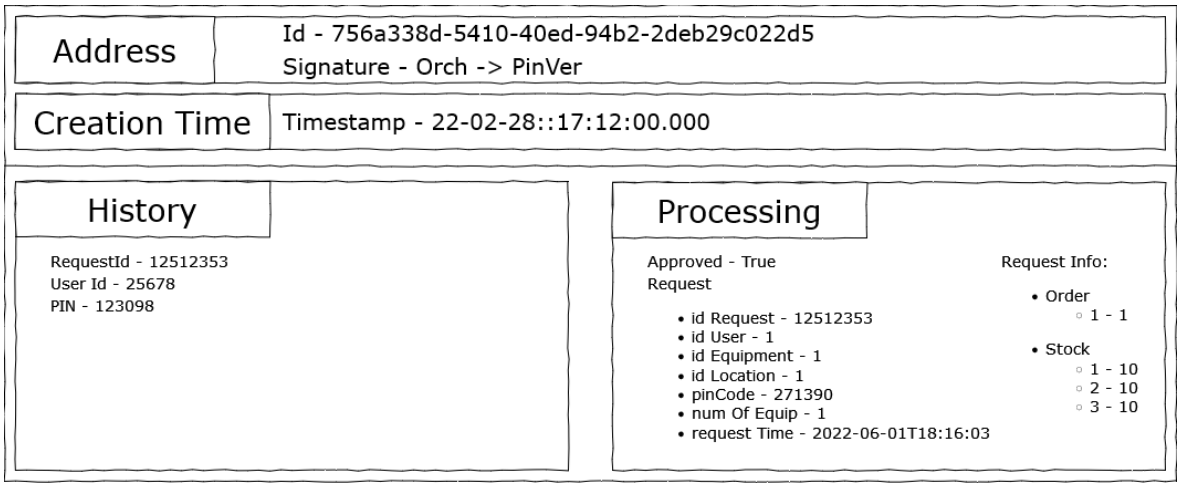

Figura 5.12: Ilustração de uma mensagem do *Pin Verification* para o *Stock Verification*.

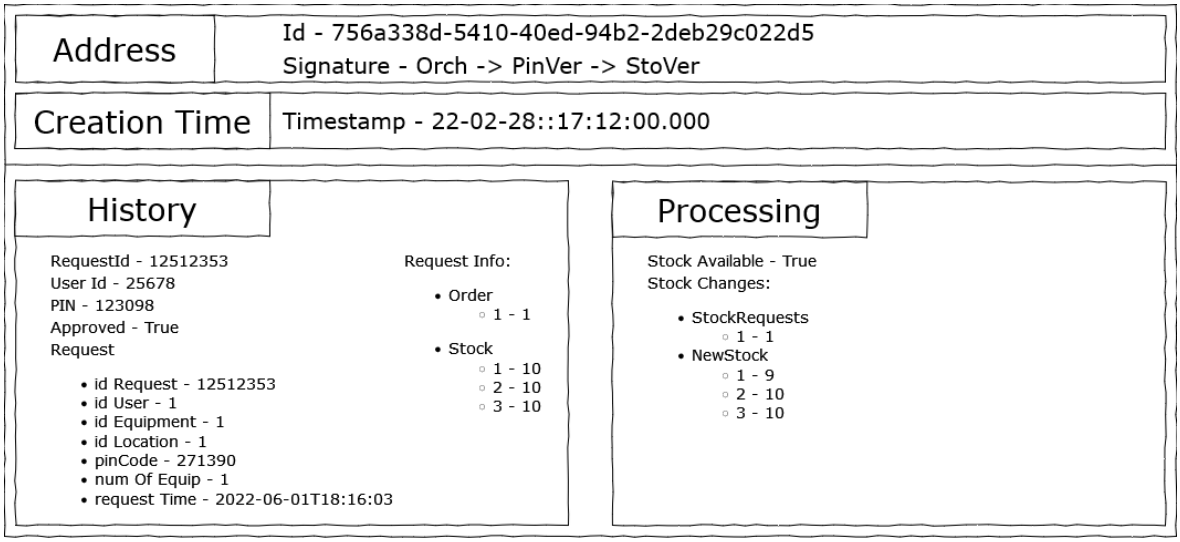

Figura 5.13: Ilustração de uma mensagem do *Stock Verification* para o *User Assign Verification*.

5.8.3.3 User Assign Verification

Para finalizar o processamento do pedido o sistema regista os produtos que foram disponibilizados ao utilizador. Este registo é importante para uma recolha destes equipamentos no caso de uma cessação de contrato, casos de abuso do sistema ou uma simples análise de fluxo de *stock* no futuro da empresa e projeção de inventário. Este processamento fica gravado no campo *processing* da mensagem como representado na figura 5.14

Este micro serviço cria os registos e modifica os pedidos do utilizador para os marcar como realizados. Isto implica que um utilizador não terá acesso ao cacifo após levantamento do artigo, como seria de esperar.

Dependendo de como será feita a implementação para recolha de informação da Altice para análise dos seus clientes, a informação disponibilizada por estas mensagens finais de processamento da recolha, pode ser utilizada para a análise de uma possível expansão ou redução do número de cacifos que estão numa dada localização. Isto pela frequência de interação dos clientes com os sistemas. Inclusive recompensas em loja por interação com o sistema,

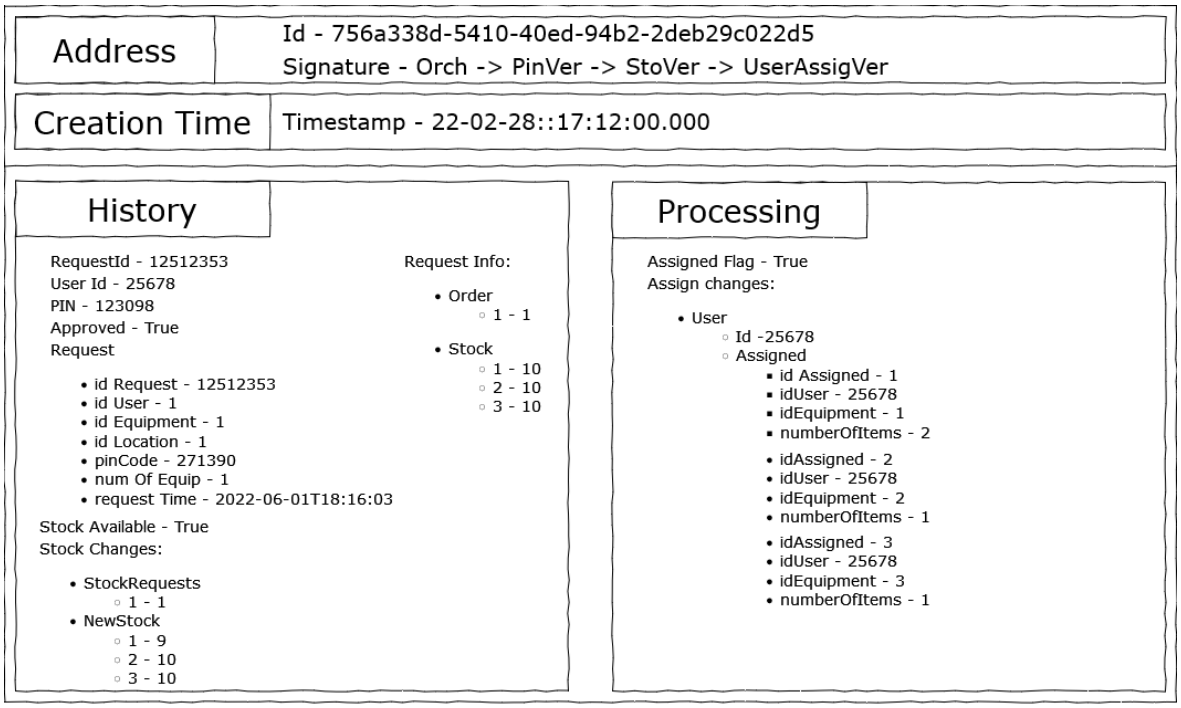

Figura 5.14: Ilustração de uma mensagem do *User Assign Verification* para o *Orchestrator*.

pela poupança intrínseca à empresa pela utilização deste serviço e pela adesão deste por um período inicial.

5.8.4 Logging

O esquema do sistema estava inicialmente idealizado para adicionar todas as mensagens ao registo de base de dados do sistema. Isto serviria para saber o que teria sido utilizado pelo sistema para uma visão posterior do que o sistema teria utilizado para o seu processamento. Isto aumentaria muito o número de registos que seriam feitos pelo sistema para cada um dos pedidos de um utilizador. Sendo que o número de registos mínimos na base de dados por cada um dos pedidos seriam 7 mensagens. Este número aumentaria por cada erro que fosse ocorrendo, por exemplo um PIN errado.
Dito isto, as mensagens que chegam a este micro serviço são as mensagens que chegam aos serviços descritos anteriormente. Ou seja, em caso de erro em qualquer um dos outros micro serviços a mensagem recebida é reenviada para este serviço e guardada na base de dados como forma de verificar que erro originou a falha posteriormente. Isto será feito por uma execução teste do micro serviço que originou o erro com a mensagem e reutilização desta em ambiente de desenvolvimento para avaliação do comportamento desse serviço.

5.9 *Hardware*

Inicialmente esta parte do sistema seria uma colaboração entre a Altice e uma empresa exterior que contribuiria com o *hardware* e experiência em sistemas semelhantes. Isto para que

o sistema fosse robusto e mantivesse o *hardware* atualizado, sendo que assim a Altice poderia manter o foco interno na melhoria do produto e outras possíveis utilizações do sistema nos negócios já estabelecidos na empresa via software.
Esta colaboração foi extinta em Maio. Com a sua extinção houve uma tentativa de implementação do sistema como algo que demonstrasse casos de abertura de um cacifo via uma arquitetura projetada em conjunto com o professor orientador. Não foram fornecidas condições para o seu desenvolvimento.
Assim o desenvolvimento deste sistema não foi conseguido. Sendo que o sistema apenas foi demonstrado à empresa pelo software desenvolvido como explicado nas secções anteriores.

Capítulo 6

Execução do Estágio

6.1 Introdução

Neste capítulo, é feita uma descrição das atividades desenvolvidas em cada semana do estagiário. Aqui faz-se a representação das atividades que ocorreram durante o estágio desde reuniões a tarefas inerentes a este. Criação das arquiteturas de *software*, programação e comunicação com a Altice e empresas externas.
Além disto também se documentou neste capítulo o histórico de implementação da solução e como tem sido alterada a solução no decorrer do estágio. Sendo todas as mudanças de funcionamento documentadas, desde bases de dados a interações entre micro serviços e APIs.

6.2 Relatório Semanal

Durante a primeira semana do estágio, que vai de 21 de fevereiro a 25 do mesmo mês, as atividades foram de introdução à empresa, sendo estas de forma remota ou na empresa. Começando por uma breve apresentação da empresa por parte dos recursos humanos. Esta teve como objetivo apresentar os vários ramos de atuação da empresa, tanto em termos de áreas de desenvolvimento como em termos organizacionais.
Com isto foi possível ter uma melhor noção da organização, métodos de pesquisa, de pessoas chave, de chefia dos vários setores da empresa, valores da empresa e algumas regras de trabalho que estão de momento implementadas no momento do estágio. Estas regras variam desde regras organizacionais, políticas de trabalho em regimes remoto e teletrabalho, entre outras.
Posteriormente foi feita a receção da máquina para desenvolvimento da solução. Foi necessário configurar termos de utilizador, instalação de programas como *visual studio code* para começar o desenvolvimento. Para viabilizar o trabalho remoto também foi configurada a VPN interna da empresa a pedido desta.
Para efeitos de registo de *hardware* utilizado para desenvolvimento do projeto refere-se aqui que a máquina que foi fornecida tinha as seguintes características de *hardware*:

- Processador - Intel Core i3-6100U CPU 2.30GHz
- RAM - 4.00 GB, sendo 3.90 GB usável sendo que 3.6 são utilizados pelo sistema operativo em inatividade

Posteriormente a 28 de abril a RAM da máquina foi aumentada para 8.00 GB, sendo 7.90 GB usável com os mesmos valores de ocupação pelo sistema operativo.

A configuração da máquina foi algo que representou um problema até dia 3 de março, sendo que nenhuma instalação podia ser feita a não ser que fosse *software* aprovado para qualquer pessoa da empresa. Para o desenvolvimento do projeto havia a necessidade de instalar ferramentas cruciais como o PHP, que não está instalado por defeito, mas pedida pela empresa e, por isso, seria a linguagem que seria a utilizada. O pedido da utilização da linguagem PHP representou uma condição que foi alterada a meio de março, por um lado devido a dificuldades no acompanhamento diário do estágio por parte da empresa e por outro lado numa perspetiva de futuro, tendo surgido o reconhecimento que existem alguns problemas inerentes a PHP em comparação com a plataforma *.NET* que está muito mais desenvolvida.

Nesta primeira semana também se começou a explorar melhores práticas de implementação e tecnologias a utilizar para o modelo de implementação pretendido. Esta tarefa ficou em espera devido à falta da instalação do PHP. Entretanto foram fornecidas por parte do orientador da Altice, engenheiro Valter Vale, alguns programas que seriam utilizados no desenvolvimento da solução para manter o projeto para posterior inovação e manutenção por parte da empresa mesmo depois da conclusão do projeto por parte do aluno. Estes são PHP, *laragon*, *netbins php store*, *fillezilla*, *putty*, *mttree*, *sql studio*, *workbech mysql* e *heidi*, sendo alguns destes apenas opções para uma mesma função. Vários destes programas foram utilizados durante o percurso do sistema sendo que alguns entraram em desuso devido à mudança de linguagem para C#.

Assim sendo, no decorrer desta primeira semana foi feita a arquitetura do sistema de forma que quando se tornasse possível apenas fazer desenvolvimento sem que fosse necessário parar para planeamento. Isto revelou-se ser uma mais-valia para a criação de uma solução coesa do sistema e já com algumas nuances sobre o funcionamento através da avaliação do que é necessário ao sistema.

Assim o primeiro passo foi a idealização do sistema de forma muito breve, para identificação de *pipelines* de processamento para uma arquitetura em micro serviços.

Esta análise de requisitos resultou em dois diagramas, um referente à base de dados e o outro à organização de micro serviços para o processamento de pedidos dos utilizadores pela *Web API* e pelo próprio cacifo para validação do código inserido e processamento de que cacifos serão abertos para a satisfazer o pedido do cliente. O diagrama referente à base de dados está representado na figura 6.1. O referente à arquitetura de micro serviços está na figura 6.2.

A base de dados idealizada está mais focada na funcionalidade do que na segurança, para o momento em que foi criado, sendo esta apenas um rascunho provisório. Isto ocorre porque o sistema é um protótipo que servirá como base para o sistema que virá a ser implementado pela Altice. Posteriormente, foi atualizada para ter um método de *login* mais abrangente, incluindo um nome de utilizador ou *email* para *login*.

O desenvolvimento posterior solidificou mais esta estrutura, sendo que o facto de esta ser um método de demonstração mantém-se. As cores das tabelas ilustram a idealização do nível de acesso necessário. Vermelho indica um nível de administrador, amarelo o nível de processamento e informações cruciais do sistema e verde é o nível de acesso público. Esta última é a informação que o utilizador poderá consultar diretamente via *Web App*.

A arquitetura do projeto criado pelo estagiário é dividida em 2 APIs internamente, dando

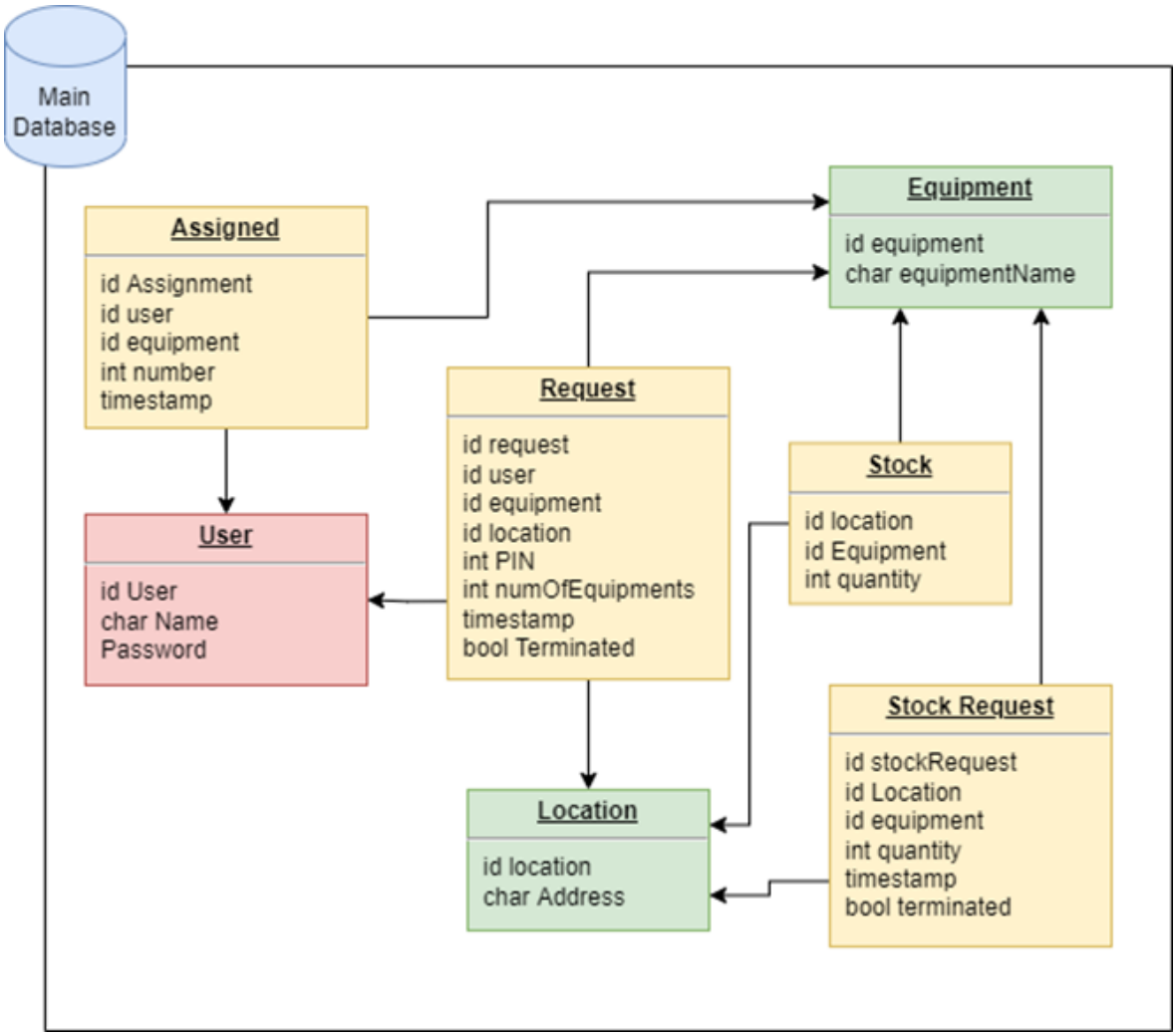

Figura 6.1: Diagrama a alto nível de arquitetura inicial da Base de Dados.

uma abordagem mais escalável à solução. Isto por separação dos principais tipos de dados que o projeto irá utilizar: base de dados para registos persistentes e filas para troca de mensagens entre micro serviços. A primeira API envolvida é o *Orchestrator*, este inicia e finaliza os *pipelines* e trataria de eventuais execuções e registo de erros, sendo que esta função passou no futuro para um micro serviço de *Logging* para a base de dados. A API a vermelho é a API que trata da interação com a base de dados, para cada serviço e a última API a mencionar. As *pipelines* identificadas estão representadas a roxo e amarelo, para uma mais fácil visualização e separação. A *pipeline* a amarelo é responsável pelo processamento e atribuição de *stock* e criação de PINs. O *pipeline* a roxo faz a verificação do código PIN e do *stock* do local. Este último também enviaria para o controlador do cacifo as informações de qual dos cacifos a abrir, conforme solicitado, sendo que este último passo é feito pelo *Orchestrator*.

Também está incluído no último diagrama o fluxo de dados, de onde vem, o que será processado e o que será passado para o próximo micro serviço. Isto foi usado posteriormente para criar a estrutura das mensagens do sistema.

Um estudo de fundamentos de PHP foi começado, para o conhecimento das características básicas da linguagem. Com base neste estudo, um tutorial encontrado nesta semana foi

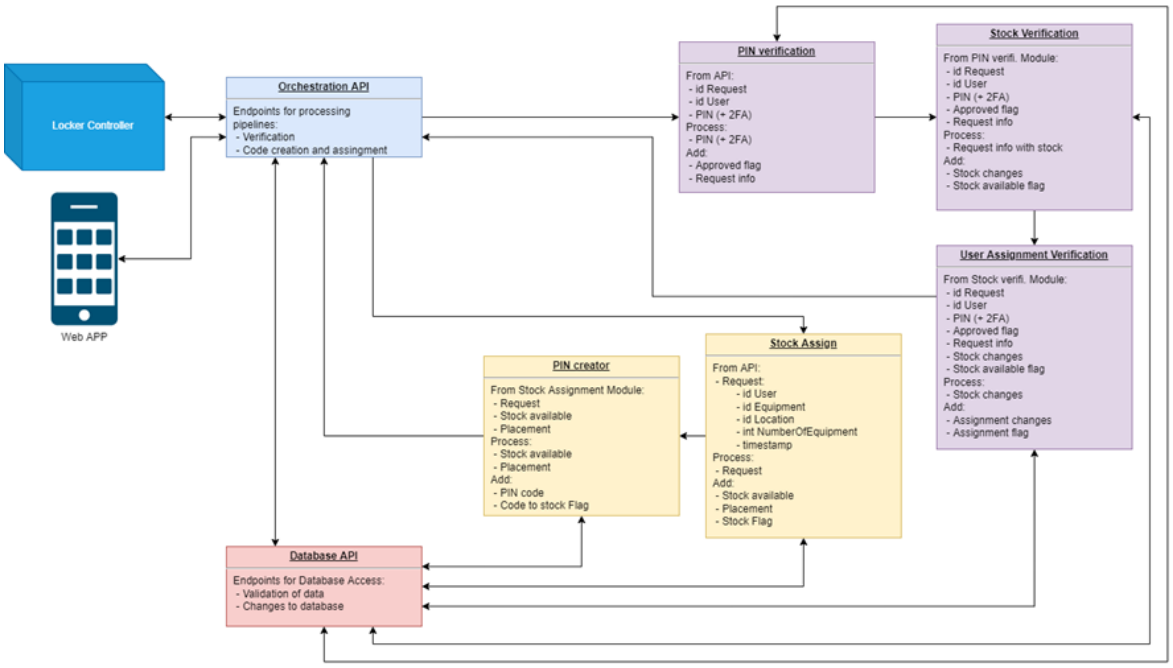

Figura 6.2: Diagrama a alto nível de arquitetura inicial do projeto.

usado para implementar as 2 APIs principais. O foco era a API de base de dados, este parecia ser o passo mais importante, pelo manuseio dos dados ser a parte principal de qualquer *software*. Isto foi também uma oportunidade de extensão da arquitetura proposta pelo site, passando o processamento dos dados para uma camada Business em vez de ser feito no nível do *controller*, tornando estes mais complexos. Para finalizar o desenvolvimento base desta API foram criadas todas as *queries* que eram consideradas obrigatórias.
A estrutura de ficheiros que foi sugerida pelo tutorial é a seguinte:

- *Index* – este ficheiro é executado quando queremos iniciar o programa, este utiliza o ficheiro *bootstrap* para inicializar o programa.
- *Controller*
  - API
    - *Base Controller* – ficheiro que será estendido por cada *controller* para manter o tratamento de URIs uniforme, evitando código repetido.
    - Qualquer controlador será adicionado aqui no mesmo nível do controlador base.
- Inc
  - *Bootstrap* – ficheiro contém os caminhos para o ficheiro de configuração e os ficheiros das restantes pastas, sendo o conteúdo, referências para todos os ficheiros necessários para executar o programa.
  - *Config* – ficheiro que contém segredos para operação como nome da base de dados, nome de utilizador e password.

- *Model*
  - *Database* – define as interações básicas com a base de dados pelo mesmo motivo do base *controller*, clareza e evitar duplicação de código.
  - Os modelos de dados definem a interação com cada uma das tabelas da base de dados, as consultas que são estritamente necessárias foram criadas, mas podem ser expandidas no futuro caso seja necessário.

Este tipo de arquitetura também é o aconselhado para C#. Como tal, esta estrutura de organização do código foi mantida, mesmo após a alteração da linguagem utilizada no desenvolvimento da solução.
A estrutura de ficheiros inicial de todo o projeto foi criada para manter o código desenvolvido separado para cada tipo de solução e manter a clareza do que é tratado em cada uma. Isto passou pela divisão das APIs e micro serviços em pastas separadas. A separação do código em PHP, foi dividida em APIs com cada uma com a sua própria pasta e os micro serviços foram separados por *pipelines*, sendo que cada um destes também tem a sua própria pasta. Isto foi uma mais-valia para a posterior conversão da solução para C#, por ter sido mais rápida esta mudança. As soluções em C# não partilham estas divisões de pastas, sendo que todas ficaram ao mesmo nível. A descrição do sistema coincide com os nomes no repositório sendo este este fator não cria confusão.
O controlo de versões foi feito por backups no final de cada dia para a *cloud* dos serviços fornecidos pelo *Microsoft OneDrive*. Esta solução foi posteriormente atualizada para utilizar o Azure como *github* privado da empresa.
Também foi concedida aprovação para administração da máquina, com a condição de ser alocado permanentemente ao estagiário durante o desenvolvimento do sistema. Isto já seria esperado devido à política da empresa para proteção de dados e *hardware*. Isto significa que, assim que as configurações fossem implementadas, estariam reunidas as condições para iniciar desenvolvimentos. Que, como mencionado, foi a 3 de março, na semana seguinte.
Com a continuação da pesquisa e a escala do projeto que foi elaborado, o estagiário teve como sugestão a utilização de bibliotecas para uma melhor regulação do código do sistema e redução de código duplicado. Neste caso foi logo idealizado que pelo menos 2 bibliotecas teriam de ser feitas para o sistema. Um cliente HTTP para a comunicação com a base de dados e uma para a comunicação entre as *pipelines*. Bibliotecas tinham sido encontradas como o *MeekroDB*, *Guzzle* e *RabbitMQ* para interações com base de dados, cliente de HTTP e mensagens entre micro serviços respetivamente. Quando foi passado para C# estas passaram a ser *Entity Framework Core*, *Http Client* integrado na linguagem e manteve-se *RabbitMQ*. Todas estas são *open source*, que também era um objetivo para a criação deste protótipo para a redução de custos.
Na conclusão desta primeira semana foram feitas várias reuniões. Uma destas sendo com o orientador de estágio por forma a fazer uma revisão do sistema idealizado pelo estagiário. Esta arquitetura foi aprovada com algum *feedback* para melhoria da base de dados que foi posteriormente implementado e refletido na última versão que está representada neste documento. Outra reunião foi com uma empresa especialista em soluções de *smart lockers* e

que se inseriu neste projeto na perspetiva de colaborar fazendo parte do sistema, providenciando *hardware* e *software* para interação com o mesmo. Esta reunião teve como objetivo criar um canal de comunicação e uma apresentação da empresa ao estagiário e ao professor orientador deste estágio e também para combinar o que seria disponibilizado no futuro. É de mencionar que esta parceria foi extinta em maio devido ao facto de não ter sido disponibilizado qualquer *hardware* ou conhecimento técnico do sistema por parte desta empresa.

Durante a segunda semana do estágio foi necessário fazer formação especifica, da empresa, integrada em políticas internas de formação. Alguns dos temas abordados foram informações de segurança no trabalho, sistema de gestão da empresa e regulamentos de proteção de dados.

Entretanto a pesquisa do funcionamento da solução que seria providenciada pela empresa parceira, especialista em soluções de *smart lockers* foi difícil tendo em conta que existe pouca informação existe sobre o sistema. No entanto verificou-se que era utilizado um sistema de *touchscreen* para a inserção de PINs, que, tanto por razões de segurança e comodidade do utilizador, pode ser um futuro acesso ao sistema.

Os seguintes passos foram a consideração de pontos chave da arquitetura do sistema e breve implementação de bibliotecas de mensagens. Uma destas que já estava prevista, a que utiliza o *RabbitMQ* para abstrair a sua configuração e outra que surgiu na sequência desta e que é uma biblioteca de manipulação de mensagens. Esta última surge na sequência de tentar reduzir o tempo de implementação e possíveis erros de comunicação de cada micro serviço e, também, para manter as mensagens o mais consistentes possível. Desta forma a implementação de micro serviços tornou-se relativamente mais rápida.

Para a criação da biblioteca de processamento de mensagens foi feito o planeamento do que estaria incluído numa mensagem. O estagiário chegou ao resultado que se manteve até ao final do estágio. A estrutura foi um campo *Address* para identificação da mensagem e do micro serviços pelo qual passou, um campo *Processing* que tem as informações a ser processadas de um ponto anterior e um campo *History* para guardar informações que tenham sido utilizadas anteriormente para, caso exista um erro, saber-se que informações contribuíram para a má execução do micro serviço. Este campo também tem como objetivo ser utilizado como um conjunto de informações anteriores de *queries* para evitar o uso da base de dados mais vezes que o necessário. Posteriormente foi adicionado um campo *CreationTime* que guarda a data que esta mensagem foi criada para, caso exista um erro, ser mais fácil a pesquisa do mesmo. A figura 6.3 representa um objeto JSON, esta foi a primeira idealização por parte do estagiário.

Foi feita a arquitetura e versão base do que seria o processamento de mensagens por *RabbitMQ*. Isto dividiu-se em 3 classes, um *QueueHandler*, que cria a conexão com o serviço do *RabbitMQ*, um *Publisher*, que contém a lógica para envio de mensagens e, finalmente, o *Worker* que contém a lógica para receber mensagens. Tanto o *Publisher* quanto o *Worker* estendem o *QueueHandler* para evitar a duplicação de código em ambas as classes para a lógica de conexão a uma *Queue*.

A biblioteca de manipulação de mensagens foi criada para lidar com dados de mensagens não específicos. Isto significa a criação de uma nova mensagem, edição dos campos *History*

Figura 6.3: Ficheiro JSON com um exemplo inicial de uma mensagem.

e *Processing*, assinatura da mensagem, *timestamps* de data/hora e conversões de JSON para texto e de texto para JSON. Isto é útil para abstração e simplificação do código final dos micro serviços. Uma simples chamada para uma função que cria a estrutura da mensagem não torna apenas o código mais limpo, mas também garante a estrutura da mensagem. Ao longo do desenvolvimento, esta classe foi expandida para manter as interações com mensagens centralizadas e, em consequência, coerentes com o que é esperado por cada micro serviço.

Para complementar as mensagens e guardar um histórico destas, inicialmente esperava-se utilizar uma base de dados separada para guardar *logs* do sistema. Esta base de dados está representada na figura 6.4. Este sistema guardaria todas as mensagens que passassem pelo sistema, sem exceção, sendo que mensagens de erro seriam registadas na tabela *Error* e mensagens que fossem de processamento normal do sistema seriam registadas em *Message*. A tabela *Service* seria apenas para o registo dos micro serviços para que fosse possível confirmar nomes no sistema.

Este sistema não foi possível implementar e também foi determinado como demasiado pesado para o protótipo. Concluiu-se que isto se deve ao facto de que guardar todas as mensagens é um ponto redundante em termos de processamento de informação. Assim as únicas mensagens que será sempre relevante guardar são as mensagens que resultam em erro por um dos micro serviços.

Também foi levantada a questão de que não seria uma base de dados separada e que seria incluída na base de dados já mencionada. Como tal poderia ser um problema no futuro por adição de demasiados registos em paralelo com consultas a esta. Assim o sistema foi simpli-

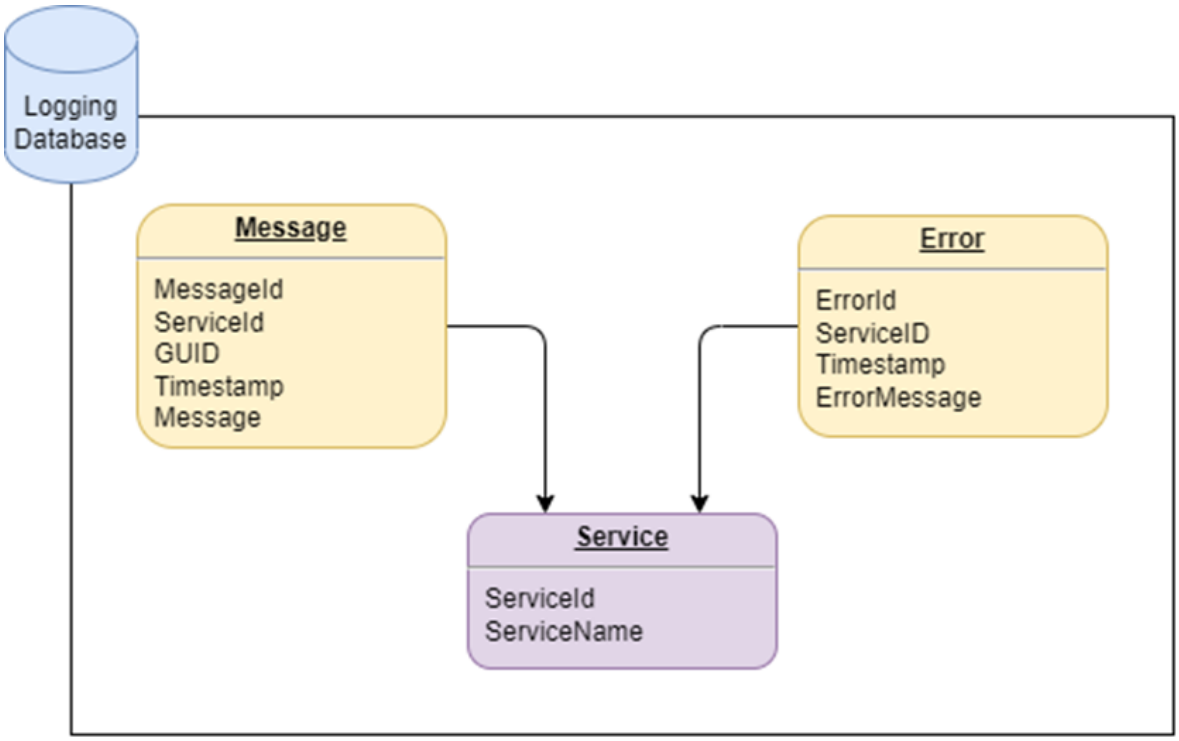

Figura 6.4: Diagrama a alto nível de arquitetura inicial da Base de Dados para *Logging*.

ficado no final para apenas registar erros e ser apenas um ID e um campo de texto.

A estrutura dos micro serviços foi elaborada também nesta semana. Esta é composta por pasta com o nome *Worker* com uma lógica paralela das APIs criadas anteriormente. A aplicação começava com o agora chamado *WorkerStarter*, imitando o índice na frente da API, que inicia o micro serviço e inicializava as dependências e variáveis globais necessárias para execução do programa em PHP. Em seguida, o *WorkerStarter* inicia o *Worker*, que é responsável por ler as informações da fila que está atribuída ao micro serviço, que é então passada para a camada *Business* do micro serviço. A camada de negócios começa por validar as informações que foram recebidas pelo serviço e a aceitação ou rejeição da mensagem é feita neste momento pelo conteúdo. Em seguida, processa as informações fornecidas com a ajuda da API que interage com a base de dados, caso seja necessário. Depois este micro serviço cria a nova mensagem e envia-a para a próxima fila ou *Queue* como referido. A lógica de envio de uma mensagem para uma *Queue* é separada, de forma semelhante às interações com a base de dados, por bibliotecas mencionadas anteriormente. As *Queues* atribuídas ao micro serviço são pelo menos 2: a *Queue* de erros para *Logging* e a *Queue* para passar as informações para o próximo serviço. Esta estrutura divide o processamento de maneira a promover clareza. A estrutura de ficheiros escolhida também é uma forma de manter a sua organização semelhante às APIs, isto para facilitar a manutenção após o projeto ser entregue à empresa, que o irá executar.

Esta estrutura manteve-se quando foi transitada para a C# pelas mesmas razões mencionadas. O processo de passagem de conhecimento inclusive para adicionar pessoas à manutenção do projeto seria inclusive feita de forma mais eficaz.

Para facilitar a implementação da comunicação de micro serviços com a API de base de dados,

existe a necessidade de um cliente HTTP que compõe o código para tal comunicação. Este cliente de comunicação implementa a lógica para comunicar com a API, ou seja, manipulação de URIs e criação da *query* associada a esta consulta. Isto limita o código duplicado e melhora a legibilidade da solução. A arquitetura foi criada para espelhar os *endpoints* que se destinam à API de base de dados.

Foram encontrados alguns problemas para a criação desta biblioteca, como: quais são estes *endpoints*? Quantos existem? E qual é o seu URI? Portanto revelou-se necessária a implementação da API, para responder a estas questões e implementar esta biblioteca para os micro serviços. Pelo que estas foram as principais tarefas desenvolvidas ao longo da restante semana.

A terceira semana do estágio foi uma continuação da implementação da API de base de dados. Sendo que isto foi concluído relativamente rápido e expôs o facto de ser necessária a criação da base de dados de forma local. Esta foi criada no início desta semana em *MySQL* pela simplicidade de criação desta por meio do *MySQL Workbench*. Após a instalação deste *software* e do servidor da mesma tecnologia, a tarefa seguinte foi criar a base de dados e alimentar alguns dados para testes de acesso. O *software* depois de algumas configurações gera um ficheiro SQL que descreve a criação e o preenchimento de dados. A explicação da base de dados já foi feita anteriormente, se necessário, recorrer a essa secção para uma melhor interpretação desta.

Depois disto foi iniciada a implementação da API *Orchestrator*. Isto foi relativamente simples, por esta API ser mais simples que a API de base de dados, com apenas 2 *controllers* e um *endpoint* para cada um dos *pipelines*. O foco aqui foi criar a mesma estrutura de camadas. Isto é relativamente simples de fazer, sendo a única diferença entre as duas APIs foi a camada de dados, sendo na API *Orchestrator* feito código de interação com *Queues* em vez de interações com a base de dados.

Também foram feitos testes das bibliotecas já implementadas através da instalação de um *software* específico de PHP chamado Composer. Isto permitiu testar a implementação com *RabbitMQ*. Testes de código criado anteriormente também foram bem-sucedidos, o que demonstrou que todos os serviços que usam mensagens já podiam ser implementados. Como grande parte do trabalho de interação com a base de dados já estava concluído, o próximo passo seria criar o cliente HTTP que comunica com a API de base de dados, o que significava utilizar *Guzzle*, biblioteca de PHP para esta aplicação, para lidar com este tipo de comunicação. Também foi feita implementação da biblioteca *MeekroDB* na API de base de dados, pela implementação completa de interação com esta em vez de *queries* programadas diretamente que podiam ter problemas de segurança. Para terminar o desenvolvimento das bibliotecas foi adicionada documentação diretamente no código, para que a função de cada método das 3 bibliotecas fosse explicada. Tudo isto foi concluído no decorrer desta semana.

A novidade no final da semana foi o uso do Azure como repositório *Git*. A plataforma é familiar para o estagiário, sendo que a configuração no computador pessoal deste repositório foi relativamente fácil já com a integração com o IDE de escolha, o *Visual Studio Code*. Surgiram alguns problemas devido à falta de acesso para criar repositórios, mas o supervisor de estágio por parte da Altice, resolveu este problema sendo que é o administrador deste con-

junto de repositórios. Ficou acordado que as metodologias de trabalho com esta plataforma seriam discutidas posteriormente.

No final desta semana também houve uma reunião acerca do estado do projeto com o supervisor por parte da Altice. Esta reunião teve como objetivo fazer um relatório de progresso e discutir detalhes do projeto. Após um breve relatório de progresso na programação da solução que estava a ser criada foi dado algum *feedback* nomeadamente a necessidade de uma tabela de perfis, no futuro a tabela *Role*. Esta tabela guarda possíveis perfis de autorização dos utilizadores. Esses perfis são principalmente para designar pessoas para ver informações como solicitações de *stock* e afins. A linguagem de programação também foi discutida. Sendo que esta foi a primeira menção de poder ser utilizada a linguagem de C#. Isto muda o paradigma em termos de experiência do estagiário na tecnologia utilizada, uma vez que ele possui experiência profissional nessa linguagem. A possibilidade de alteração da linguagem de programação revela-se benéfica para acelerar o ciclo de desenvolvimento. A principal vantagem desta mudança é a segurança que a linguagem tem em relação a PHP, principalmente variáveis estáticas na execução, como por exemplo passwords, nomes de utilizador, IP's para acesso a servidor que contêm a base de dados.

Na reunião foi ainda discutida a possibilidade de mudança do ambiente de trabalho do Data Center da Covilhã para o edifício MEO Torres Novas – Edifício Altice, mas num estilo mais remoto.

A quarta semana começou com a revisão do código feita na semana anterior, muito ao de leve mas sem grandes avanços uma vez que estávamos a aguardar que fosse decidido se era para manter PHP ou mudar para C#. Sendo que a alteração se veio a concretizar, como referido anteriormente em catorze de março

Apesar das vantagens, a mudança para a nova linguagem de programação é um retrocesso no tempo de desenvolvimento, já que toda a programação feita até então é excluída e tem de ser refeita na nova linguagem. Mas antes seria necessário fazer a instalação e configuração do ambiente de desenvolvimento para *.Net* 6. Foram então instaladas todas as ferramentas necessárias para tais desenvolvimentos. Esta é uma instalação bastante pesada, especialmente com as especificações que a máquina tinha na altura: 4 GB e processador de 2 cores. Isto para preparar o ambiente de desenvolvimento para a linguagem de programação fazendo com que qualquer desenvolvimento posterior da solução anterior, em PHP, fosse irrelevante.

Posteriormente, realizou-se uma reunião com a empresa parceira do projeto. Esta reunião teve como objetivo apresentar uma visão da solução que poderia ser dada ao estagiário como base para criar o que viria a ser o *Smart Locker*. A empresa disponibilizaria um conjunto de cacifos totalmente montado com comunicação através de uma API para os seus vários compartimentos, esta solução retiraria a necessidade do estagiário desenvolver qualquer solução neste sentido. Da documentação disponibilizada pela empresa especialista em *smart lockers*, foi criado um documento da sua análise. Isto iria determinar quais seriam os próximos passos de implementação da comunicação com uma solução física do *backend* que foi planeado. Para ilustrar os *endpoints* desejados e um reunir de todas as questões que surgiram dessa análise, foi criado um documento para facilitar essa comunicação. Este documento foi partilhado com o supervisor de estágio para análise.

Desenvolver este projeto com uma empresa especialista na área, embora seja uma mais-valia pode também representar um fator de dependência da solução face à empresa parceira. No entanto este assunto não constituiu entraves e foi aprovado pela Altice, ficando perspetivado que o desenvolvimento da solução seria base desta parceria. Outro destaque do encontro foi a existência de uma plataforma que simularia o comportamento dos cacifos que são disponibilizados, permitindo o desenvolvimento de uma solução sem que o cacifo físico esteja presente e só seja inserido posteriormente como prova de conceito.

Daí em diante começou a análise, documentação e criação da base da solução C# .NET. Neste caso, à semelhança da solução PHP o primeiro passo é a API da Base de Dados, pela mesma razão de interação com dados permanentes. Os pacotes *Nuget* que são usados para a comunicação com a base de dados *MySQL* são o *Entity Framework Core*, especificamente para *MySQL*, este *Nuget* abstrai consultas *SQL* e mantém o código em C# para fazer essas consultas, o que se revela uma vantagem para a posterior manutenção. Outro benefício do *Nuget* é o facto de poder construir a base de dados autonomamente através de apenas 2 comandos com todos os detalhes de como esta deve ser criada abstraindo envolvimento do programador. Este também pode modificar os ficheiros criados se for necessário e tiver as capacidades para fazê-lo. Após este trabalho todos os outros *endpoints* podem ser criados mais rapidamente, pois é um espelhar das conexões através das camadas de *Controller*, *Business* e *Database* do que já foi feito nesta tabela. Entretanto desenvolveram-se os *endpoints* desta API mas não foram testados a 100% nesta semana.

Para além do desenvolvimento, foi também feita uma apresentação sobre os progressos realizados numa reunião com o Engenheiro Valter, supervisor do estágio por parte da Altice. O progresso no desenvolvimento foi rápido e, portanto, foi estabelecido que seria utilizado um pseudo *scrum* como metodologia de trabalho. Os novos desenvolvimentos também serão hospedados num novo repositório. Foi também comunicada a aprovação da mudança do local de trabalho oficial para o estagiário, além de uma abordagem primariamente remota ao trabalho no projeto.

Ainda durante esta semana iniciou-se a criação das bibliotecas do projeto, à semelhança do que já tinha sido desenvolvido em PHP. Este desenvolvimento resultou na criação das bibliotecas *Message* e *Queueing*. Essas bibliotecas fazem exatamente o mesmo que suas contrapartes PHP. Embora a programação tenha uma abordagem diferente para a biblioteca *Message*, adotando mais uma abordagem de programação orientada a objetos, isso facilita a leitura do código, as únicas manipulações que estão disponíveis fora do objeto são as transformações de e para JSON, uma vez que fazem mais sentido nesta configuração. A função *ToString* foi alterada para gerar a representação em JSON da mensagem. Para a biblioteca *Queue* a principal prioridade foi fazer com que o fecho da ligação entre o *RabbitMQ* e o utilizador da biblioteca fosse não destrutivo, sendo que a biblioteca faz a gestão da ligação pelas boas práticas da documentação. Também foi implementada a integração do utilizador e password, esta ainda não havia sido desenvolvida na solução PHP. Esta última biblioteca usa o *Nuget* *RabbitMQ.Client* para as conexões.

Para complementar a mudança na linguagem, na quinta semana, foi criado um novo repositório e com isso a coordenação como projeto *git* com o novo repositório, e agora com *Visual*

Studio 2022 invés do *Visual Studio Code*. Este repositório contém todo o código que foi feito no contexto da solução, sendo então, um *hub* de todo o código.

No início da semana o estagiário começou a implementação do cliente da Base de Dados. Essa tarefa também serviu como forma de rever a API de base de dados. A biblioteca cliente foi concluída, mas não testada, o que ocorreu posteriormente em conjunto com a implementação dos micro serviços.

A implementação dos micro serviços terá a abstração dos detalhes de como criar mensagens, enviar estas mensagens para uma fila e como se comunicar com a Base de Dados. Essa base de abstração garante que a duplicação de código seja mínima e que o código seja mais fácil de corrigir.

Para criar a biblioteca foi necessária uma separação dos DTOs da camada de negócios da API. Isto foi feito para não conceder acesso ao código da camada de *Business* da API e apenas aos objetos que devem ser transmitidos por meio dela. Este foi um aspeto que foi corrigido e será utilizado nas próximas APIs que funcionam de forma semelhante; que requerem a comunicação ou retornam objetos que são específicos a estas.

A exceção à separação de DTO's da camada de *business* seria uma API de *Logging*, por esta consumir apenas dados e não retornar nada. Esta abordagem não foi utilizada, mas é uma possível evolução do sistema com a separação de *Logging* da base de dados do sistema.

A semana continuou com a implementação da API de *orchestration* tal como já planeado no sistema em PHP. Esta não teve modificações significativas desde a sua concepção e como tal foi rapidamente implementada, tendo em conta que tinha apenas dois *endpoints* para programar.

Assim que esta API foi completada iniciou-se a implementação dos micro serviços. Começando pela *pipeline Assign* sendo o primeiro micro serviço o *StockAssign* a ser criado e foi concluído até ao final desta semana. A Implementação deste micro serviço revelou a necessidade de alguns ajustes da API de base de dados.

Além desta modificação também foi feita uma melhoria da biblioteca *Queue* para fazer o consumo de mensagens de duas formas, sendo estas opções exclusivas uma da outra, no sentido em que a utilização de uma exclui a utilização da outra. Uma destas formas é um consumo ativo por chamada do sistema que contém as mensagens de forma periódica. A outra é utilização de eventos, sendo que a máquina que contém o sistema de mensagens envia um evento para o servidor que interpreta a mensagem associada a este. A maneira correta de implementar um micro serviço passa por utilizar esta última abordagem para manter a utilização de CPU o mais baixa possível. A possibilidade de uma espera ativa serve para algum caso que surja no futuro.

No início da sexta semana finalizou-se a implementação da *pipeline* de *Assign*. O *PinCreator* permitiu comprovar a velocidade com que seria possível criar um novo micro serviço de raiz para o sistema. Sendo necessário um dia para cada um, desde que não existissem problemas a resolver como novos pontos de acesso à base de dados.

Para facilitar esta implementação foi feita a estrutura dos micro serviços da *pipeline* de *Verification*. Esta opção foi uma tentativa de minimizar o tempo de criação de cada um dos micro serviços que compõem a *pipeline*. Isto pela abordagem de especialização como uma

linha de montagem. Esta abordagem reduziu o tempo de programação desta parte dos micro serviços.

Também foi feita uma análise das respostas que tinham sido enviadas pela empresa parceira sobre a documentação que foi disponibilizada. As respostas a alguns dos itens confirmaram o que já era esperado dos campos JSON. No entanto, outras respostas eram extremamente diferentes do que seria esperado. Estas respostas influenciaram a continuidade da implementação, sendo que mesmo assim foram propostas novas perguntas e sugestões.

Nos dias seguintes completou-se a implementação dos micro serviços. No entanto foi necessário complementar este trabalho com adaptações à base de dados e, consequentemente, à biblioteca cliente que trabalha com esta.

Com estes desenvolvimentos na implementação de tratamento de dados do sistema, na sétima semana começou-se a fazer a implementação de como interagir com os dados que se estiveram a preparar.

Assim, esta é a primeira semana de implementação da *Web App*. Para começar foi feita uma pesquisa de *frameworks* disponíveis. Uma destas foi o *Blazor* que é apropriado para este tipo de desenvolvimento, mas não era prático de utilizar com JWT. Esta havia sido a única hipótese tomada como um possível meio de desenvolvimento, mas foi descartado o desenvolvimento foi feito em MVC que é "*Model*, *View*, *Controller*".

Foi também criado o início das páginas que seriam utilizadas, modificando o *template* que o *Visual Studio 2022* já tem como base. Algumas modificações surgem com o intuito de evitar que exista poluição visual. Iniciou-se pelas páginas de cliente e posteriormente seguiram-se as páginas de administrador. Também começou aqui a pesquisa de como utilizar a autenticação em JWT nesta modalidade, especialmente como passar o *Token* que foi emitido pela *Web App* de retorno para confirmar que o pedido está autorizado. Isto tornou-se um desafio, principalmente para encontrar a melhor maneira de guardar esta informação, que é sensível e, caso seja captado por terceiros, pode originar a usurpação do perfil do utilizador e password pela duração do *Token*.

Finalmente foram feitas algumas adaptações à API de base de dados e respetivo cliente. Isto deveu-se ao facto de, inicialmente, algumas das *queries* necessárias à *Web App* não estarem previstas por lapso, sendo adicionadas nesta fase.

Esta oitava semana foi dedicada a uma revisão geral do funcionamento dos micro serviços para garantir que estes estavam a funcionar como pretendido. Para isto definiram-se e aplicaram-se baterias de testes para garantir o seu funcionamento. Também foram feitos *updates* dos *Nugets* das bibliotecas para que tenham a mesma versão que a *Web App*.

Foram realizadas também algumas modificações na Base de Dados. Sendo estas modificações a adição das tabelas *Locker* e *Role* e a alteração de pormenores de como é guardada a password do utilizador e adicionado um campo para identificar o seu *Role*. Foi feita ainda a associação da tabela *Locker* às tabelas de *stock* e pedidos de *stock*.

Contudo, estas modificações dos sistemas intrínsecos à base de dados implicam sempre a inevitabilidade de fazer atualizações à API e Cliente. Estas mudanças foram executadas de forma mais lenta devido à necessidade de relocalizar tudo o que é para modificar na API. Este processo ocupou o resto da semana.

Nesta nona semana foi disponibilizado o servidor que deveria ser utilizado para alojar a solução criada pelo estagiário. Com isto o estagiário foi informado do facto de que a base de dados deveria ser em *SQL server* e não em *MySQL*. Pelo que se iniciou esta mudança e instalação de duas ferramentas de interação com o servidor, o *FileZilla* e o *SQL Server Management Studio* (SSMS).

Tanto um como o outro foram configurados e instalados pelo estagiário, tendo sido necessária assistência com o *FileZilla*. Isto porque também existia um lapso no IP fornecido, mas rapidamente se resolveu a questão.

Esta modificação da base de dados criou a possibilidade de utilização do *Nuget Entity Framework Core* design que facilita a criação do *script SQL* de base de dados em *background*. Ou seja, faz uma abstração apenas pelo uso de anotações no código. A interpretação deste *Nuget* das classes que representam as tabelas fica registada num ficheiro ao criar uma migração.

Isto faz com que seja fácil a modificação da base de dados e esta estar ajustada ao código que foi feito. Isto por ser este código que cria a base de dados, sendo apenas necessário pensar uma vez na modificação e num único formato centralizado.

Com esta modificação teve de ser verificado se os *endpoints* estavam disponíveis da mesma forma. Para tal foram testados sucessivamente, tanto por chamadas da API como pelo uso do cliente da mesma. Foram também feitas algumas revisões do sistema em termos de possíveis casos de uso de informação.

Nesta semana também foi decidido que o sistema de *Logging* iria ser feito por um micro serviço. Um micro serviço será sempre mais escalável que uma API, pela facilidade de inicialização de vários em paralelo para os mesmos recursos computacionais que uma única API. Assim todos os outros micro serviços adicionaram uma conexão a este micro serviço e foram verificados para a utilização da nova atualização da API.

Também foi implementada a lógica de comunicação com micro serviços por parte da API *Orchestrator* que até agora não recebia informação por *Queues*. Com esta alteração o sistema tem as bases para fazer todo ciclo de processamento desde o pedido de um produto até à interação com o *hardware* que está associado a este projeto.

De seguida chegou-se à fase de rever como estava a autenticação da *Web App* e o que era necessário fazer para a completar. Foi reimplementado o JWT por forma a que este criasse um JWT válido, uma vez que na implementação anterior estava errada. Esta revelou-se uma das questões das que mais tempo absorveu na resolução de todo o sistema, a par da sua utilização em contexto da API.

O ciclo de reimplementações de JWT continuou na décima semana. Procedeu-se a uma sucessiva pesquisa e implementação, sendo que os recursos que aludem a como trabalhar com os *Nugets* e os *Nugets* em si não estão de acordo com o que são as normas de criação de um destes *tokens*, por exemplo como é o *header* de autorização do pedido HTTP. Isto agrava-se pelo facto de os *Nugets* que criam o JWT não serem os mesmos que os avaliam se este estiver válido ao nível do *middleware*.

Após as implementações que foram feitas na semana anterior, esta décima primeira semana foi dedicada à utilização do JWT já válido. Como a *Web API* é fulcral e está a haver muito

desenvolvimento que pode quebrar este equilíbrio de funcionar e não funcionar foi criado um novo projeto para testes para a implementação da sua autenticação. Sendo que esta depois inclui a implementação resultante.

Este trabalho continuou pelas duas semanas seguintes, sendo que este desenvolvimento também adicionou ao sistema a utilização de AJAX para a realização de pedidos com o *header* específico. Esta foi a única opção que foi possível realizar pela pesquisa feita de como adicionar código a páginas *Razor* de C#.

Nesta fase foi tomada a decisão de terminar este desenvolvimento e passar esta implementação com a utilização de *Partial Views* para um *div* da página para atualização da página em vez de uma *View* total. Esta opção deve-se à resposta da *query* de AJAX.

Também foi durante estas duas semanas que houve uma troca de vários *emails* com a empresa que seria parceira da Altice e que culminou com o cancelamento da parceria. Isto deveu-se ao facto de que não lhes seria possível disponibilizar nenhum componente do sistema de *hardware*.

Tendo em conta que esta parceria tinha já sido planeada desde a conceção foi inesperado que isto acontecesse. Além de que, nesta fase, cancelar a parceria torna praticamente impossível criar um protótipo físico para a implementação do sistema. Isto é algo que estava fora de controlo do estagiário e tendo em conta o tempo restante foi feito um agendamento de uma reunião informativa entre o estagiário e os orientadores para saber como se iria proceder.

Foi agendada uma reunião com o professor orientador da universidade de forma a ser elaborado um plano de execução deste componente do sistema. Sendo esta reunião também um modo de passagem de conhecimento de como concretizar este tipo de sistema.

Na décima quarta semana começou-se a elaborar a documentação para explicação do funcionamento do sistema tanto para o relatório de estágio como para a empresa.

Além disto foi feita a reunião com o professor orientador que deu várias sugestões que iriam necessitar *hardware* adicional àquilo que tinha sido fornecido até então. Neste sentido o plano de execução deste sistema teria de ser aprovado e o material providenciado por parte da Altice.

Estas informações foram fornecidas à Altice sendo que nunca foi dada uma resposta que indicasse se o recente plano estava aprovado ou não. Mesmo que tivesse sido aprovado, o material nunca chegou e, consequentemente, nunca foi executado.

Na décima quinta semana concluiu-se a elaboração da documentação acerca de como ficou desenhado o sistema e de como este funciona, incluindo a arquitetura do mesmo. Isto tanto para inclusão no relatório como para a manutenção/utilização do sistema por parte dos colaboradores da Altice. Esta informação foi enviada para os respetivos orientadores para aprovação do conteúdo.

Nas seguintes semanas foi feita uma memória descritiva acerca de como decorreu o estágio desde o início até ao seu término. Esta documentação pretende ser, tanto uma abordagem demonstrativa do percurso de estágio, como um complemento à interpretação de algumas razões que levaram à tomada de certas opções de implementação e, visa ainda, fazer um paralelo ao planeamento inicial que tinha sido feito pré-estágio com a realidade do trabalho realizado.

Capítulo 7

Conclusões e sugestões de Melhoria

7.1 Introdução

Este capítulo tem como objetivo fazer uma reflexão sobre os temas tratados no estágio e o seu planeamento e desenvolvimento. Estas conclusões são retiradas após o término do estágio e o culminar da informação do presente relatório. Começa-se por uma análise de melhorias e possíveis pontos que a solução final podia ter sido melhorada, seguido de uma análise do que se compreende como pontos essenciais do estágio.

7.2 Sugestões de melhoria

Em termos do que foi elaborado o ponto que se verifica como necessidade mais fulcral seria a integração de um complexo de cacifos no sistema com uma solução personalizada e realizada internamente na Altice. Isto exigiria principalmente tempo e recursos de hardware que não estavam disponíveis no momento de estágio. Sendo que o principal era a componente de tempo para a integração do sistema com o *backend*. Esta integração implicaria uma possível mudança de funcionamento dos serviços e dos dados guardados para otimização. Por exemplo, a separação do acesso para o hardware para que fosse possível levantar produtos mesmo que exista uma falha por parte do servidor central, dando autonomia aos cacifos por confirmação de informações de forma periódica com este. Isto seria feito pela utilização da memória de um computador de placa única, ou *single-board computer* (SBC), como o Arduíno e um meio de interação com o *smart locker* como um teclado numérico para inserir PINs.
Um segundo tópico de melhoria seria a criação de uma aplicação móvel híbrida, sendo a Web App mantida para continuar a garantir acesso a quem não tem acesso ao sistema via browser num computador. Nesta nota também seria benéfico para o utilizador possibilitar a receção de um PIN via SMS, para que fosse possível utilizar o sistema mesmo que não se tenha acesso à *internet*. Esta última interação teria de ser mediada por uma central de atendimento e, como tal, seria dispendioso em relação a um sistema automático. Este sistema também poderia ter a vantagem de um diagnóstico por parte da empresa antes de aprovar uma requisição de hardware.
Para concluir as melhorias do sistema, seria importante incluir uma pipeline para cancelamento de requisitos mais robusta. Como ficou realizado o sistema, a solução cancela um registo de requisição de um produto por dar esta por terminada. Esta abordagem não faz o reverter do processamento da pipeline de *Assign*, sendo que o inventário continua bloqueado para o utilizador assim como o cacifo do produto, sendo que deve ser feita uma recontagem dos produtos que estão disponíveis periodicamente além da manutenção habitual do sistema.

7.3 Conclusões

Para uma solução protótipo, considera-se que o sistema criado foi uma importante primeira tentativa, na qual foram encontrados alguns obstáculos como o cancelamento da parceria prevista com uma empresa externa e a questão do tempo que estava disponível para o estágio. A solução conseguida foi a aplicação de várias áreas do leque da Engenharia Informática, sendo que o projeto inclui a Engenharia de software, comunicação em rede, IoT, desenvolvimento Web e em geral a engenharia de uma solução desde a conceção da ideia até à sua realização.
O sistema poderia ser melhorado como mencionado na secção anterior, sendo que alguns dos pontos poderiam ter sido conseguidos, caso a parceria com a empresa especializada na área dos cacifos inteligentes tivesse sido concretizada e definida anteriormente ao estágio. Apesar disto, esta situação é um ponto de aprendizagem futura num regime profissional, apresentando a importância de um plano de ação caso este tipo de situação ocorra, sendo relevante ter preparada uma alternativa para estas circunstâncias de dependências externas.
Em termos de evolução pessoal também foi uma importante forma de aplicação de conhecimentos adquiridos em algumas das áreas estudadas no Mestrado em que se integra este estágio, nomeadamente IoT e como lidar com este tipo de sistemas. Mais áreas podiam ser incluídas para melhoria do sistema, no entanto revelam-se incomportáveis para criação deste tipo de protótipo de acordo com a conceção atual, falamos, por exemplo, de poder ser implementada inteligência artificial em vários pontos do sistema para melhoria da solução, tais como, a gestão de que equipamentos são necessários em cada região ou previsão de aprovação de um pedido de hardware consoante padrões de uso das várias regiões. Globalmente considera-se uma experiência positiva com identificação de alguns pontos a redefinir para a evolução a partir do estado a que se chegou nesta fase.

Bibliografia

[1] L. Faugere and B. Montreuil, "Hyperconnected city logistics: smart lockers terminals & last mile delivery networks," in *Proceedings of the 3rd International Physical Internet Conference, Atlanta, GA, USA*, vol. 29, 2016. 5

[2] Z. Xiao, J. J. Wang, J. Lenzer, and Y. Sun, "Understanding the diversity of final delivery solutions for online retailing: A case of shenzhen, china," *Transportation research procedia*, vol. 25, pp. 985–998, 2017. 5

[3] J. Sa-ngiampak, C. Hirankanokkul, Y. Sunthornyotin, J. Mingmongkolmitr, S. Thunprateep, N. Rojsrikul, T. Tantipiwatanaskul, K. Techapichetvanich, A. Pongsawang, T. Prayoonkittikul, U. Wattanakulchart, N. Prompoon, C. Ratanamahatana, and M. Pipattanasomporn, "Lockerswarm: An iot-based smart locker system with access sharing," in *2019 IEEE International Smart Cities Conference (ISC2)*, 2019, pp. 587–592. 5

[4] A. Narkhede, V. Mapari, and A. Karande, "Iot smart locker," in *Cybernetics, Cognition and Machine Learning Applications*, V. K. Gunjan, P. N. Suganthan, J. Haase, and A. Kumar, Eds. Singapore: Springer Singapore, 2021, pp. 1–7. 5

[5] R. Mash, C. Christian, and R. Chigwanda, "Alternative mechanisms for delivery of medication in south africa: A scoping review," *South African Family Practice*, vol. 63, no. 1, p. 8, 2021. [Online]. Available: https://safpj.co.za/index.php/safpj/article/view/5274 5

[6] T. Gundu, "Smart locker system acceptance for rural last-mile delivery," in *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2020, pp. 1–7. 6

[7] L. Faugère, S. S. Malladi, B. Montreuil, and C. C. W. III, "Smart locker based access hub network capacity deployment in hyperconnected parcel logistics," in *the Fifth International Physical Internet Conference*, 2018. 6

[8] K. F. Yuen, X. Wang, F. Ma, and Y. D. Wong, "The determinants of customers' intention to use smart lockers for last-mile deliveries," *Journal of Retailing and Consumer Services*, vol. 49, pp. 316–326, 2019. 6

[9] Y. M. Tang, K. Y. Chau, D. Xu, and X. Liu, "Consumer perceptions to support iot based smart parcel locker logistics in china," *Journal of Retailing and Consumer Services*, vol. 62, p. 102659, 2021. 6

[10] Y.-T. Tsai and P. Tiwasing, "Customers' intention to adopt smart lockers in last-mile delivery service: A multi-theory perspective," *Journal of Retailing and Consumer Services*, vol. 61, p. 102514, 2021. 6

[11] A. Alia and M. A. Rafiqb, "The determinants of customers' intention to use smart lockers for last-mile deliveries: A case of pakistan." 6

[12] T. T. Huong and B. N. Thiet, "Smart locker-a sustainable urban last-mile delivery solution: Benefits and challenges in implementing in vietnam." 6

[13] C. Malyack and P. Egbelu, "Understanding smart locker user behavior through twitter," in *14th International Conference on ICT, Society, and Human Beings, ICT 2021, 18th International Conference on Web Based Communities and Social Media, WBC 2021 and 13th International Conference on e-Health, EH 2021-Held at the 15th Multi-Conference on Computer Science and Information Systems, MCCSIS 2021*. IADIS, 2021, pp. 110–117. 6

[14] A. Huang and T. Chen, "Research on express smart locker location considering customer satisfaction under e-commerce environment," 2016. 6

[15] R. Villa and A. Monzón, "A metro-based system as sustainable alternative for urban logistics in the era of e-commerce," *Sustainability*, vol. 13, no. 8, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/8/4479 6

[16] I. Karakikes and E. Nathanail, "Assessing the impacts of crowdshipping using public transport: A case study in a middle-sized greek city," *Future Transportation*, vol. 2, no. 1, pp. 55–83, 2022. [Online]. Available: https://www.mdpi.com/2673-7590/2/1/4 6

[17] V. Kiousis, E. Nathanail, and I. Karakikes, "Assessing traffic and environmental impacts of smart lockers logistics measure in a medium-sized municipality of athens," in *The 4th conference on sustainable urban mobility*. Springer, 2018, pp. 614–621. 6

[18] T. Refaningati, S. Tangkudung, A. Kusuma *et al.*, "Analysis of characteristics and efficiency of smart locker system (case study: Jabodetabek)," 2020. 6

[19] R. Veerdonk, "Hergebruik van de jaloeziekast tot lockerkast: het is tijd om het gebruik van de jaloeziekast om te gooien," B.S. thesis, University of Twente, 2019. 6

[20] L. Faugere and B. Montreuil, "Hyperconnected pickup & delivery locker networks," in *Proceedings of the 4th International Physical Internet Conference*, vol. 6, 2017, pp. 1–14. 7

[21] ——, "Smart locker bank design: A scenario based optimization approach," in *Actes du Congrès International de Génie Industriel (Proceedings of Industrial Engineering Congress)*, 2017. 7

[22] J.-W. Lian, C.-T. Chen, L.-F. Shen, and H.-M. Chen, "Understanding user acceptance of blockchain-based smart locker," *The Electronic Library*, 2020. 8

[23] S.-T. Wang, M.-H. Li, and C.-C. Lien, "Analysis of the optimal application of blockchain-based smart lockers in the logistics industry based on ffd-saga

and grey decision-making," *Symmetry*, vol. 13, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2073-8994/13/2/329 8

[24] S. Paavolainen and A. Antonino, "Demo: Secure marketplace for access to ubiquitous goods," in *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021, pp. 1–2. 8

[25] "Franchising fnac um conceito Único," https://static.fnac-static.com/multimedia/PT/PUB/Guides/Franchising/FNACFRANCHISING-BROCHURE-19.pdf, accessed: 2022-01-31. 9

[26] "Os cacifos ctt agora são locky," https://www.ctt.pt/particulares/receber/locais-de-entrega-alternativos/cacifos-24h/index, accessed: 2022-06-27. 10

[27] "Site official da lokk," https://www.lokk.me/, accessed: 2022-01-31. 10

[28] "What is a ups access point locker?" https://www.youtube.com/watch?v=ONwWXAFPyiw, accessed: 2022-01-31. 11

[29] "Amazon hub," https://www.amazon.com/b?ie=UTF8&node=13853235011, accessed: 2022-06-27. 11

[30] "Smart lockers," https://www.ricoh-europe.com/business-services/all-services/service-advantage/smart-lockers/, accessed: 2022-01-31. 11

[31] "Secure transfer of goods through contactless processes," https://www.locktec.com/en/, accessed: 2022-01-31. 11

[32] "Truly smart lockers," https://bradfordsystems.com/smart-lockers/, accessed: 2022-01-31. 11

[33] "Welcome to velocity smart lockers," https://www.velocity-smart.com/smart-lockers, accessed: 2022-01-31. 11

[34] "Meridian's smart lockers," https://www.meridiankiosks.com/solutions/automated-smart-locker-system/, accessed: 2022-01-31. 11