
JOÃO BERNARDO LADISLAU DE FREITAS

# Aplicação móvel para leitura e validação de códigos de barra da DHL Express

Relatório de Projeto do Mestrado em Engenharia de Software

**ORIENTADOR**

Professor Adjunto, Nuno Miguel Vicente de Pina Gonçalves

**COORIENTADOR**

Assistente Convidado, Filipe Alexandre da Silva Mariano

Fevereiro 2021

IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

JOÃO BERNARDO LADISLAU DE FREITAS

# Aplicação móvel para leitura e validação de códigos de barra de cartas de porte da DHL Express

**JÚRI**

*Presidente:* Doutor, Cláudio Miguel Garcia Loureiros dos Santos, Instituto Politécnico de Setúbal

*Orientador:* Professor Adjunto, Nuno Miguel Vicente de Pina Gonçalves, Instituto Politécnico de Setúbal

*Coorientador:* Assistente Convidado, Filipe Alexandre da Silva Mariano, Instituto Politécnico de Setúbal

*Arguente: Professor Adjunto, José António Sena Pereira, Instituto Politécnico de Setúbal*

Fevereiro 2021

# Agradecimentos

Primeiramente, gostaria de apresentar os meus sinceros agradecimentos à Escola Superior de Tecnologia do Instituto Politécnico de Setúbal, em particular ao corpo Docente que constitui o Mestrado em Engenharia de Software, que contribuiu para o enriquecimento do meu conhecimento.

Apresento um especial agradecimento aos Docentes Joaquim Filipe e Cláudio Sapateiro por gerirem de forma eficiente todo o processo curricular, bem como providenciarem um vasto conhecimento na área de programação e gestão de projetos.

Também agradeço ao Docente Nuno Pina por se apresentar como meu orientador de estágio; ao Docente Filipe Mariano por se apresentar como meu coorientador de estágio e ao Engenheiro Luís Silva por se apresentar como meu supervisor de projeto e amigo, na DHL Express. O Eng.º Luís Silva desempenhou um papel fulcral neste projeto, tanto no desenvolvimento do mesmo, como na visão de negócio necessária para que o mesmo suprimisse as falhas de processo atuais.

Agradeço a todos os meus colegas da DHL Express Portugal, que contribuíram de uma forma ou de outra para o meu sucesso no projeto que desenvolvi, bem como a disponibilidade para esclarecimento de dúvidas e material fornecido para desenvolvimento de competências.

Um especial agradecimento aos meus pais, bem como toda a minha família, pelo apoio e condições necessárias para que o término do meu Mestrado fosse possível.

Especial agradecimento à minha namorada Filipa, por toda a ajuda, suporte, carinho e muita paciência para me ajudar a ultrapassar todos os obstáculos que surgiram durante o decorrer deste projeto.

Por fim, gostaria de agradecer aos meus colegas e amigos, que conheci durante o decorrer desta etapa, por toda a entreajuda e apoio que me ofereceram

# Resumo

O presente documento ilustra toda a análise, estudo e implementação de uma solução criada para melhorar um processo já existente numa empresa de logística de renome da atualidade.

Neste documento é feita uma contextualização da situação atual e como essa falha de processo afeta a eficiência da empresa durante a sua operação diária. Estas falhas ocorrem devido a malformações presentes em códigos de barras, existentes nas cartas de porte utilizadas para efetuar um envio. Com o crescente número de vias de integração disponibilizadas pela DHL Express, diferentes clientes ganham autonomia na geração das suas próprias cartas de porte. No entanto, como um sistema livre de utilização, o mesmo está sujeito a falhas. Recorrentemente, cartas de porte com códigos de barras inválidos são gerados e recolhidos. Estes envios, ao chegaram aos terminais de processamento, são rejeitados pelos processos automatizados, sendo, assim, processados manualmente.

A solução apresentada, desenvolvida em Android, seguinte uma visão Lean com especial incidência na metodologia Kanban, visa colmatar este problema na origem: no momento de criação da carta de porte de envio.

Toda a gestão e manipulação dos dados lidos por esta aplicação são feitos com recurso a uma base de dados disponibilizada pelo Google – O Firebase.

**Palavras-chave:** Carta de porte; Código de barras; Android; Lean; Kanban; Firebase.

# Abstract

The following document illustrates the analysis, study and implementation of a solution created to improve an existing process in a renowned logistics company.

This document provides a contextualization of the current situation and how this process failure affects the company's efficiency during its daily operation. These failures occur due to malformations present in bar codes, existing on the waybills used to make a shipment. With the growing number of integrations made available by DHL Express, different customers gain autonomy in the generation of their own waybills. However, as a free-to-use system, it is subject to failure. Recurrently, waybills with invalid bar codes are generated and collected. These shipments, when they arrive at the processing terminals, are rejected by the automated processes, being processed manually.

The solution presented, developed on Android, following a Lean vision with a special focus on the Kanban methodology, aims to address this problem at the origin: when creating the waybill.

All management and manipulation of the data read by this application are done using a database made available by Google – Firebase.

**Keywords:** Android; *Optical Image Recognition;* Java; XML; Firebase; Airwaybill; Bar code.

# ÍNDICE

# Lista de Figuras

1

# 1. Introdução

## 1.1 Contextualização

O presente documento ilustra toda a investigação, análise, planeamento e implementação de uma aplicação Móvel no âmbito de projeto final do curso do Mestrado em Engenharia de Software pelo Instituto Politécnico de Setúbal.

A necessidade deste desenvolvimento surge da necessidade de otimizar um processo de uma empresa mundial de renome da atualidade – A DHL Express. Atualmente, esta falha de processo gera quebras de desempenho e de rendimento, num mercado onde a rapidez é o foco de todo o negócio. Para contextualizar a problemática inerente a esse processo, é necessário compreendermos alguns conceitos básicos que serão sem dúvida essenciais para a compreensão deste documento.

Todos os envios da DHL Express são feitos com recurso a uma carta de porte. Mas o que significa carta de porte? Uma carta de porte, também conhecida como etiqueta ou *waybill* é um documento que tanto pode ser impresso numa folha A4 comum, em qualquer impressora, ou numa etiqueta autocolante impressa em "etiquetadoras" próprias para esse efeito. Uma carta de porte é um documento que contém três secções bem definidas, todas de igual importância, que permite a um individuo efetuar uma expedição pela DHL Express.

A primeira secção é a zona onde estão disponíveis os dados de expedidor e de destinatário. Os dados de morada, contacto, código-postal, cidade e país estão expostos nesta região. Para efeitos de privacidade, o contacto do expedidor e do destinatário são omitidos na carta de porte impressa, estando, no entanto, disponíveis nos sistemas informáticos da DHL Express. Esta zona contém também o código do aeroporto de origem da mercadoria. A segunda secção contém três códigos fundamentais para o despacho da mercadoria, que são o país de destino, o aeroporto de destino e o armazém de destino. Finalmente, a terceira secção é a zona onde estão

apresentados os códigos de barras, que serão lidos durante a expedição para guiar as encomendas para os seus destinos. Ao todo são apresentados três códigos de barras, sendo o primeiro é referente ao número da carta de porte, o segundo uma representação do código de país-aeroporto-armazém e o último representando o número de peça. É sobre esta secção, mais concretamente sobre o primeiro código de barras, que este estudo será feito.

Posto isto, estamos agora munidos da informação necessária para se apresentar sucintamente o Estado da Arte. A DHL Express é uma empresa multinacional, líder de mercado na Europa. Para atingir este estatuto de soberania perante toda a sua concorrência é necessário oferecer a todos os clientes um serviço de excelência e as ferramentas necessárias que os clientes necessitem nas suas operações diárias, ao nível de negócio e logística. Através de um vasto leque de opções e ferramentas, a DHL Express oferece ao cliente uma escolha de integrações a seguir com a empresa. Quando um cliente desenvolve uma integração com a DHL Express, mais dificilmente irá mudar para a concorrência. Ou seja, estas integrações representam quase que uma "garantia" que o cliente continuará a trabalhar com a empresa.

Estas soluções de integração estão disponíveis tanto numa vertente *online* como numa vertente *offline* e é concretamente nesta última vertente que as falhas podem surgir e, efetivamente, surgem. As ferramentas offline que a DHL Express oferece aos seus clientes, dão-lhes a total liberdade para criação das suas etiquetas de envio. Este sistema apresenta inúmeras vantagens operacionais, sendo que uma das que mais se destaca pela sua enorme importância será a rapidez e eficiência na geração e impressão de cartas de porte.

É muitíssimo importante frisar que tempo é efetivamente dinheiro e em negócios cuja exportação diária ultrapassa largamente mil envios, perdas de tempo na causadas por demoras na geração de cartas de porte são simplesmente inviáveis e inaceitáveis na ótica do cliente: desperdício de esforços; perdas de rendimento; má gestão do tempo – conceitos absolutamente defendidos pela metodologia *Kanban*, numa abordagem *Lean*, tal como apresentado sucintamente no capítulo três.

Tendo todos estes pontos em consideração, é então necessário perceber primeiramente como é que estes erros surgem. Ao dar ao cliente toda a responsabilidade em gerar a sua carta de porte, iremos sempre estar expostos a potenciais falhas no processo de geração do mesmo. Estas falhas podem vir simplesmente de uma má organização de dados de entrada – erros não graves - totalmente detetados nas fases iniciais de desenvolvimento ou erros de estrutura – graves - sendo estes, no conceito do *código de barras,* o foco deste projeto.

Neste sentido é fulcral perceber que proporções estes ditos códigos de barra devem seguir e quais são os seus tamanhos. Atualmente, apenas um código de barras é aceite pela DHL Express na geração das suas etiquetas. Existem projetos em desenvolvimento para novas soluções neste campo, no entanto, no imediato, apenas um é lido corretamente pelas máquinas

automatizadas nos terminais de expedição da DHL. Existem vários tipos de proporções para uma carta de porte da DHL Express, no entanto, as dimensões deste código de barras são um valor estático, não dinâmico. Ou seja, não é mutável conforme o tamanho da carta de porte em si.

Devemos agora focar-nos em como esta situação impacta diretamente com a atividade diária da DHL Express, no que toca ao processamento massivo de encomendas que chegam aos diferentes terminais mundiais. De forma a perceber como funciona o fluxo após chegada ao terminal, vamos imaginar uma simples caixa X. A caixa X é retirada de uma das diversas carrinhas operacionais e colocada na cinta de processamento. Após entrada na cinta, a caixa passa uma zona pré-determinada que é conhecida como "balança", cujo propósito é calcular o peso volumétrico da mercadoria, bem como efetuar automaticamente a leitura do código de barras presente na carta de porte. O objetivo nesta etapa é conduzir a encomenda para a sua cinta correta. É também aqui que surge o erro que procuramos solucionar. Vamos então apresentar o dito problema.

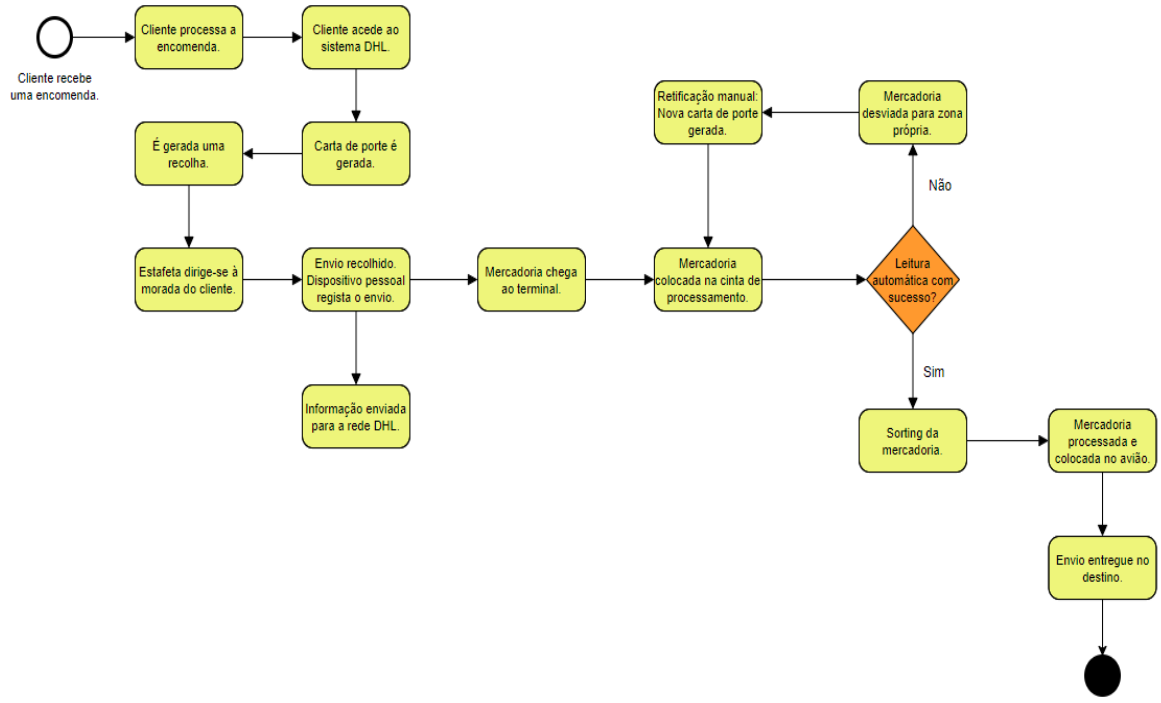


Figura 1 - *Business Process Diagram* (BPD) de uma encomenda DHL Express.

Quando um envio é detetado como inválido, após passagem na "balança", este é automaticamente desviado por uma cancela na cinta para uma área designada pela DHL como a *Hospital Area*. Essencialmente a *Hospital Area* é a zona para onde toda a mercadoria é desviada quando a sua leitura automatizada não é possível de se efetuar. Para estes envios

em concreto existe então a necessidade de intervenção manual por parte de um dos operadores: impressão de nova carta de porte, feita no momento, com um código de barras que efetivamente já esteja conforme os parâmetros esperados para uma leitura automatizada pela “balança”. Após esta retificação, o envio volta ao início do processo, para ser lido automaticamente.

É nesta fase que é muitíssimo importante frisar que esta situação é real, ocorre diariamente na operação da DHL Express. É necessário entender os atrasos ao nível de tempo e perdas de eficiência de processamento que estes erros geram, pois é uma situação que não ocorre para num mas em centenas ou milhares de envios por dia em toda a operação.

Por fim, é também importante notar que estas falhas não só afetam a empresa negativamente, como o cliente em si, pois em última análise este corre o risco de ficar com a sua mercadoria em terra por não existir tempo suficiente para a processar a mercadoria e como tal, a sua entrada no avião da DHL. Numa empresa onde o seu negócio é o transporte de mercadoria na vertente expresso, a não entrada da mercadoria num avião invalida uma entrega em 24h em qualquer ponto do mundo, perdendo-se assim a janela de oportunidade de concretização do cerne do negócio.

Tendo em conta todas estas afirmações, surge então a necessidade de blindar este processo através de uma solução nova, desenvolvida para esse efeito. É neste contexto que é feito um estudo em como solucionar esta questão e se chega à conclusão que seria necessária uma aplicação capaz de detetar esta falha na sua origem: quando o estafeta da DHL recolhe a mercadoria do cliente. Posto isto, surge então a oportunidade de desenvolver uma aplicação de raiz que teria de ser capaz de medir o código de barras presente na carta de porte e validar, através de medições com a câmera do telemóvel do estafeta, se a “balança” seria capaz de ler e processar corretamente esta mercadoria do cliente.

Esta aplicação deveria ser o mais simples e intuitiva possível, com o mínimo de menus possível, para que o tempo necessário para validar uma carta de porte no momento da recolha da mercadoria fosse quase instantâneo. A aplicação deveria, também, correr diretamente nos dispositivos que os estafetas da DHL dispusessem, de modo a que fosse possível reutilizar o material já existente, não sendo assim necessário a aquisição de dispositivos próprios para este efeito.

São então colocadas várias hipóteses sobre o tipo de aplicação que se deveria desenvolver para melhorar este processo. Seria uma aplicação *web* um bom investimento para este tema? Ou seria uma aplicação móvel uma melhor escolha? Podemos potencialmente descartar a hipótese de uma aplicação web, pois a solução a desenvolver teria de ser executada numa dos dispositivos do estafeta. Viramo-nos assim, um pouco mais para uma aplicação móvel. Ainda assim, dentro do tema das aplicação móveis, existem vários tipos de

linguagens de programação a utilizar, cada uma com as suas próprias vantagens. Seria pertinente desenvolver utilizado, por exemplo, React Native? Ou seria Android uma “aposta” mais segura?

No fim, ponderadas todas as possibilidades, a opção escolhida foi o desenvolvimento de uma aplicação móvel nativa, totalmente desenvolvida em Android com recurso ao Android Studio.

## 1.2 Estrutura

O presente documento encontra-se dividido em sete capítulos diferentes, cada um contendo informação relativamente a uma parte deste projeto.

O capítulo 1 é o capítulo introdutório, onde é feita uma apresentação do problema de modo a que seja possível transmitir a qualquer leitor deste documento qual o tópico que o mesmo retrata. Assim, este capítulo descreve o problema, o seu impacto empresarial, como contribui negativamente e o que é proposto que se faça para que o mesmo possa ser ultrapassado.

O capítulo 2 representa um capítulo onde se aborda o assunto na vertente da contextualização empresarial. É um capítulo pequeno e de rápida leitura que possui apenas a finalidade de dar a conhecer um pouco a empresa para o qual este projeto é desenvolvido.

O capítulo 3 é a secção do planeamento, da abordagem e da metodologia utilizada e seguida neste projeto. É aqui que são apresentadas as datas e as diferentes etapas delineadas para o desenvolvimento da aplicação em si. É neste capítulo que é feita uma explicação detalhada da metodologia seguida e como é que esta afetou o desenvolvimento do projeto positivamente: os conceitos, regras e fundamentos gerais desta visão de desenvolvimento. É nesta secção que são apresentados os conceitos de *Lean* e *Kanban*.

De seguida, o capítulo 4 é um capítulo mais técnico. Este é o capítulo onde são apresentadas as ferramentas e as tecnologias utilizadas no desenvolvimento desta aplicação. É aqui que é feita a apresentação do estudo de mercado do que já existe, na vertente de analisar que aplicações já existiam no mercado global e se poderíamos reutilizar alguma destas, em vez de se desenvolver um projeto de raiz. Neste capítulo é feita toda a apresentação da linguagem de programação Android, bem como todos os outros recursos - XML, Firebase, Trello, etc. - e como estes foram utilizados.

O capítulo 5 é um capítulo mais focado já na vertente do projeto em si, nomeadamente nos códigos de barras. Aqui é explicado efetivamente o que é um código de barras, quais as suas vantagens de utilização e para que serve. É feita uma contextualização no âmbito de

projeto no que toca às diferentes proporções e dimensões aceites pela DHL para leitura destes.

De seguida temos o capítulo da implementação – 6. É aqui que é apresentado todo o trabalho desenvolvido para que se concretizasse o idealizado. Este capítulo visa explicar passo a passo cada uma das abordagens, atividades e métodos programados para que os requisitos impostos pela DHL fossem cumpridos. Representa o *core* do presente documento, revelando todo o esforço e trabalho investido na aplicação.

Por fim, temos o capítulo das conclusões e do trabalho futuro. É nesta secção que são tiradas a conclusões do desenvolvimento desta aplicação e como ela irá contribuir positivamente para atividade da empresa e melhoramento de um processo já existente. No que toca ao trabalho futuro, é feita uma análise dos requisitos funcionais que não foram implementados nesta primeira versão do projeto, sendo que este tópico assenta exatamente nestes: requisitos funcionais não implementados que foram então migrados para uma versão 2.0 da aplicação.

Estes requisitos foram migrados para uma futura versão pois não representavam uma necessidade imediata da empresa mas sim uma expansão das funcionalidades da aplicação em si.

2

# 2. DHL Express Portugal

## 2.1 Introdução

Neste capítulo é apresentada a empresa DHL Express, para a qual este projeto é desenvolvido. A empresa é apresentada desde o seu nascimento, em 1969, bem como os seus fundadores, até o paradigma atual que se vive hoje. É feito uma pequena contextualização histórica do surgimento da DHL Express em Portugal e por fim apresento algumas estatísticas gerais da empresa, bem como as políticas e visões que a mesma possui para um futuro a médio-longo prazo.

## 2.2 A Empresa

Quando Adrian Dalsey, Larry Hillblom e Robert Lynn fundaram a DHL, em 1969, estes três empreendedores revolucionaram a indústria internacional de transporte expresso (Deutsche Post DHL Group, n.d.-b).

Em 1969, a DHL começou a operar o primeiro serviço internacional de entrega expressa porta-a-porta mundial (Deutsche Post DHL Group, n.d.-b).

Após 10 anos de atividade, em 1979, a DHL opera mais de 360 centros logísticos de distribuição, tendo em carteira mais de 85 mil clientes, que confiam no serviço expresso para colmatar as suas necessidades de negócio (Deutsche Post DHL Group, n.d.-b).

A DHL Express é a principal empresa internacional de logística e prestadora de serviço expresso da atualidade. A sua rede global abrange, neste momento, mais de 220 países e territórios (Coltman et al., 2010). A empresa disponibiliza o envio de embalagens ou documentos por diversos meios, nomeadamente por via terrestre, aérea ou marítima.

A família DHL é composta por seis unidades de negócio, altamente especializadas no seu ramo de operação, sendo que cada uma destas fornece uma variedade de produtos e soluções de logística para os mercados doméstico e internacional (Deutsche Post DHL Group, n.d.-a):

- DHL Parcel Germany
- DHL Express
- DHL Global Forwarding
- DHL Freight
- DHL Supply Chain
- DHL eCommerce Solutions


Figura 2 - Unidades de negócio da DHL. (Deutsche Post DHL Group, n.d.-a).

Nos últimos 30 anos, a DHL tem desfrutado de uma clara influência de mercado na região Ásia-Pacífico (Coltman et al., 2010). As perspetivas neste território permanecem altas devido ao crescimento contínuo do mercado *onshore* e *offshore,* baseados em elevados números do produto interno bruto (PIB) e melhores condições de negócio para empresas estrangeiras (Coltman et al., 2010).

## 2.3 DHL Express em Portugal

Em 1981, doze anos após a sua fundação, a DHL abre escritórios em Portugal, onde hoje está formalmente registada como DHL Express Portugal, Lda.

Em 2019, a DHL inaugura a sua primeira loja física, situada no Areeiro, em Lisboa, fruto da estratégia de expensão no mercado Português (Grande Consumo, 2019). Em suma, esta pequena loja visa aumentar a rapidez e conveniência dos serviços de transporte expresso nacional e internacional (Grande Consumo, 2019), dando uma maior flexibilidade aos clientes particulares na entrega e envio de mercadoria. Este investimento tem como objetivo acompanhar as crescentes necessidades dos consumidores Portugueses, assim como a fomentar o desenvolvimento do comércio online, que por sua vez, incentiva o negócio particular (Reis, 2019a).

Atualmente, em Portugal, a DHL Express opera com uma equipa de mais de 500 colaboradores e uma frota com cerca de 200 veículos operacionais e 2 aviões Boeing 747 dedicados, que garantem diariamente a ligação de Portugal, através do Aeroporto Humberto Delgado, em Lisboa e do Aeroporto Francisco Sá Carneiro, no Porto, com os grandes centros operacionais em todo o mundo (Marcela, 2019).

## 2.4 Planeamento Futuro

No que toca ao planeamento de um futuro melhor para todos nós, a DHL Express ampliou, recentemente, o comprometimento com a Sustentabilidade e os objetivos do Grupo DPDHL para, até 2050, atingir zero emissões.

Internamente, conhecido como GoGreen, este serviço é destinado a todos os clientes que pretendam um método fiável para neutralizar as emissões de carbono inerentes ao transporte do seu envio: queima de combustíveis fosseis por aviões, camiões, etc.

O cálculo das emissões é feito através de uma ferramenta patenteada, em função da origem e destino do envio, e a respetiva compensação é calculada através do investimento em programas ambientais já implementados, nomeadamente, na criação de redes de energia eólica e na reflorestação (Transportes, 2019). Este serviço está disponível para todos os clientes DHL Express, com ou sem acordo comercial com a empresa, para qualquer tipologia de envio com um custo adicional de 10 cêntimos por kg, que reverte na íntegra para os programas ambientais.

Sendo a DHL Express a maior empresa de transportes e logística a nível global, a DHL tem a responsabilidade acrescida de tornar o transporte mais sustentável, usar os recursos naturais de maneira mais eficiente e sensibilizar os seus clientes para a importância da proteção ambiental (Reis, 2019b).

3

# 3. Planeamento, Abordagem e Metodologia

## 3.1 Introdução

Este capítulo descreve a metodologia utilizada durante o desenvolver desta aplicação e todo o seu trabalho relacionado. É feita uma descrição detalhada do planeamento e planificação do trabalho a desenvolver, bem como as diferentes possíveis abordagens a seguir. Neste capítulo é também apresentada a linguagem de programação utilizada e como ela se distancia das outras, sendo a escolhida neste projeto.

## 3.2 Planeamento

Para melhor visualização do planeamento elaborado no desenvolvimento desta aplicação, foi criado o cronograma ilustrado na figura 3.

O projeto arrancou na segunda semana de Outubro com a Preparação para o desenvolvimento. Todo o material necessário foi reunido e devidamente organizado. O cerne desta primeira etapa foi criar as condições necessárias para que o projeto tivesse sucesso.

De seguida, foi abordada a etapa da Especificação, onde foram definidos os requisitos funcionais e os requisitos não funcionais da aplicação. Esta etapa visou definir o *core* do projeto, ou seja, planificar as capacidades da aplicação no sentido prático e na visão do utilizador. Estas duas fundamentais etapas decorreram durante o mês de Outubro.

Posteriormente, apresenta-se a etapa da Desenho. Nesta etapa são definidos os diferentes componentes do *software*, as diferentes interligações e interações entre si. Em suma, é neste capítulo que é definido como o sistema funciona e os diferentes sistemas com que a aplicação comunica. A fase de Desenho ficou agendada para o mês de Novembro, sendo a sua duração de cerca de 1 mês.

Seguidamente, a etapa do Desenvolvimento e Implementação da aplicação. Esta etapa consiste no desenvolvimento do código-fonte utilizado para a lógica, e a implementação de várias interfaces gráficas utilizadas para comunicar com o utilizador, neste caso. Esta secção será detalhada mais à frente, nos capítulos 4 e 6.

Esta etapa consiste em cerca de 7 meses, tendo início em Dezembro e fim em Julho colmatando com as duas últimas etapas – Testes e Documentação.

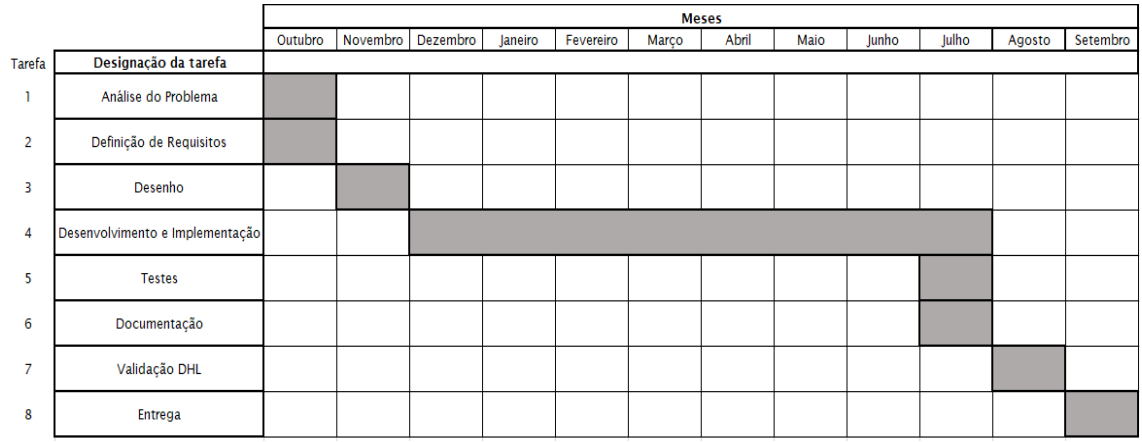


Figura 3 - Cronograma da aplicação.

## 3.3 Abordagem

Pondo em prática algum do conhecimento adquirido em várias unidades curriculares da Licenciatura, que frequentei em Engenharia Informática, o primeiro mapa mental feito para encarar o projeto e desafio em questão acaba por ser conceptualizar e idealizar uma abordagem *Lean* ao problema.

A aplicação em questão é desenvolvida única e exclusivamente pelo Autor da mesma, pelo que, após alguma análise e estudo das diferentes abordagens a seguir, a abordagem *Lean* revelou-se como sendo a escolha mais apropriada para este projeto.

O grande fator por detrás desta abordagem é o facto de seguir uma entrega continua de produto (Guévin, n.d.), seja este mais ou menos viável no ponto de vista das características e

funcionalidades já implementadas para o utilizador. Além desta grande vantagem, um dos principais requisitos que defini *à priori,* antes do avanço do desenvolvimento do projeto, foi eliminar todo e qualquer desperdício de esforço que fosse necessário para completar cada tarefa, sendo esta uma das principais "filosofias" defendida pela abordagem *Lean*.

## 3.4 Metodologia - Lean

Antes de efetivamente se apresentar a metodologia *Lean*, é importante que se perceba o porquê de esta ter sido a "visão" adotada. Tal como irei apresentar no desenvolvimento deste tópico, o *Lean* é uma metodologia orientada a processos, sendo exatamente isso que iremos melhorar com a implementação desta solução. Este projeto foi desenvolvido exclusivamente por um único programador, sendo esse outro dos principais motivos: o facto de não ser um projeto de equipa, mas sim uma quase como uma "tarefa" pessoal. É uma metodologia que surgiu em contexto operacional e industrial, ao contrário de *Scrum*, por exemplo, que surgiu em contextos de desenvolvimentos de software. Para finalizar esta justificação, é muitíssimo importante perceber que o foco de *Lean* é a eliminação de desperdício, sendo aqui aplicado a um processo empresarial.

A prática do desenvolvimento de software tem sido atormentada por taxas de sucesso surpreendentemente baixas há décadas. Ao mesmo tempo, o número de produtos e serviços baseados em software continua a aumentar drasticamente a cada ano (Curt Hibbs, 2009).

O desenvolvimento *Lean* é um paradigma de desenvolvimento de produtos com foco *end-to-end* na criação de valor para o cliente, eliminação de desperdícios, otimização de fluxos de valor, capacitação de pessoas e melhoria contínua (M. Poppendieck, 2010). Na sua essência, o desenvolvimento de Software *Lean* é a aplicação dos princípios do Sistema de Desenvolvimento de Produtos da Toyota ao desenvolvimento de software (M. Poppendieck, 2016).

Os princípios e a mentalidade desta metodologia provaram ser notavelmente aplicáveis à melhoria da produtividade e qualidade de praticamente qualquer empreendimento (Curt Hibbs, 2009).

A mentalidade *Lean* já é aplicada há bastante tempo em diversos setores económicos da nossa sociedade, por exemplo, na *Banca* ou nos diferentes canais de *Distribuição*, estando apenas há relativamente poucos anos a ser aplicada ao desenvolvimento de software. É uma mentalidade, uma maneira de pensar em como agregar valor ao cliente mais rapidamente, encontrando e eliminando desperdícios (Curt Hibbs, 2009). Ou seja, abolir todos e quaisquer impedimentos à qualidade e à produtividade.

De forma concisa, todas as análises ao desenvolvimento de software *Lean* seguem sete princípios chave, inicialmente apresentados por Mary e Tom Poppendieck, no seu livro *Lean Software Development: An Agile Toolkit for Software Development Managers, 2003:*

1. Eliminar o desperdício
2. Integrar qualidade
3. Criar conhecimento
4. Adiar compromissos
5. Entregar rapidamente
6. Respeitar as pessoas
7. Otimizar o todo

### 3.4.1 Princípio 1 - Eliminar o desperdício

A abordagem *Lean* tem uma visão bastante estrita no que toca ao desperdício. Shigeo Shingo, um dos responsáveis pelo desenvolvimento deste sistema na Toyota identificou sete tipos diferentes de desperdício.

- Defeitos - **D**
- Superprodução - **O**
- Transporte - **T**
- Tempo de espera - **W**
- Inventário - **I**
- Fluxo - **M**
- Processamento - **P**

Estes tópicos constituem os sete tipos de desperdícios letais para uma corporação: ***DOTWIMP***.

Para que seja efetivamente possível detalhar cada um destes tipos de desperdício, é necessário converter estes tópicos para a vertente de desenvolvimento de Software (Curt Hibbs, 2009), ou seja:

Defeitos → Defeitos;

Superprodução → *Features* extra;

Transporte → Entregas;

Tempo de espera → Atrasos;

Inventário → Trabalho parcialmente completo;

Fluxo → Sequência de tarefas;

Processamento → Processos desnecessários;

### 3.4.2 Princípio 2 - Integrar qualidade

Fazendo novamente uma pequena retrospetiva ao Sistema de Produção da Toyota, uma das suas abordagens à questão da qualidade refere que um produto não deveria ser inspecionado no final da linha de montagem. Essa mesma abordagem deteta possíveis falhas de fabrico e ou, qualidade, mas não tem qualquer impacto direto para o seu impedimento. Como alternativa, cada diferente fase do processo deve estar blindada ao máximo quanto a falhas para que as mesmas sejam detetadas no momento da sua criação.

De forma a otimizar o processo, e, como o próprio título do tópico afirma, para que possamos integrar qualidade junto deste, quando uma falha é detetada, todo o processo adjacente deve parar até que seja feita uma análise minuciosa à falha e se detete a sua causa.

Infelizmente, no que toca ao desenvolvimento de *software*, não é esta a visão que as equipas têm. Neste momento ainda permitimos que os defeitos passem livremente pelo *software* e sejam apenas detetados posteriormente pelas inspeções de controlo de qualidade (Curt Hibbs, 2009), ou seja, durante as sessões de testes ao produto.

A abordagem *Lean* defende que a lógica de código implementada deve ser à prova de erros e que esta deve ser testada à medida que vai sendo desenvolvida (Curt Hibbs, 2009). Ou seja, toda e qualquer funcionalidade desenvolvida deve possuir uma bateria de testes associada à mesma que corrobore o bom funcionamento da mesma com o *output* esperado. Desta forma, estes testes impedem que alterações subsequentes no código introduzam novos defeitos não detetados (Curt Hibbs, 2009).

### 3.4.3 Princípio 3 – Criar conhecimento

Cometer os mesmos erros recorrentemente ou reaprender como algo funciona representa uma perda de tempo e esforço para toda a equipa (Curt Hibbs, 2009). O grande cerne deste tópico, na abordagem *Lean*, é de não se reaprender o que já se sabe desde o início. Qualquer que seja o membro da equipa, se o conhecimento para um dado tema já existe, este não deve ser reaprendido. O conhecimento deve ser partilhado por todos os intervenientes para que se possam suprimir os desperdícios da força mais otimizada possível.

Para tal, devem ser encontradas maneiras de registar o conhecimento da sua equipa como um todo, para que, qualquer membro possa localizá-lo facilmente quando for necessário (Curt Hibbs, 2009).

No entanto, toda esta linha de pensamento e abordagem pode parecer uma pouco abstrata. Isto deve-se ao facto de ser difícil ser-se específico sobre algo, porque o que faz sentido e o que funciona para um determinado individuo depende muito do contexto em que este está envolvido e das suas ações necessárias.

No entanto, habitualmente é preferível manter um determinado conhecimento mais próximo da sua origem (Curt Hibbs, 2009). Ou seja, no contexto de um projeto de *software*: É preferível que o conhecimento necessário para produzir um correto modelo de arquitetura de uma aplicação esteja junto dos arquitetos de sistema e, porventura, junto dos analistas de sistema. Por outro lado, questões relacionadas com implementações lógicas de determinadas funcionalidades, devem ser alocadas o mais próximo possível da equipa de desenvolvimento (programadores).

### 3.4.4 Princípio 4 – Adiar compromissos

Frequentemente, as melhores decisões são tomadas no preciso momento em que mais e melhor informação está disponível (Curt Hibbs, 2009).

Assim, todas as decisões e todos os compromissos devem ser adiados o mais possível até que todo o leque de informação necessária para os tomar seja o mais coerente e correta possível. No entanto, analogamente, estas decisões também não devem ser o mais adiadas possível de forma a não impactarem com os restantes temas envolvidos no desenvolvimento de um projeto (Curt Hibbs, 2009).

Desta forma, as decisões irreversíveis devem ser adiadas até ao ultimo momento em que estas sejam prudentes (Curt Hibbs, 2009).

### 3.4.5 Princípio 5 – Entregar rapidamente

Apesar do desenvolvimento de *software* ser uma tarefa algo abstrata, todos nós trabalhamos mais eficientemente quando existe uma meta e um objetivo concreto a atingir (Curt Hibbs, 2009), seja ele através de representações esquemáticas de um sistema ou através de requisitos de funcionalidades.

No entanto, apesar disto, os objetivos são difíceis de atingir pois todas estas representações não são objetos concretos ou artigos palpáveis. É este o motivo para a constante mudança dos requisitos de um software, e, como tal, a sua volatilidade (Curt Hibbs, 2009).

*Entregar rapidamente* nada mais significa que projetar e desenvolver pequenas funcionalidades e entregá-las ao cliente o mais rapidamente possível, recursivamente. Ou seja, sempre que exista uma nova funcionalidade, esta deve imediatamente ser partilhada com o cliente para

que o mesmo possa dar o seu *feedback*, prevenindo assim, falhas no desenvolvimento do *software* ou alterações nos já existentes requisitos funcionais.

O término de cada pequena iteração oferece a oportunidade de alterar e privilegiar um ou outros requisitos com base em *feedback* (Curt Hibbs, 2009) e em cenários reais de teste pelo utilizador.

O produto final gerado será um produto mais próximo do que foi inicialmente imaginado pelo cliente. Por outra palavras, um produto que consiga colmatar as necessidades do mesmo enquanto simultaneamente elimina desperdícios no seu desenvolvimento sempre que existam alterações dos requisitos.

### 3.4.6 Princípio 6 – Respeitar as pessoas

Respeitar as pessoas significa depositar e confiar nestas para saber a melhor abordagem para executar o seu trabalho. Envolvê-las para evidenciar possíveis falhas no processo atual e incentivá-las a encontrar maneiras de aprimorar as suas incumbências e os procedimentos seguidos ou envolvidos (Curt Hibbs, 2009).

Os recursos não devem ser desperdiçados, os intervenientes de um projeto ou de uma empresa são os seus melhores ativos para o cumprimento de objetivo e o alcance de metas.

Em suma, o respeito pelas pessoas significa reconhecer as suas capacidades técnicas e os seus feitos, solicitando ativamente a estas a sua visão e opinião sobre dados temas (Curt Hibbs, 2009).

### 3.4.7 Princípio 7 – Otimizar o todo

Independentemente da área onde estamos a seguir uma abordagem *Lean*, este é um dos seus tópicos mais fulcrais.

Sempre que um processo local é otimizado, recorrentemente isso significa quebras no resto de fluxo de valor (Curt Hibbs, 2009).Se não se possui total controlo do fluxo de valor, os gestores podem ser forçados a otimizar apenas parte dele. Ou seja, devem ser feitos incrementos de otimização para que os processos a jusante não sofram qualquer impacto indevido e involuntário por este. No entanto, deve-se procurar abranger sempre o máximo possível deste fluxo.

## 3.5 *Kanban*

Perante um mercado em constante evolução e com os avanços tecnológicos diários, as empresas e organizações lutam freneticamente numa busca incansável para otimizar as suas estruturas, as suas estratégias de ataque a novas oportunidades, e até mesmo as suas próprias políticas internas em resposta à procura diária de soluções. Atualmente, os custos operacionais são reduzidos e o tempo de colocação no mercado é, também, reduzido. Resultando assim, num aumento da produtividade (Ahmad et al., 2013).

O *Kanban* é um sistema de gestão de tarefas bastante simples, no entanto, muito competente, para maximizar o fluxo e minimizar o trabalho em desenvolvimento (B. M. Poppendieck & Poppendieck, 2003). Deriva das palavras japonesas *kan* - cartão e *ban* - sinal.

Recentemente, tornou-se numa abordagem mais comum no desenvolvimento de *software* sendo a mais recente adição ao método ágil e *Lean* (Ahmad et al., 2013).

### 3.5.1 *Kanban* - Implementação

Numa perspetiva de desenvolvimento de *software*, um sistema *Kanban* é implementado com recurso a cartões coloridos colados num quadro branco.

Um quadro *Kanban* pode ter tantas etapas quanto as que seja idealizadas, no entanto, a título de exemplo, vamos imaginar uma abordagem simplificada: Desenho e Definição, Implementação, Testes e *Deploy* – ilustrado na figura 4.

De forma geral, cada um dos cartões, dentro de cada etapa, tem um tempo estimado de esforço necessário para conclusão. Como cada um destes cartões é atribuído a um determinado elemento da equipa de desenvolvimento, este tipo de quadro *Kanban* permite que qualquer colaborador veja e analise o estado atual de uma determinada funcionalidade (Curt Hibbs, 2009).

Os cartões *Kanban* informam os programadores acerca do que necessitam de criar e implementar, ou seja: o trabalho a realizar (B. M. Poppendieck & Poppendieck, 2003). Apesar disto, por norma estes cartões não são claros o suficiente para orientar cada um dos colaboradores no que desenvolver e que vias seguir, pelo que é habitual que existirem reuniões diárias de orientação (B. M. Poppendieck & Poppendieck, 2003).

É importante notar que o esforço necessário para definir e desenhar uma funcionalidade é habitualmente e substancialmente inferior ao tempo de a implementar. É por esta razão que podem surgir atrasos:, isto é, variações no tempo de esforço de cada tarefa. Estes atrasos são considerados *desperdício*, tal como abordado no tema anterior, que o *Lean* procura eliminar a todo o custo (Curt Hibbs, 2009). Uma possível solução para esta questão passará pela adição

de mais uma etapa, denominada, por exemplo, de *queue* (Curt Hibbs, 2009). Esta nova etapa nada mais representa que o *backlog* de etapas que existem para fazer.

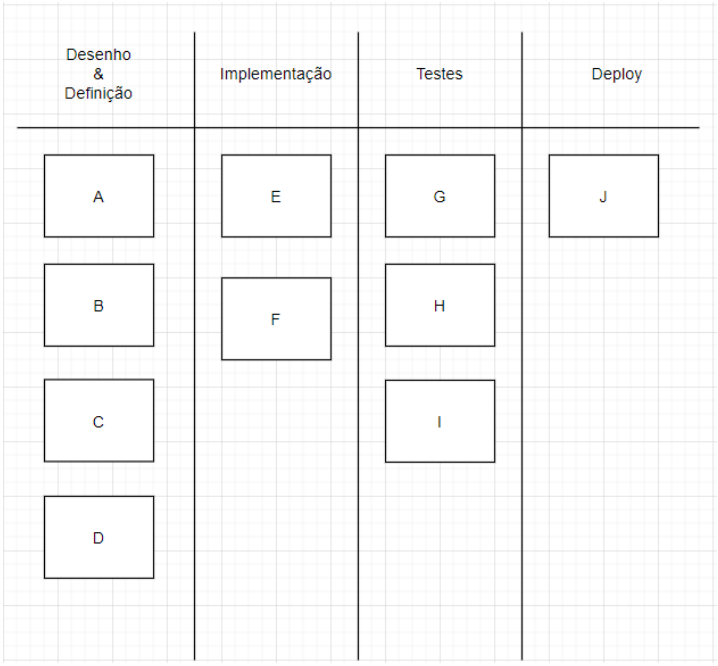


Figura 4 - Quadro *Kanban*.

4

# 4. Ferramentas e Tecnologias Utilizadas

## 4.1 Introdução

Neste capítulo abordo o ambiente de desenvolvimento utilizado no decorrer deste projeto: o *Android Studio*. É feito um breve resumo do surgimento desta ferramenta no mundo digital, os seus criadores, e as grandes vantagens que o mesmo traz para os seus utilizadores.

São abordadas as duas principais linguagens utilizadas neste projeto, nomeadamente Java, para a componente de lógica e XML para as *interfaces* gráficas que interagem com o utilizador.

## 4.2 Análise da Atualidade

É importante que se compreenda na totalidade a razão de não ter sido utilizada qualquer uma das aplicações já existentes no mercado global para suprimir esta necessidade da DHL.

Atualmente no mercado já existe um vasto leque de aplicações móveis que efetivamente são capazes de efetuar a leitura de um código de barras. Ainda assim, será que estas aplicações conseguem cumprir e colmatar as necessidades operacionais da DHL? É importante considerar que estas aplicações não foram desenvolvidas com o objetivo de solucionar o problema mencionado anteriormente. No subcapítulo seguinte são apresentadas algumas opções já existentes, bem como os fundamentos da sua possível utilização, ou não.

É também crucial nesta etapa compreender a escolha de linguagem de programação a utilizar. Das imensas opções disponíveis, quais seriam as que se enquadrariam melhor nesta tarefa? Quais seriam as mais fáceis de implementar dado as necessidades do "cliente"? Esta questão será esclarecida em detalhe na secção seguinte.

### 4.2.1 Aplicações já existentes no mercado

Visitando a *app store* da Google na presente data, 3 de outubro de 2020, e pesquisando por este tema, encontramos as três principais aplicações no mercado para esta funcionalidade, apesar de existirem muitas mais.

| Nome | Autor | Data de publicação | Downloads |
|---|---|---|---|
| **QR & Barcode Reader** | TeaCapps | Junho, 2016 | + 50M |
| **QR & Barcode Scanner** | Gamma Play | Março, 2015 | + 100M |
| **Barcode Scanner** | ZXing Team | Setembro, 2018 | + 100M |

Figura 5 - Aplicações atualmente no mercado.

Estas aplicações não são *open source*, pelo que não nos é possível estudar a lógica de funcionamento da aplicação em si. Ou seja, toda e qualquer análise da mesma terá de ser fruto da sua utilização direta. Todas elas, quando inicializadas, apresentam ao utilizador uma *interface* com os limites de onde a câmera do telemóvel em si, espera que seja detetado um código de barras.

Quando a câmera captura um código de barras nesta zona, automaticamente é feita esta leitura e apresentado o *output* numa nova janela para o utilizador. É possível assumir, assim, que todas estas aplicações possuem um método *onBarcodeDetected.* Quer isto dizer que a aplicação possui um "detetor" que assim que capta um código de barras, produz uma resposta em função deste.


Figura 6 - QR & Barcode Scanner. Autor: Gamma Play.

Este tipo de abordagem não permite solucionar a verificação e validação das dimensões de um código de barras (principal problema da DHL), apenas permite identificar se o mesmo é ou não um código de barras válido em termos de estrutura. É exatamente por esta questão que estas aplicações se excluem para os requisitos apresentados neste projeto, visto que o cerne desta aplicação é a medição do comprimento e da altura dos códigos de barras gerados pelas aplicações da DHL. O constrangimento atual surge da falta de aptidão das máquinas dos terminais da DHL de processarem encomendas com códigos de barras que não estejam mapeados nas mesmas.

Posto isto, está então confirmada a necessidade de se desenvolver e implementar uma solução, de raiz, que efetivamente tenha as competências necessárias para ultrapassar esta falha de processo.

| | Aplicações testadas | | |
|---|---|---|---|
| | QR & Barcode Reader | QR & Barcode Scanner | Barcode Scanner |
| A aplicação é Open-Source? | Não | Não | Não |
| É possivel efetuar login? | Não | Não | Não |
| Utiliza a câmera do dispositivo? | Sim | Sim | Sim |
| Mede os tamanhos dos códigos de barra? | Não | Não | Não |
| Lê códigos de barra? | Sim | Sim | Sim |
| Permite acessos a bases de dados externas? | Não | Não | Não |
| Apresenta o *preview* da leitura? | Não | Não | Não |
| Permite gerar emails com a informação lida? | Não | Não | Não |
| Podemos usar esta aplicação? | Não | Não | Não |

Figura 7 - Tabela comparativa das diferentes aplicações testadas.

A tabela acima representa um resumo das principais funcionalidades testadas das três aplicação mencionadas anteriormente. O cerne desta tabela é mostrar ao leitor que utilizar uma aplicação já existente não seria possível devido à ausência das características chaves que se procurava.

### 4.2.2 Android ou React Native?

É agora também importante entender o porquê de se ter escolhido o Android nativo como linguagem de programação para a aplicação móvel desenvolvida, em detrimento de qualquer outra, nomeadamente o React Native.

A primeira e talvez uma das maiores desvantagens atuais para as equipas de desenvolvimento de React Native é a ausência de documentação (Pro, 2019). Todas as informações disponíveis estão obsoletas e há muito poucos dados sobre como montar e configurar corretamente um ambiente de testes (Pro, 2019). O React Native é uma framework para desenvolvimento de aplicações móveis, escrita em JavaScript, que permite a renderização das interfaces nas plataformas nativas de iOS ou Android. Sendo JavaScript uma linguagem tão popular e flexível esta é uma das principais vantagens desta abordagem em oposição ao Android nativo (Trnka, 2018). No entanto, é também uma das suas principais desvantagens. O JavaScript é uma linguagem interpretada e estruturada, quer isto dizer que qualquer variável definida pelo programador pode ser qualquer coisa, a qualquer dado momento, e o compilador não irá auxiliar de qualquer forma o programador (Trnka, 2018). Analogamente, o Android nativo, que na sua essência é Java, é uma linguagem orientada a objetos, estruturada e imperativa.

Outra das principais desvantagens do React Native são os módulos nativos. Embora o React Native permita relacionar-se com um enorme volume de *use*-cases entre multiplataformas, é impossível que este se alargue a todos os módulos de Android nativo (Trnka, 2018). Quer isto

dizer que irá sempre existir a necessidade de módulos nativos (Trnka, 2018), que são essencialmente código em linguagem fonte que lida com um recurso específico.

É muitíssimo importante também compreender que Android nativo e até mesmo iOS nativo são linguagens já totalmente testadas, que já existem e são largamente utilizadas há muitos anos. Por outro lado, relativamente ao futuro do React Native, ainda existem muitas questões a colocar quanto à sua longevidade e sucesso.

Apesar de ser uma linguagem em constante crescimento e com um suporte cada vez mais dedicado, é necessário frisar que a última versão desta linguagem ainda não é uma versão 1.0 (Trnka, 2018). A versão mais recente já em "produção", durante a qual este documento está a ser escrito, é a versão 0.63.

| | React Native | Android |
|---|---|---|
| **Documentação** | Pouca | Muita |
| **Tutoriais** | Poucos | Muitos |
| **Facilidade de Utilização** | Dificil | Fácil |
| **POO** | Não | Sim |
| **Programação interpretada** | Sim | Não |
| **Modulos Nativos** | Alguns | Todos |
| **Estabilidade** | Pouca | Muita |
| **Versão mais recente** | 0.63 | 11.0 |
| **Escalabilidade** | Sim | Não |
| **Corre em vários S.O.** | Sim | Não |

Figura 8 - Tabela comparativa entre React Native e Android.

## 4.3 Android & IDE Android Studio

Oficialmente, o Android Studio é o ambiente de desenvolvimento integrado (IDE) da Google para programar em Android. Este IDE foi desenvolvido com base no IntelliJ IDEA, propriedade da JetBrains.

A primeira *release* desta plataforma foi feita em dezembro de 2014, no entanto, as primeiras versões – Beta & Alfa – já estariam disponíveis desde meados de 2013.

Em 2005, com a visão de entrar na corrida dos sistemas operativos de telemóveis, a Google adquiriu a empresa Android, Inc. e assumiu o controlo de todas as instalações e toda a equipa de desenvolvimento (DiMarzio, 2016). Após esta conquista, a Google desejava que este

novo Sistema Operativo (SO) fosse aberto e gratuito, como tal, a maior parte do código fonte do Android foi lançado sob a licença Apache em modo *open-source* (DiMarzio, 2016).

Visto ser uma plataforma *open-source* qualquer *developer*, com os devidos conhecimentos, ou qualquer *Software House*, podem criar e implementar novos módulos diretamente no SO de modo a customizar este conforme um conjunto de necessidade previamente determinadas (DiMarzio, 2016).

Esta customização é um dos principais fatores que levam o Android Studio a ser uma ferramenta altamente atrativa para consultoria (DiMarzio, 2016). A principal mais valia de se utilizar o Android é que este apresenta uma abordagem centralizada para o desenvolvimento de aplicativos (DiMarzio, 2016).

Atualmente, na sociedade que estamos inseridos, os telemóveis representam uma grande mais valia no sucesso de cada um. Nesta era digital, as aplicações são potencialmente o ativo mais valioso de uma cadeia de sucesso (DiMarzio, 2016).

## 4.4 Java

O Java é uma das linguagens e plataformas de programação mais ilustres e extensivamente usadas na atualidade (Prabhu, 2019). Uma plataforma é um ambiente que ajuda a desenvolver e executar programas escritos em qualquer linguagem de programação (Prabhu, 2019).

Apresenta-se como uma linguagem de programação orientada a objetos, projetada para ter o menor número possível de dependências de implementação. Destina-se a permitir que os programadores de aplicações escrevam apenas uma vez o código e o executem em qualquer lado – *Write Once Run Everywhere* (WORA) (Computerweekly, 2002)

O Java é utilizado para desenvolver aplicações móveis e *desktop*, processamento de *big data* e processamento de dados já embutidos em sistemas. De acordo com a *Oracle*, a empresa proprietária desta linguagem de programação, o Java ”corre” em cerca 3 mil milhões de dispositivos em todo o mundo, o que faz com que esta seja uma das linguagens de programação mais populares da atualidade (Programiz, n.d.).

De facto, dados de 2019 mostram que 88% de todos os smartphones do mercado correm em Android, o sistema operacional móvel escrito em Java (Codeacademy, n.d.).

## 4.5 XML & Interface Gráfica

XML significa eXtensible Markup Language. Esta linguagem de marcação altamente customizável, e como tal, extensível, que acaba por herdar daí o seu nome – eXtensible (Jackson,

2017). De facto, é inteiramente possível gerar um conjunto de “etiquetas” em XML e dotá-las de qualquer lógica de programação desejada (Jackson, 2017).

As linguagens de marcação diferem das linguagens de programação, pois usam “etiquetas”, atributos (dentro das próprias “etiquetas”) e estruturas encapsuladas para realizar tarefas que as linguagens de programação de alto nível, como por exemplo o Java, implementam usando estruturas de programação complexas recorrendo a matrizes de dados, ciclos lógicos e chamadas de métodos (Jackson, 2017).

Os atributos fazem parte das “etiquetas”e são usados para configurar e adaptar o que cada uma dessas implementa, bem como para referenciar quaisquer novos recursos, tipos de letra de texto ou valores de cores, estilos ou temas (Jackson, 2017).

Desta forma, o XML presente em Android foi especificamente desenvolvido, personalizado e implementado para o desenvolvimento de aplicações Android (Jackson, 2017). Um exemplo claro desta criação é a “etiqueta” *ConstraintLayout* (Jackson, 2017), permite implementar interfaces do utilizador que coloquem cada um dos elementos na página com determinado distanciamento em percentagem, de outro. Desta forma, o *layout* terá sempre a possibilidade de se adaptar conforme o tipo de dispositivo móvel que esteja a utilizar a aplicação. Em suma, quando estamos perante um *ConstraintLayout*, estamos perante uma interface que será responsiva. Durante o desenvolvimento da aplicação, este foi a via que segui para implementar os ecrãs do utilizador. Desta forma, podemos garantir a máxima flexibilidade da aplicação sem que a mesma comprometa a boa experiência de utilização do utilizador.

## 4.6 Firebase

O Firebase é uma ferramenta de *Backend As A Service* (BaaS) bastante popular lançada em meados de 2012. Apresenta-se como uma das plataformas dominantnessa vertente, que aprimora continuamente a experiência em *cloud* através da implementação de novos recursos e novas funcionalidades (Kumar, 2018). O Firebase é o único fornecedor de um serviço de bases de dados com sincronização automática (Kumar, 2018).

Desde que foi comprado pelo *Google*, em 2014, o Firebase transformou-se numa ferramenta bastante soberana que possui suporte para a maioria das abordagens necessárias ao desenvolvimento de software (Kumar, 2018).

Esta ferramenta foi contruída baseada em três princípios chave (Moroney, 2017):

- Desenvolvimento
- Crescimentos
- Ganhos

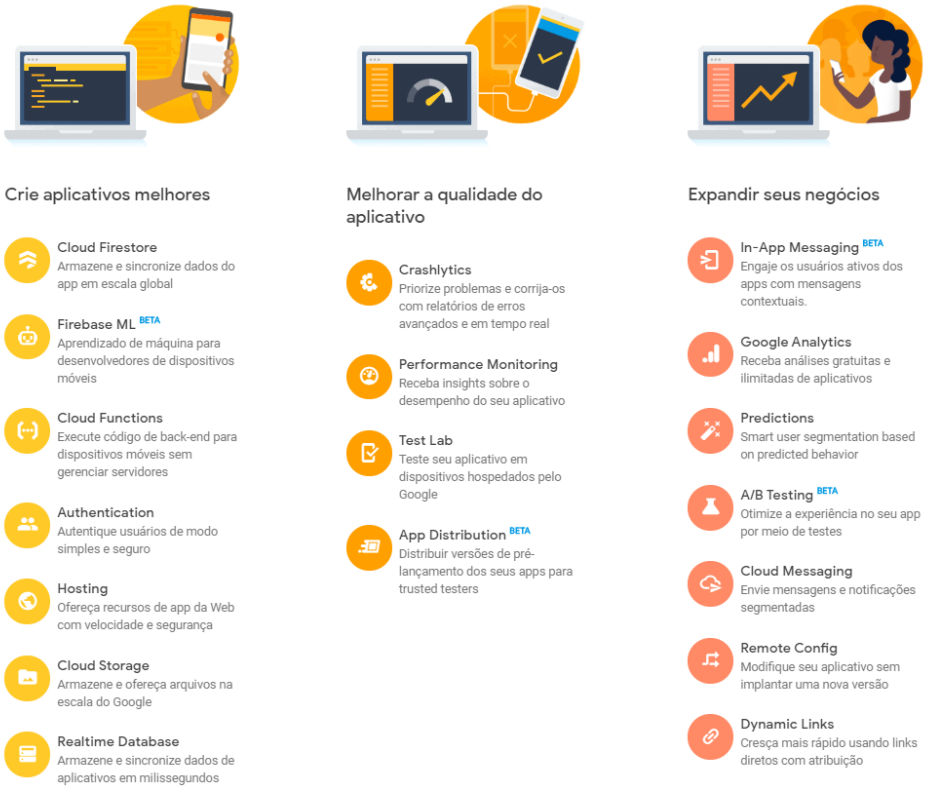


Figura 9 - Princípios chave do Firebase (Firebase, n.d.)

Todas estas funcionalidades estão interligadas entre si através de *Analytics* (Moroney, 2017). É importante notar que, apesar de todas estas funcionalidades estarem livres para uso, não é necessário que a totalidade das mesmas seja implementada em qualquer projeto de Software, pelo que cada equipa decide efetivamente que serviços pretende usufruir (Moroney, 2017).

### 4.6.1 Realtime Database

A base de dados em tempo real do Firebase - *Realtime Database* - permite que sejam criadas aplicações com um elevado nível de segurança a partir de código do lado do cliente (Kumar, 2018).

Quando novos dados são registados, toda esta informação fica armazenada em *cache* e mesmo sem acesso à internet, os eventos gerados continuam sistematicamente a ser dispara-

dos, provocando assim, uma *response* para o cliente. Quando o dispositivo em questão recupera o acesso à internet, a *Realtime Database* sincroniza todos os dados armazenados localmente com a informação que possui em nuvem, bem como quaisquer *updates* que possam ter sido necessários efetuar, para que o cliente final possua sempre os dados com as mais recentes atualizações (Kumar, 2018).

A nível de implementação desta lógica neste projeto, o *Firebase* é usado para gestão e controlo de utilizadores. Assim, permite as típicas funcionalidades associadas a estes comandos básicos, nomeadamente o registo, e o reset de password através de o envio de um email para alteração da mesma. Todas estas funcionalidades são disponibilizadas pelo *Firebase* propriamente dito, não sendo necessária a utilização de qualquer API externa. Para além disto, esta ferramenta é também utilizada para armazenar as dimensões aceites para os códigos de barra das etiquetas da DHL, bem como as funcionalidades básicas para gestão das mesmas: CRUD - Create, Read, Update, Delete.

### 4.6.2 Autenticação

Seja qual for o tipo de aplicação e o seu contexto, é importante que exista um sistema de autenticação para controlo de acesso de utilizadores a qualquer *software*. Assim, podemos definir preferências, armazenar dados e fornecer experiências personalizadas em qualquer dispositivo para qualquer utilizador (Moroney, 2017). Adicionalmente, através deste sistema de integração, a aplicação consegue gerir os tipos de menus visíveis para utilizadores do tipo administrador e do tipo não-administrador.

O Firebase permite que sejam implementados diversos métodos de autenticação, desde o habitual email-password até à utilização de fontes externas para tal (ver Figura 9), como por exemplo, o *Facebook, Google,* ou *Twitter*, através da sua API responsável para tal (Moroney, 2017). Visto que a aplicação desenvolvida visa a utilização profissional em ambiente corporativo, apenas foi implementada a autenticação via email-password.

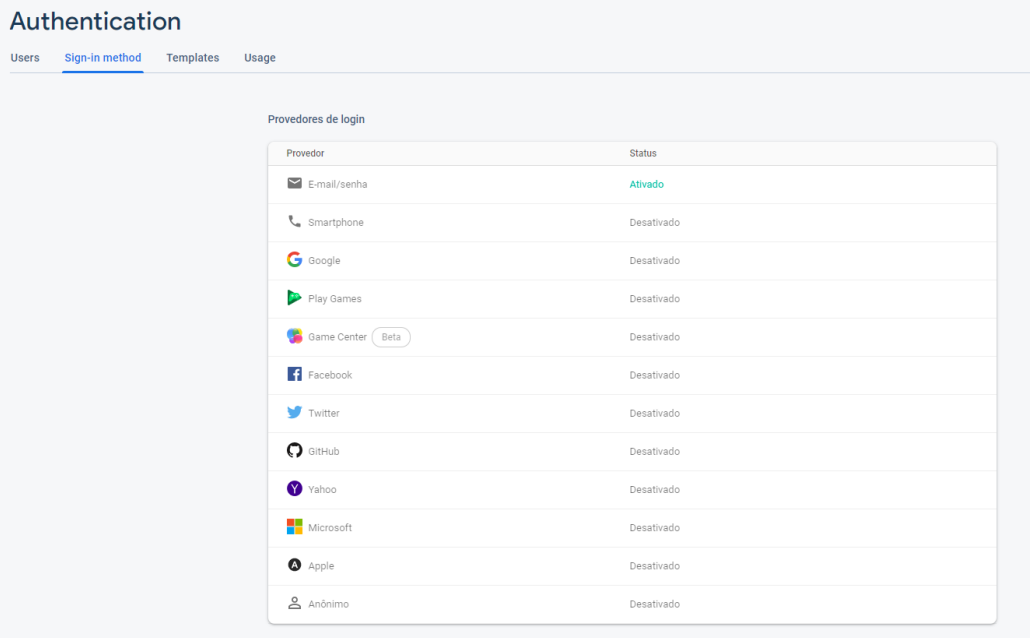


Figura 10 - Métodos de autenticação disponibilizados (Console, n.d.)

## 4.7 Trello

O *Trello* é uma ferramenta de gestão de tarefas de projetos baseada nos princípios *Kanban* (Ostergaard, 2016), abordados na secção 3.5 deste documento. Trata-se de uma ferramenta web, ou seja, o seu acesso é feito através do *browser* do utilizador em questão. Esta plataforma está disponível em três versões, no entanto, a utilizada no decorrer deste projeto foi a versão *free*.

O T*rello* é organizado como se se tratasse de um quadro virtual, onde se colocam notificações para tarefas específicas e/ou referências chave para diferentes projetos (Kaur, 2018).

O grande foco desta lógica é criar um fluxo de tarefas em execução ou que necessitam de ser completadas em conjunção com outras. O utilizador tem a livre possibilidade de, quando uma tarefa lhe é atribuída, mover essa mesma tarefa, que habitualmente, no *Trello*, é conhecida por *cartão* para os diferentes estados definidos previamente, por exemplo, "Em testes" ou "Completa".

É exatamente nesta lógica de pensamento que os princípios *Kanban* e o *Trello* estão relacionados. Um quadro *Kanban* pode ser tão simples quanto o apresentado na figura 10.

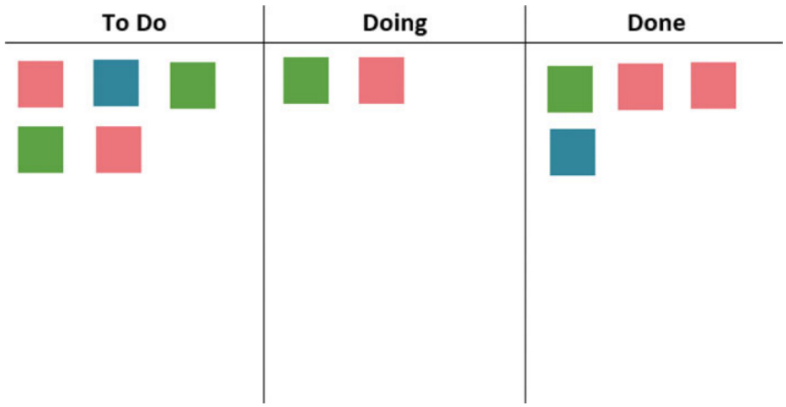


Figura 11 - Exemplo de um quadro Kanban. Fonte (Ostergaard, 2016)

Um dos pontos chave do *Kanban* é a eliminação de todo e qualquer desperdício durante o desenvolvimento de um produto: o Trello é uma ótima opção de escolha considerando este princípio chave pois possui uma interface extremamente fácil e intuitiva, estando a geração de um quadro de gestão de tarefas, criado ao gosto do utilizador, pronto a utilizar numa questão de minutos, com um esforço necessário mínimo (Ostergaard, 2016).

No âmbito deste projeto foram idealizados dois tipos de cartões a colocar na secção de tarefas a realizar - To Do. Esta seção foi então posteriormente dividida pelos dois principais componentes da aplicação móvel, nomeadamente a implementação lógica e a implementação gráfica. Assim, foram gerados cartões com funcionalidades a desenvolver e atribuídos aos tópicos *To Do - Lógica* e *To Do - Layout,* como ilustra a figura 11.

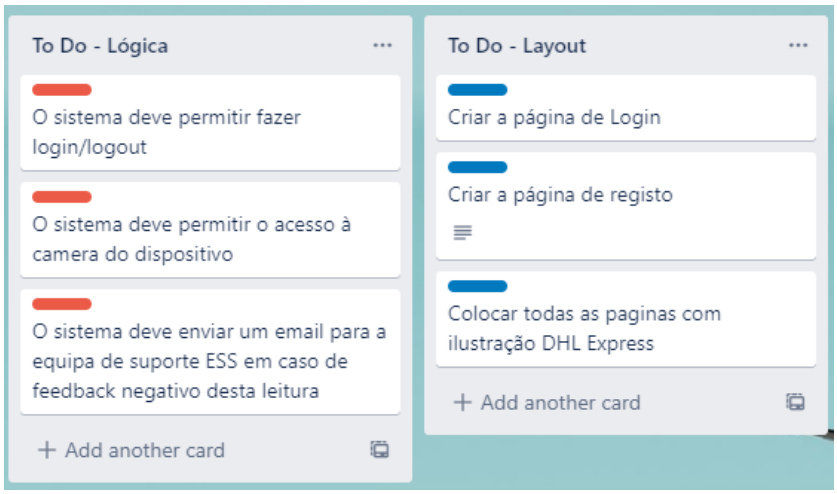


Figura 12 - Fragmento do quadro Trello utilizado com os cartões atribuídos na seção de tarefas a implementar.

Nesta implementação em concreto foi gerado um quadro Trello com as seguintes secções:

- App
- Notes
- Milestones
- To Do - Lógica
- To Do - Layout
- In progress
- Testing
- Done
- Canceled

A seção *App* é responsável por gerir o mapeamento de cores de cada cartão de tarefas com o respetivo setor onde pertence, por exemplo: vermelho - lógica ou amarelo - datas.

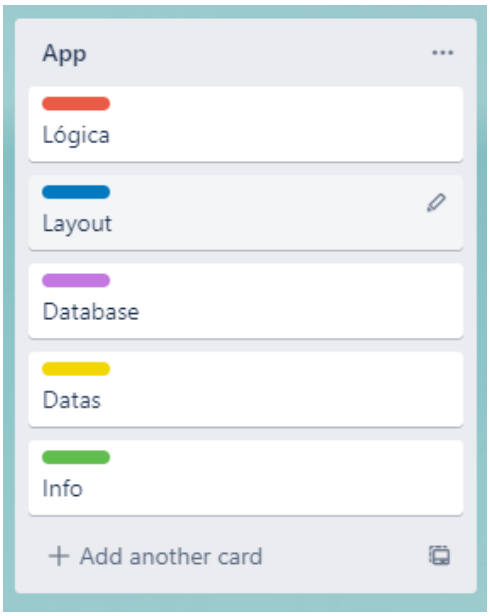


Figura 13 - Secção App.

De seguida, a seção *Notas* é a seção onde foram colocadas notas importantes e necessárias recolhidas durante o decorrer do projeto, nomeadamente links úteis e códigos de cores.

*Milestones* foi a seção onde foram definidas metas e datas a cumprir para implementação de determinadas funcionalidades. Estas funcionalidades deveriam estar devidamente testadas e implementas na aplicação para que o *milestone* fosse cumprido. De certa forma, este tópico acabou por seguir uma lógica semelhante a um *Sprint*, utilizado na metodologia *SCRUM*.

Como já abordado anteriormente, os tópicos To Do - Logica e To Do - Layout foram responsáveis por receberem os cartões com as diferentes tarefas a executar e programar.

Por sua vez, o tópico *In Progress* ficou responsável por receber tarefas que estavam no preciso momento a ser implementadas, do ponto de vista do programador e os cartões em *Testing* representam os cartões já finalizadas de *In Progress* que estão a ser alvos de várias baterias de testes..

Finalmente, os tópicos *Done* e *Canceled* representam os cartões que já se encontram implementados na última versão da aplicação, tendo previamente sido expostos a vários testes por parte do programador, para garantir que todos os seus aspetos tinham sido corretamente codificados. O tópico *Canceled*representa os cartões que, em ultima análise não foram implementados na aplicação,

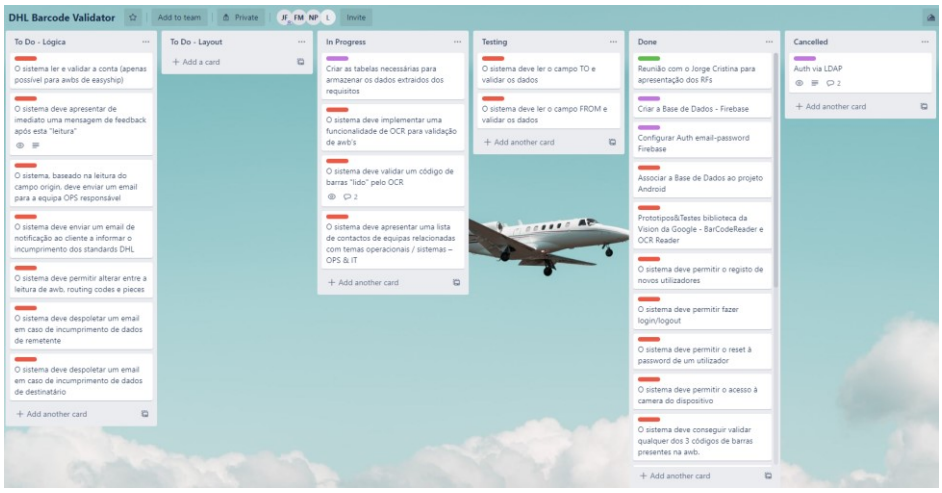


Figura 14 - Quadro Trello utilizado para gestão de tarefas no desenvolvimento da aplicação.

## 4.7 Balsamiq Wireframes

O *Balsamiq Wireframes* é uma ferramenta de geração de interfaces gráficas de baixa fidelidade que reproduz a experiência de gerar diferentes esboços, como seria feito num bloco de notas ou num quadro, mas utilizando um computador (Balsamiq, n.d.).

No nosso contexto, esta ferramenta foi utilizada para criar os protótipos de baixa fidelidade das principais interfaces gráficas utilizadas na aplicação. Este tópico será abordado em mais detalhe no capítulo 6 – Desenvolvimento da Aplicação.


**Figura 15 - Logótipo do produto** (Balsamiq, n.d.)**.**

5

# 5. Leitura de Código de Barras

## 5.1 Introdução

Neste capítulo abordo o tema da leitura e interpretação de um código de barras e as diferentes formas de o fazer no que toca a ferramentas.

É feita uma contextualização deste tema no âmbito do projeto e as suas diferentes aplicabilidades, relativamente ao ambiente corporativo e profissional.

Neste capítulo apresento também a biblioteca que foi utilizada para o desenvolvimento desta aplicação, sendo que esta questão é mais extensamente abordada no capítulo seguinte.

## 5.2 Utilização e vantagens

De modo a que possamos ter um melhor enquadramento do que efetivamente é um código de barras e as vantagens que o mesmo traz para o quotidiano, é necessário compreender como estes funcionam.

Um código de barras é uma etiqueta que é colocada em todas as mercadorias ou em todos os produtos que permite que dispositivos digitais, como por exemplo computadores, rastreiem e os registem a um ritmo muitíssimo mais rápido e eficiente do que se fosse feito manualmente (Barcodes Inc, n.d.). Atualmente estima-se que no tempo que um humano demora a clicar em 2

teclas separadas de um computador, um código de barras pode ler as informações de um produto inteiro (Barcodes Inc, n.d.). Assim, é possível ter um desempenho 10 vezes mais rápido do que se o trabalho fosse feito manualmente (Barcodes Inc, n.d.).

Ao recorrermos a um computador e a um sistema automatizado para processamento de produtos e mercadorias, acabamos por estar menos expostos ao possível surgimento de erros. Estes erros, em última análise geram trabalho adicional de retificação, o que por sua vez, tal como apresentado no capítulo 3, para além de gerar desperdícios de recursos de uma empresa, gera também perdas de tempo laboral.

Uma das principais vantagens da utilização de códigos de barras no meio corporativo e profissional, é a redução de tempo de formação dos colaboradores (Barcodes Inc, n.d.). Aprender a manusear e operar um *scanner* é uma tarefa que demora apenas alguns minutos, deixando, assim, a parte do processamento da leitura efetuada para o próprio *scanner* e para o computador. Esta abordagem é significativamente mais simplista do que perder horas de formação e treino com os diversos colaboradores de uma empresa enquanto estes estão a ser formados em como gerir as operações de produtos e afins manualmente (Barcodes Inc, n.d.).

Outra das principais vantagens da sua utilização é a sua eficácia de custo. Ou seja, o custo de impressão de uma nova etiqueta com um código de barras tem um valor despendido totalmente suportável – na ordem de 1 cêntimo por impressão – o que permite às corporações efetuar grandes alterações a nível de dados de *stock*, por exemplo, com um custo de investimento mínimo (Barcodes Inc, n.d.). Devemos ter ainda em noção, a somar a este facto, que uma etiqueta com um código de barras pode ser colocada ou colada em virtualmente qualquer coisa, independentemente da sua localização.

Desta forma é fácil compreender o porquê de os códigos de barras serem um método muito mais eficaz e fácil de gerir todo o *stock* de uma entidade, independentemente de possuir dezenas, centenas ou milhares de produtos. A economia do seculo XXI exige que as empresas fiquem à frente da concorrência e acompanhem o ritmo quando se trata de tecnologia associada aos negócios. A utilização de códigos de barras é uma das peças fundamentais para esta vantagem competitiva (Barcodes Inc, n.d.).

## 5.3 Aplicabilidade

Os códigos de barras representam atualmente uma das vias mais eficientes de transmitir informação entre o mundo real e um pedaço de software (Google, 2020).

A *API Mobile Vision* fornece uma *framework* que permite localizar objetos em fotos e vídeos, em tempo real, através de detetores que localizam e descrevem objetos visuais (Google,

2017). Oferece, também, uma poderosa *API* totalmente orientada a eventos, gerados pelo Android, que permite o rastreamento da posição dos ditos objetos em observação, neste caso, com recurso a um vídeo – câmera do telemóvel (Google, 2017).

É precisamente com recurso à câmera do telemóvel que esta solução irá ser implementada e desenvolvida. Quer isto dizer que é a partir desta biblioteca que é feito o rastreio e deteção de um código de barras.

Esta *framework*, totalmente desenvolvida de raíz pela Google, apresenta neste momento interfaces e funcionalidades, não só para deteção e processamento de códigos de barras, mas também para deteção facial  e deteção de texto, através de reconhecimento de caracteres – OCR. No cerne de um projeto, todas estas funcionalidades podem ser implementadas em simultâneo (Google, 2017b), no entanto, neste projeto em concreto, apenas iremos utilizar a funcionalidade *barcode*.

Em suma, a API *barcode* deteta códigos de barras em tempo real, no dispositivo do utilizador seja qual for a orientação do mesmo: vertical ou horizontal (Google, 2017a). Também é capaz de detetar vários códigos de barras em simultâneo, algo que não traz valor acrescido ao nosso desenvolvimento, visto apenas ser necessário a validação de um *barcode*. Todas as informações recolhidas pelo *barcode* ficam disponíveis para manipulação através de métodos disponibilizados pelo objeto em si.

Em termos de performance e experiência do utilizador, é importante notar que todo o processamento de qualquer *barcode* detetado é feito localmente, no dispositivo do utilizador, através da lógica de programação já desenvolvida por detrás, não sendo assim necessária qualquer comunicação com fontes exteriores, o que em última análise, colocaria sempre um atraso na ótica do utilizador.

## 5.4 Deteção

As classes para detetar e analisar códigos de barras estão disponíveis no pacote *open-source com.google.android.gms.vision.barcode*. A classe BarcodeDetector é o cerne de todo o trabalho da *framework*: processa e analisa objetos do tipo *Frame* para retornar uma lista de códigos de barras – Array <Barcode> (Google, n.d.).

Na sua essência, o tipo *Barcode* representa o código de barras detetado num determinado momento e todos os valores codificados no mesmo. Quando se trata de códigos de barras de apenas uma dimensão, como os que serão utilizados no foco da aplicação desenvolvida, este *barcode* apenas devolve um número – geralmente associado a um *id* - que se encontra codificado no mesmo. No entanto, também é possível efetuar a leitura a códigos de barras de duas

dimensões, por exemplo, códigos QR. Nestes casos, não é apenas retornado um número mas todo o tipo de dados que estejam codificados (Developers, 2015).

A mero titulo de curiosidade, a API está preparada para detetar os seguintes formatos de códigos de barra de 1 dimensão (Moroney, 2015):

- EAN-13
- EAN-8
- UPC-A
- UPC-E
- Code-39
- Code-93
- Code-128
- ITF
- Codabar

Analogamente, são suportados os seguintes formatos de duas dimensões (Moroney, 2015):

- QR Code
- Data Matrix
- PDF 417

No contexto da DHL Express, este código de barras tem exatamente as seguintes medidas:

- 5,95 centímetros de comprimento
- 1,2 centímetros de altura

Tratam-se de códigos de barras de 1 dimensão, codificados no formato Code-39 e Code-128, dois dos formatos suportados pela biblioteca utilizada, da Google.

Quando um código de barras é detetado pela aplicação, é devolvido um objeto de tipo BarcodeGraphic que possui os métodos necessários para que seja possível desenhar os contornos do *barcode* detetado, juntamente com um *preview* do seu valor, como é possível observar na figura 14.


Figura 16 - Contornos do código de barras detetado e pré-visualização do valor codificado.

É através destes métodos e dos contornos que é possível calcular a distância entre pontos, e, por sua vez, calcular a dimensão de um dito código de barras. Este tema e a implementação e lógica construída serão apresentados com maior nível de detalhe e rigor no próximo capítulo.

6

# 6. Desenvolvimento da Aplicação

## 6.1 Introdução

O presente capítulo visa explicar a implementação da solução pretendida para resolver o problema descrito anteriormente. Tal como já mencionado, trata-se de uma aplicação móvel e os detalhes, especificidades e decisões de implementação serão aqui descritos. Apresento exaustivamente os principais métodos utilizados, a lógica que segui e na qual me baseei para que fosse possível programar uma aplicação que efetivamente conseguisse suprimir todos os requisitos iniciais propostos pela DHL. Assim, este capítulo apresenta-se como o cerne de todo o projeto e todo o trabalho de desenvolvido.

## 6.2 Levantamento de Requisitos

Primeiramente e antes de se iniciar qualquer desenvolvimento de um projeto de software, seja que tipo de aplicação for, é sempre necessário fazer uma análise do que já existe na atualidade e os requisitos que queremos implementar, na nossa solução a desenvolver, para que consigamos colmatar a falha no processo atual.

Como tal, no seguimento de várias sessões de *brainstorming* com diversos intervenientes da DHL Express, tanto do ramo das operações, como do ramo da informática, foram levantados os seguintes requisitos funcionais a implementar nesta nova aplicação:

# Requisitos Funcionais

**RF01** - O sistema deverá permitir fazer login;

**RF02** - O sistema deverá permitir o registo de novos utilizadores;

**RF03** - O sistema deverá permitir o reset à password de um utilizador;

**RF04** - O sistema deverá permitir o acesso à câmera do dispositivo;

**RF05** - O sistema deverá apresentar e permitir a consulta de uma escada à qual o código de barras é baseado.

**RF06** - O sistema deverá implementar uma funcionalidade de OCR para validação de cartas de porte;

**RF07** - O sistema deverá validar um código de barras "lido" pelo *barcode reader*;

**RF08** – O sistema deverá conseguir validar qualquer dos 3 códigos de barras presentes na carta de porte;

**RF09** - O sistema deverá apresentar de imediato uma mensagem de feedback após esta leitura;

**RF10** - O sistema deverá enviar um email para a equipa de suporte ESS em caso de feedback negativo desta leitura;

**RF11** – O sistema, baseado na leitura do campo origem, deverá enviar um email para a equipa OPS responsável;

**RF12** - O sistema deverá enviar um email de notificação ao cliente a informar o incumprimento dos standards DHL Express;

**RF13** – O sistema deverá apresentar uma lista de contactos de equipas relacionadas com temas operacionais / sistemas – OPS & IT;

**RF14** – O sistema deverá permitir alterar entre a leitura de carta de porte , routing codes e pieces;

**RF15** – O sistema deverá ler o campo FROM e validar os dados;

**RF16** – O sistema deverá despoletar um email em caso de incumprimento de dados de remetente;

**RF17** – O sistema deverá ler o campo TO e validar os dados;

**RF18** – O sistema deverá despoletar um email em caso de incumprimento de dados de destinatário;

**RF19** – O sistema deverá ler e validar a conta do cliente;

# Requisitos de Qualidade

**RQ01** – O sistema deverá inicializar em menos de 3 segundos;

**RQ02** – O sistema não deverá demorar mais de 3 segundos a validar um código de barras;

**RQ03** – O sistema deverá permitir uma leitura em menos de 5 cliques;

**RQ04** – O sistema deverá ser o mais intuitivo possível;

**RQ05** – O sistema não deverá demorar mais de 1 segundo a mudar de menu após clique do utilizador;

**RQ06** – O sistema deverá ser implementado no IDE Android Studio.

**RQ07** – O sistema deverá não deverá permitir múltiplos registos com o mesmo email.

**RQ08** – O sistema deverá correr em várias versões do Sistema Operativo Android;

**RQ09** – O sistema deverá possuir ecrãs que se adaptem ao dispositivo;

**RQ10** – O sistema deverá estar disponível para todos os colaboradores DHL Express.

Com base em todos estes possíveis requisitos funcionais e requisitos não funcionais levantados, foi então feita uma abordagem MoSCoW aos mesmos, com o intuito de priorizar os mais críticos comparativamente aos menos críticos. Esta abordagem apenas incorporou os requisitos funcionais.

Qualquer metodologia ágil não define a forma pela qual as equipas de desenvolvimento devem priorizar o seu *backlog* de requisitos a implementar (Ferreira, 2020). Muitas equipas acabam por seguir uma abordagem baseada em valor (Ferreira, 2020), na ótica do cliente, pois é daí mesmo que surge a satisfação ao utilizar uma aplicação.

Ainda assim, outra forma muito comumente utilizada é uma organização e priorização orientada pelo esforço (Ferreira, 2020), muito defendida na metodologia *Kanban*.

O método MoSCoW é uma técnica de priorização utilizada na gestão de projetos e desenvolvimento de *softwares* com o objetivo de encontrar um consenso em comum entre as múltiplas partes interessadas sobre a importância que cada uma delas atribuem a cada requisito funcional

(Pires, 2019).Em todo e qualquer projeto ágil é vital compreender a importância de diferentes coisas. O tempo é um recurso fixo, não extensível, como tal, é fundamental compreender a prioridade de cada requisito, de cada serviço, de cada caso de uso, etc (Pires, 2019).

O termo MoSCoW é um acrónimo da linguagem inglesa que deriva da primeira letra de cada uma das quatro categorias com os "os" no meio para fazer a palavra ser pronunciável (Pires, 2019).

Desta forma, obtemos:

- Must Have;
- Should Have;
- Could have;
- Won't have;


Figura 17 - Mapeamento dos requisitos funcionais com a abordagem MoSCoW.

### 6.2.1 Levantamento de Requisitos: prioridades

Tal como o nome sugere, todo e qualquer requisito que esteja categorizado como *must have* terá a importância máxima a implementar. Representam necessidades não-negociáveis para o projeto, serviço, produto ou lançamento (Pires, 2019).

Seguidamente, os requisitos *should have* representam o segundo grau de importância de requisitos, apenas ultrapassado pelos *must have*. Representam requisitos importantes para o projeto em questão, mas não totalmente vitais para o funcionamento do mesmo (Pires, 2019). A sua ausência não significa a falha do software em questão, no entanto, quando presentes, acrescentam valor aumentado ao produto.

Após esta tipologia, temos os requisitos do tipo *could have*. Os requisitos alocados a nesta categoria, regra geral, são os primeiros a ser excluídos durante a análise dos mesmos (Pires, 2019).

Não são necessárias para o funcionamento do projeto, apesar de geralmente estarem associados a funcionalidades que, não sendo de primeiro grau de importância, seria agradável implementar numa altura de menor pressão por parte do cliente.

Por fim, temos surgem os requisitos *won't have*: representam requisitos que não serão implementados no escopo do projeto. São requisitos que não trazem valor acrescido ao projeto, não sendo para equacionada a sua implementação.

## 6.3 Implementação

A implementação desta aplicação começou primeiramente pela criação e definição dos primeiro *mockups* visuais. Ou seja, durante as diversas sessões de *brainstorming* e definição de requisitos funcionais que deveriam ser implementados.

Estes *mockups*, também conhecidos como protótipos de baixa fidelidade, foram gerados através da ferramenta *Balsamiq Wireframes*, de modo a que fosse possível ilustrar o que seria, inicialmente, a visão da aplicação desenvolvida.


Figura 18 - Protótipos de baixa fidelidade da aplicação.

Estes protótipos iniciais não devem ser descartados devido à sua baixíssima complexidade. Do ponto de vista da implementação de uma aplicação estes constituem um sólido guia com as linhas necessárias de pensamento que os programadores deverão ter durante o desenvolvimento.

Ainda assim, e falando já um pouco da estrutura do projeto em si, esta implementação foi dividida em 2 pacotes. Um pacote ficou responsável por todo o processamento lógico do código de barras em si, bem como gestão de logins e registos de novos utilizadores e funcionalidades associadas, enquanto o outro pacote ficou responsável pela gestão e controlo das atividades associadas ao lançamento e manipulação da câmara do dispositivo, bem como toda a informação obtida através daí. Estes dois pacotes são:

- com.google.android.gms.source.vision.packageBarcode.logic
- com.google.android.gms.source.vision.packageBarcode.ui.camera

O diagrama de pacotes abaixo mostra estes dois pacotes com as suas respetivas atividades.

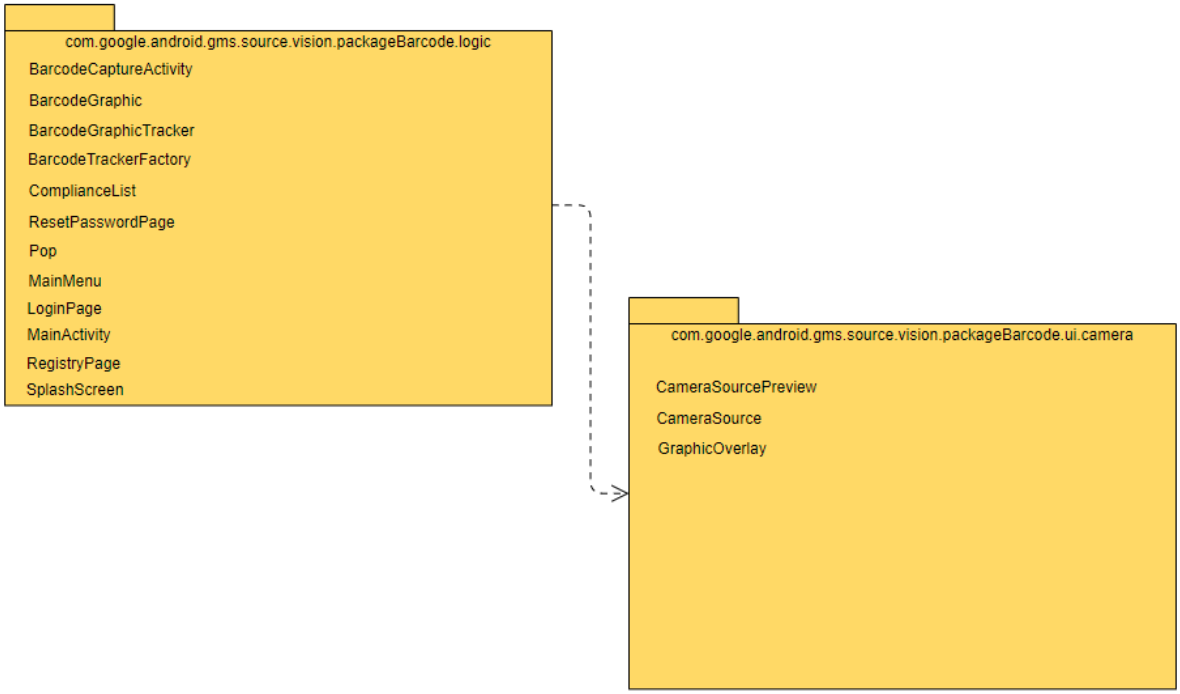


Figura 19 - Diagrama de pacotes da aplicação.

### 6.3.1 Implementação das interfaces gráficas

A programação da *app* começou com a criação e subsequente definição de todas as interfaces gráficas - *layouts* - que a aplicação deveria apresentar ao utilizador: menu de autenticação, menu de registo, *splash screen,* etc.

O *Android Studio* possui uma funcionalidade associada a estas interfaces gráficas ao qual é dado o nome *ConstraintLayout*.

Através da utilização de um *ConstraintLayout* é possível construir interfaces de elevada complexidade com uma hierarquia de visualização plana entre os diferentes elementos (Wojtek Kalicinski, 2020). Quer isto dizer que podemos gerar páginas que se adaptem autonomamente. Ou seja, a mesma página irá manter as proporções e espaçamentos entre os diferentes elementos estando um utilizador a correr a aplicação num dispositivo com um *display* de 4 polegadas ou um *tablet* de 9 polegadas. Em suma, a utilização de um *ConstraintLayout* permite gerar páginas *responsive*, sendo esse o principal motivo pelo qual todos os ecrãs utilizados nesta aplicação são baseados nesta funcionalidade.

Por outro lado, e de forma a manter uma alta coerência visual entre as diferentes páginas visualizadas durante a execução da aplicação, foi gerado um *toolbar*. Este *toolbar* apresenta-se como um cabeçalho que será importado por todos os diferentes layouts. Assim, de um ponto de vista de programação, estamos a efetuar uma otimização de código escrito, sem que exista a necessidade de definir os elementos visuais do dito *toolbar* para cada página. Em vez disso, esta é definida apenas uma vez, e agregada por todos os *layouts*.

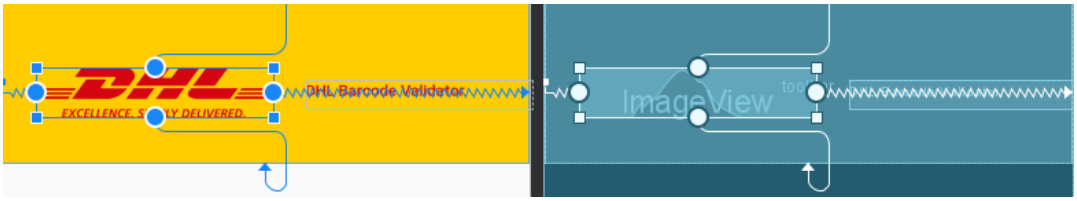


Figura 20 - Definição e implementação do cabeçalho.

Na figura 20 podemos observar a implementação visual deste *layout*. De notar todas as proporções e dependências que estão alocadas ao componente *ImageView* e ao componente *TextView*. Estas dependência são o cerne de uma página adaptativa, capaz de manter as proporções seja qual for o tamanho do ecrã onde a aplicação está a ser executada.

Foram implementadas as seguintes interfaces:

- splash_screen.xml
- login_page.xml
- register_menu.xml
- reset_password_page.xml
- toolbar.xml
- activity_main.xml
- barcode_capture.xml
- compliance_list.xml
- pop.xml

Cada uma destas páginas tem um claro propósito, bem definido e está respetivamente associada a um ficheiro.Java, onde é implementada a sua lógica funcional.

O splash_screen.xml é a primeira página que um utilizador visualiza assim que inicializa a aplicação. A página não apresenta quaisquer informações, apenas apresenta uma imagem de

marca da DHL, enquanto o resto dos módulos é carregado. Este processo está associado a um atraso de sensivelmente 1.5 segundos.

Após a inicialização, é apresentado ao utilizador a página de login (login_page.xml), em que o utilizador terá oportunidade para se autenticar, ou caso não esteja registado, terá a opção de seguir para a página de registo (register_menu.xml) onde deverá inserir os seus dados. Após registo, a aplicação retorna automaticamente ao menu de login. Ainda nesta página, o utilizador poderá selecionar seguir para a página onde é feito o *reset* à sua password (reset_password_page.xml). Esta por sua vez e tal como o menu de registo, após inserção dos respetivos dados, retorna à página de autenticação.

Feito o *login* o utilizador visualiza o menu activity_main.xml. Este menu é responsável por ativar a câmara do telemóvel, através de um clique no botão responsável para tal. Este clique irá "chamar" a página barcode_capture.xml, onde será apresentado tudo o que a câmara do telemóvel esteja a captar.

É com esta página que o utilizador deverá efetuar as medições dos códigos de barra, apontando o telemóvel para as respetivas cartas de porte. O activity_main.xml, para além de inicializar todo o cerne da aplicação, é também responsável pela apresentação de dados e feedback destas ditas leituras. Em caso de leitura de um código de barras que esteja inválido, a aplicação lança uma nova janela (pop.xml), que ocupa apenas 1/3 da página anterior. Este novo menu questiona o utilizador se pretende enviar um email para a equipa de *Electronic Shipping Solutions*, da DHL.

Finalmente, o último menu da aplicação (compliance_list.xml) é acedido através do botão respetivo para tal, que está disponível na página activity_main.xml. O grande foco deste menu é mostrar ao utilizador as proporções e a distância à qual o utilizador deve colocar a sua câmara de telemóvel para garantir leituras corretas, sem falsos-positivos.

### 6.3.2 Implementação de recursos visuais

Todos os projetos construídos em Android Studio irão gerar automaticamente um conjunto de pastas onde serão guardadas propriedades, imagens e até como elementos gráficos. É um processo totalmente automatizado feito pelo IDE.

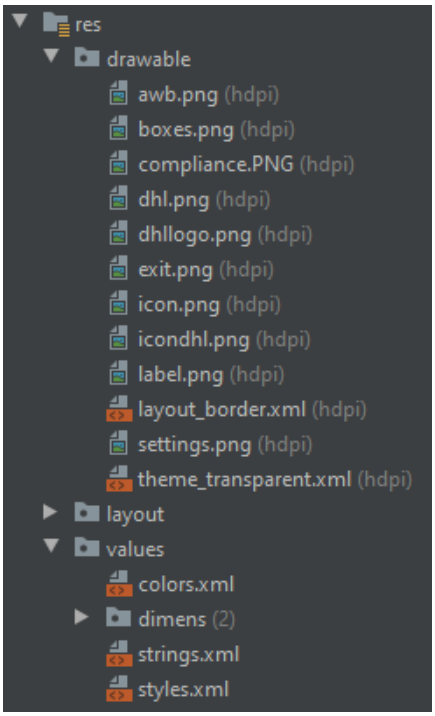


Figura 21 - Estrutura de pastas dos recursos visuais.

Neste projeto em concreto foram utilizadas algumas imagens em formato PNG que representam recursos da DHL Express. Por norma, estas imagens ficam sempre associadas à pasta *drawable*, do Android Studio.

Por outro lado, utilizei a pasta *value* para definir constantes, por exemplo: frases de *output* de erro / informação e nomes de botões, para definir códigos de cores sobre o formato hexadecimal e até como estilos (styles.xml) onde são definidos os recursos de cada elemento a utilizar na interface do utilizador.

```
<style name="Theme.AppCompat.Light.NoActionBar.FullScreen" parent="@style/Theme.AppCompat.Light.NoActionBar">
    <item name="android:windowNoTitle">true</item>
    <item name="android:windowActionBar">false</item>
    <item name="android:windowFullscreen">true</item>
    <item name="android:windowContentOverlay">@null</item>
```

Figura 22 - Implementação dos recursos necessários a tornar as interfaces *fullscreen* e sem barra de notificações.

Optou-se por se definir este estilo e subsequentemente colocar todas as interfaces a herdar esta propriedade para que fosse possível gerar uma melhor experiência de utilização da aplicação ao utilizador, colocando a aplicação a cobrir a totalidade do ecrã do telemóvel.

Por outro lado, e como mencionado anteriormente, foram implementados diversos textos padrão a utilizar em diversos locais da aplicação. O grande foco desta abordagem é permitir que estas propriedades sejam o mais propagáveis possíveis, por entre o projeto, e que em caso de qualquer alteração ao seu valor propriamente dito, não ser necessário editar manualmente cada botão em si. Desta forma, conseguimos ter um código mais coeso, com o mínimo de desperdício de tempo por parte do programador, nos eventuais *updates* necessários.

### 6.3.3 Implementação lógica

Após a aplicação ser inicializada, a primeira interface com a qual o utilizador terá uma interação direta, ou seja, onde é preciso uma ação deste para que aconteça "algo", é o menu de *login*.

A implementação lógica deste menu segue uma abordagem bastante simplista: um botão para se registar e um botão para efetuar o *reset* à password. Cada um destes botões reencaminha o utilizador para novas classes Java, respetivamente. Adicionalmente, existe como cerne o botão *login*. Quando este botão é clicado, a aplicação irá consumir os dados que se encontram nas respetivas *textview* onde os dados são inseridos. Como mencionado no capítulo 4, a autenticação é feita com recurso ao Firebase: esta entidade faculta essa dita funcionalidade que deverá ser chamada num método responsável para tal. Como complemento, foi criado um método para validação dos dados que são inseridos pelo utilizador. O objetivo aqui, é impossibilitar este de avançar para novos menus com dados a *null*.


Figura 23 - Implementação do Login com recurso ao Firebase.

Analogamente, a implementação lógica para registo de um novo utilizador segue uma abordagem semelhante. A diferença fulcral reside no facto de o método em si não tentar efetuar um mapeamento entre utilizador-password já registados no Firebase mas sim gerar um novo registo para este, encriptando a password no processo.

De facto, as três principais funcionalidades para gestão de um utilizador são totalmente suportadas pelo Firebase. Existem métodos responsáveis para tal na sua implementação em Android Studio, apenas é necessário invocar um método para tal, com o devido tratamento e captura destes dados.

Como mencionado anteriormente tanto no capítulo 4 como no capítulo 5, esta aplicação utiliza a biblioteca *Vision* da Google. Sendo esta biblioteca totalmente *open-source*, temos a possibilidade de a adaptar totalmente às nossas necessidades, sendo esta a abordagem seguida. Existe sete classes Java, totalmente "herdadas" desta biblioteca, que são:

- BarcodeCaptureActivity.java
- BarcodeGraphic.java
- BarcodeGraphicTracker.java
- BarcodeTrackerFactory.java
- CameraSource.Java
- CameraSourcePreview.Java
- GraphicOverlay.Java

Todas estas classes mantêm o seu código fonte totalmente inalterável com a pequena exceção da classe CameraSourcePreview.

Por defeito, a API da Google não apresenta a pré-visualização da câmara como uma página *fullscreen*. Ou seja, iremos sempre observar uma parte do ecrã que não se encontra preenchida com qualquer informação, estando apenas a "ocupar espaço". Isto faz com que a aplicação apresente um aspeto pouco cuidado e profissional.Esta retificação do código fonte é feito alterando apenas a lógica deste fragmento de modo a permitir uma experiência de utilização mais agradável.

Todas as classes do tipo Barcode são responsáveis por capturar informação e gerir todo o comportamento desta durante a execução do programa.

A classe BarcodeGraphic é responsável por gerar os limites do código de barras que são detetados pela câmara do telemóvel. Todas as classes têm igual importância e contributo para o bom funcionamento da aplicação, no entanto esta classe é uma das diversas que irá permitir calcular efetivamente valores e proporções destes *barcodes*.O método principal desta classe é método *draw*, que, na sua essência, recebe apenas como atributo um objeto do tipo Canvas. Já dentro deste método utilizamos um objeto do tipo RectF em conjugação com este objeto Canvas.A classe RectF é uma das classes pré-existentes do Android que permite obter e gravar as 4 coordenadas geográficas de um retângulo, ou seja, os seus 4 vértices:

- Esquerda;
- Direita;
- Baixo;
- Cima.

Para finalizar, este método agrega os dados recolhidos pelo objeto ReatF e injeta esta informação no objeto recebido como atributo – Canvas – e apresenta o valor lido do código de barras imediatamente a baixo. Assim, é possível apresentar ao utilizar uma pré-visualização em tempo real do código de barras detetado.

```
//Draws the barcode limits
canvas.drawRect(rect, mRectPaint);
// Draws a label at the bottom of the barcode
canvas.drawText(barcode.rawValue, rect.left, rect.bottom, mTextPaint);
```

Figura 24 - Fragmento do método *draw*.

Seguidamente, a classe BarcodeGraphicTracker é a classe que gere toda o comportamento da classe anterior – BarcodeGraphic. Esta classe possui três métodos fulcrais:

- onNewItem;
- onUpdate;
- onMissing.

O método *onNewItem*, tal como o nome automaticamente releva, inicia o *tracking* do código de barras detetado, dentro da sobreposição deste. Por sua vez, o comportamento do método *onUpdate* é atualizar a posição e as características detetadas do código de barras, novamente, dentro da sobreposição deste. Finalmente, o método *onMissing* oculta esta sobreposição, que nada mais é do que o desenho dos limites do código de barras detetado, quando o código de barras não é automaticamente captado pelo câmara. Este método é executado e está geralmente associado a falhas de captura, por exemplo, quando temos um código de barras parcialmente visível.

Por outro lado, a classe BarcodeTrackerFactory é utilizada para gerir e criar *trackers,* cujo um gráfico instanciado, será associado a um código de barras detetado. Em *runtime*, o processador utiliza esta classe para gerar *trackers* de códigos de barra detetados, tanto quantos sejam necessários. É através desta funcionalidade que é possível a câmara do dispositivo onde a aplicação está a correr detetar e apresentar os contornos dos três códigos de barra presentes numa carta de porte da DHL Express.

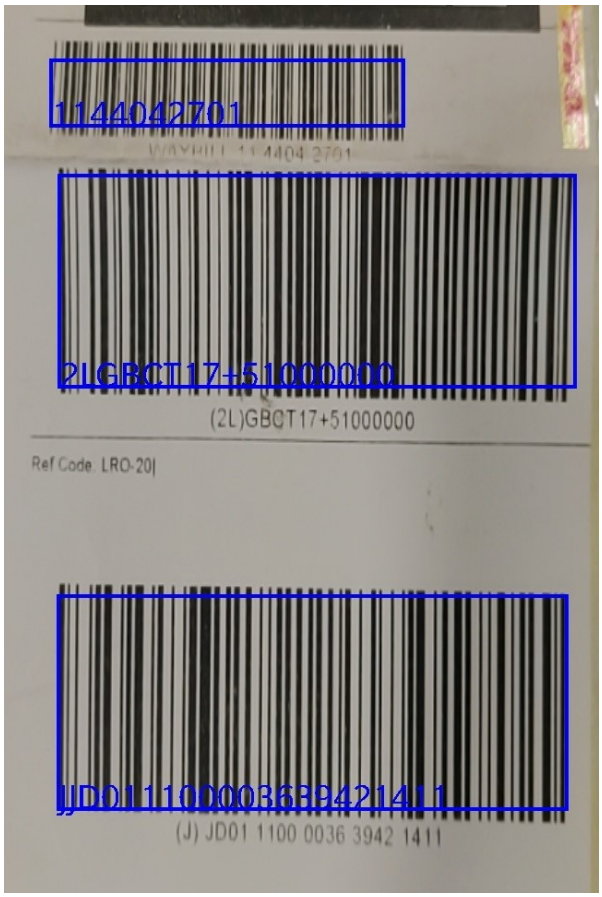


Figura 25 - Deteção de múltiplos códigos de barra em tempo real.

Afastando-nos agora um pouco das classes utilizadas para gerir todo o processo de deteção e processamento de códigos de barra, entramos nas classes designadas à implementação e gestão do controlo da câmara do dispositivo do utilizador.

Estas três classes - CameraSource, CameraSourcePreview e GraphicOverlay - estão agrupadas dentro do seu próprio pacote: ui.camera.

Esta lógica de estrutura é nativa da biblioteca utilizada - Vision API - pelo que o código fonte das mesmas manteve-se totalmente inalterado no decorrer do projeto, apesar de serem totalmente abertas a manipulação por parte de qualquer programador.

Retrocedendo agora um pouco para a sequência de classes e interfaces com as quais o utilizador interage, apresenta-se a classe SplashScreen. Esta classe representa uma das clássicas abordagens ao menu inicial de uma aplicação. Enquanto os recursos necessários de toda a aplicação carregam, o utilizador visualiza sempre a mesma interface. Assim que estes recursos

estão prontos a utilizar, a aplicação automaticamente despoleta um método que inicializa uma nova atividade, ou seja, avança para o próximo menu.

O *loading* desta biblioteca é instantâneo e toda a leitura ocorre em tempo de execução, pelo que não existiria qualquer recurso massivo a carregar que despoletasse este *delay*. Como tal, nesta classe SplashScreen, foi implementado um *timer*, que após 1 segundo da interface ter sido carregada, automaticamente inicializa uma nova atividade. Neste caso, essa nova atividade inicializada é o menu de Login.


Figura 26 - SplashScreen da aplicação.

Deixando o SplashScreen devidamente apresentado, fica apenas em falta apresentar três classes Java utilizadas na aplicação:

- ComplianceList.Java
- Pop.Java
- MainActivity.Java

A classe ComplianceList é uma classe desenvolvida para apresentar ao utilizador da aplicação uma referência de como efetuar uma leitura de um código de barras. Ao iniciar-se a aplicação, antes da câmara ser lançada, é apresentado um aviso ao utilizador a indicar que as medições devem ser feitas com uma distância de dez centímetros da carta de porte, de modo a obter a máxima fidelidade na medição do código de barras, evitando assim, resultados falsos-positivos. No entanto, é difícil ter-se a perceção do que é efetuar uma leitura a *dez centímetros* do código de barras. Assim, esta classe apresenta um *layout* exemplo do que o utilizador deverá ver no seu dispositivo antes de confirmar a leitura.

De seguida, temos a classe Pop.Java.Esta classe é responsável por gerir o input do utilizar após uma leitura inválida de um código de barras. É uma classe que se sobrepõe à classe MainActivity, no sentido do seu *layout* surgir por cima do anterior.


Figura 27 - Feedback após leitura de um código de barras.

Caso o utilizador selecione a opção *Sim*, esta classe é responsável por lançar um formulário de Gmail onde são automaticamente passados todos os dados recolhidos durante a leitura efetuada ao código de barras:

- Comprimento
- Altura
- Valor codificado

Todos estes valores são colecionados como argumento durante a execução da classe e por sua vez armazenados em variáveis locais para esse efeito.É importante notar que é necessário forçar o Android, nesta classe, a utilizar um dos Packages próprias da Google, neste caso, para abrir esta nova janela de escrita de email diretamente no Gmail. Por defeito, esta funcionalidade quando invocada irá abrir por sua vez uma lista com as potenciais aplicações que o utilizador pode utilizar para criar um email. De forma a "saltar" este passo e automaticamente apresentar ao utilizador o formulário de escrita de um email, como dito anteriormente, é forçado a passagem do Package da Google, para esse efeito.

```
Bundle extras = getIntent().getExtras();
if (extras != null) {
    String value = extras.getString( key: "awbValue");
    String width = extras.getString( key: "width");
    String height = extras.getString( key: "height");
    emailIntent.putExtra(Intent.EXTRA_EMAIL, TO);
    emailIntent.putExtra(Intent.EXTRA_CC, CC);
    emailIntent.putExtra(Intent.EXTRA_SUBJECT,  value: "Invalid Barcode Detected");
    emailIntent.putExtra(Intent.EXTRA_TEXT,  value: "Invalid barcode detected!" + "\n" + "\nAwb Number: " + value
            + "\nWidth: " + width + "\nHeight: " + height);
    emailIntent.setPackage("com.google.android.gm");
```

Figura 28 - Fragmento do código implementado para envio de email.

Finalmente, a classe que possibilita todo o processamento de dados e leituras, é a classe MainActivity.Java. Esta classe, sem contando claro está com os método e funções necessárias para mapear valores e inicializar as variáveis dos *layouts*, é composta por quatro métodos principais, cada um com uma função bem definida. Estes métodos funcionam todos entre si de modo a permitir ao utilizador atingir o objetivo final e cerne da aplicação.

- OnClick;
- OnActivityResult;
- ValidateData;
- ShowEmailPopup;

De modo a efetuar uma leitura das medições do código de barras, a aplicação está constantemente à escuta de um *input* do utilizador. Nesta aplicação o *input* do utilizador é um simples clique no ecrã do dispositivo. Não importa a zona onde é feito o clique, não é necessário que seja sequer no código de barras em si, apenas é necessário que seja efetuado.

Assim que este clique é detetado, o método onClick é executado. Este método inicializa uma nova atividade - BarcodeCaptureActivity - a atividade responsável por capturar e processar os códigos de barra. Caso as funcionalidades de *auto-focus* ou o *flash* estejam selecionados, estes parâmetros são passados como atributos para a nova atividade. Ou seja, quando a aplicação lançar a câmara do dispositivo estes poderão estar ou não ativos. Por defeito, o *auto-focus* está pré-selecionado.

De seguida temos o método onActivityResult. Este método recebe três parâmetros de entrada: um valor inteiro que representa o valor de quando um código de barras é detetado, um código de sucesso e um *intent.* Um *intent* é um objeto do tipo mensagem que pode ser utilizado para solicitar uma ação de outro componente ou atividade da aplicação.É exatamente através deste *intent* que toda a informação é transmitida entre as diferentes atividades e permite a este método processar a informação obtida pela atividade anterior.

De seguida é inicializado um objeto do tipo Barcode que irá ser inicializado com a informação presente no *intent*. Após isto, todas as variáveis são mapeadas com os corretos valores.

```
Barcode barcode = data.getParcelableExtra(BarcodeCaptureActivity.BarcodeObject);

barcodeValue.setText("Awb: " + barcode.displayValue);
awbValue = barcode.displayValue;
widthValue.setText(String.valueOf(barcode.getBoundingBox().width() / 100.0));
widthToPass = String.valueOf(barcode.getBoundingBox().width() / 100.0);
heightValue.setText(String.valueOf(barcode.getBoundingBox().height() / 100.0));
heightToPass = String.valueOf(barcode.getBoundingBox().height() / 100.0);
```

Figura 29 - Passagem dos valores colecionados pela atividade BarcodeCaptureActivity para processamento.

Após estes dados estarem devidamente populados, este método invoca o próximo método a apresentar: validateData. Este método, por sua vez, recebe dois argumentos, provenientes da atividade anterior: comprimento e largura do código de barras detetado. Assim que este método é chamado, é automaticamente feita a inicialização da instância da base de dados em Firebase que a aplicação utiliza.

Essa instância da base de dados encapsulado num objeto do tipo DatabaseReference, vai consultar as diversas entradas, os diversos registos, que estejam gravados na base de dados das diferentes conjugações e proporções dos códigos de barra.

De momento, para o código de barras que estudo, apenas existe uma solução única: 5,95 centímetros de comprimento e 1,2 centímetros de altura. Esta abordagem foi seguida pois, de momento, a aplicação apenas permite a leitura de um tipo de códigos de barra, no entanto, visto estarem nos planos futuros da aplicação, a sua escalabilidade para outros limites, foi utilizada esta lógica de implementação.

Assim, no momento da consulta do objeto de base de dados, são utilizadas duas variáveis locais para armazenar o valor do comprimento e da largura. Estas duas variáveis são então utilizadas para fazer a comparação entre os dados medidos pela atividade BarcodeCaptureActivity e os valores já registados na base de dados. Ainda assim, de modo a dar alguma margem de segurança na leitura, foram gerados dois intervalos de valores possíveis. Quer isto dizer, que, se a leitura estiver contida dentro deste intervalo de valores, o *output* irá ser positivo, caso contrário o *output* será negativo. Isto, claro está, para ambos os eixos de medição.

Caso o *output* seja de sucesso, irá surgir uma mensagem de *feedback*, com letras verdes, a apresentar esse mesmo resultado.

Caso o *output* seja de insucesso, analogamente à situação anterior, a mensagem irá surgir a letras vermelhas. Além de apresentar uma mensagem, é também automaticamente despoletado um *timer.* Este *timer* é responsável por deixar a mensagem visível ao utilizador durante 2 segundos e de seguida executar um novo método. Este método é o showEmailPopup(), que irá receber sempre três argumentos: O número da carta de porte lida, o comprimento do código de barras e a largura do código de barras

Quando este método é inicializado, a sua função é chamar e inicializar uma atividade já apresentada anteriormente - Pop.Java - passando como argumentos para a mesma, estes três valores recebidos.

Estão assim descritas todas as classes utilizadas para permitir que esta aplicação funcione como desejado, apresentando de seguida o fluxo de funcionamento da aplicação - BPD.

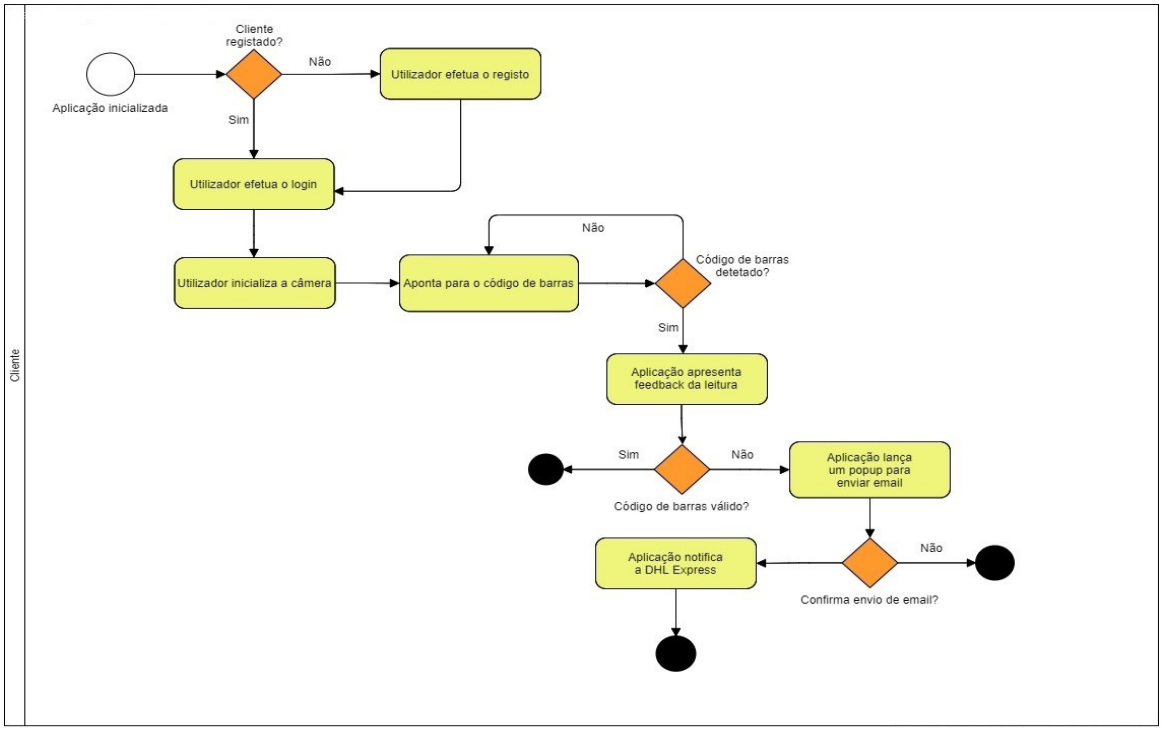


Figura 30 - Business Process Diagram da aplicação.

## 6.4 Lançamento – Feedback e resultados

Estando a aplicação finalizada é hora de a "lançar" no mercado. Neste caso, visto ser uma aplicação para uso interno corporativo, este "mercado" alvo seriam apenas os colaboradores da DHL.

Podemos agora aqui agrupar duas entidades cujos testes e validação da aplicação seriam interessantes do ponto de vista funcional:

- Equipa de informática da DHL - Electronic Shipping Solutions (ESS)
- Equipa de expedição da DHL - Operations (OPS)

A equipa de ESS é constituída por 5 elementos, sendo o autor deste projeto um desses elementos, pelo que a validação da aplicação apenas poderia ser efetuada por 4 destes. A aplicação foi partilhada sobre a forma de um executável - APK - a ser instalado no dispositivo móvel de cada um dos colaboradores. Para além deste executável, foram também partilhadas algumas cartas de porte contendo códigos de barra com proporções comprimento-altura inválidos e outras cartas de porte contendo códigos de barra totalmente válidos.

No geral, o feedback foi positivo e os resultados atingidos foram efetivamente os esperados. Cartas de porte cujos códigos de barra não apresentavam proporções corretas foram sinalizados pela aplicação. Ainda assim, o feedback não foi perfeito e foram levantados alguns pontos de melhoria e até algumas observações. Foi através destes testes que foi possível notar que algumas mensagens da aplicação não estariam conforme o esperado, nomeadamente na uniformização do idioma da aplicação, visto que algumas mensagens estavam em Inglês e outras em Português. Com base nestas notas estes pontos foram prontamente retificados e atualmente todas as mensagens de sucesso ou insucesso da aplicação encontram-se em Português.

Foi também graças a estes testes que foi possível descobrir uma lacuna na aplicação em si, que era perceber qual a distância ideal entre a câmara do dispositivo e o código de barras. A biblioteca utilizada e facultada pela Google para implementação desta aplicação corre em *run-time*, ou seja, o processamento é feito quando a aplicação está em execução. Quer isto dizer que, um código de barras poderá ter medidas, captadas pela câmara, totalmente diferentes dependendo da distância a que o dito código de barras é medido.

Após alguns testes, verificou-se que a distância ideal para que se consiga efetuar leituras corretas - não existindo falsos negativos ou falsos positivos - é de 10 centímetros. Ou seja, sempre que a medição for feita a cerca de 10 centímetros do código de barras, teremos a certeza de que os valores devolvidos pela aplicação serão os corretos. Tendo em conta este feedback, foram tomadas algumas medidas preventivas para que não tornassem o produto inviável para lançamento, tais como:

- Apresentar esta mensagem diretamente no monitor do dispositivo no menu onde o utilizador "lança" a câmara.
- Foi também adicionada uma atividade que apresenta uma pré-visualização que o utilizador deve tentar replicar quando efetuar uma leitura. O grande objetivo desta atividade é dar ao utilizador uma noção do que efetivamente é a distância ideal de medição: 10 centímetros.

Em suma, a equipa de Electronic Shipping Solutions da DHL Express validou a aplicação e considerou-a apta para avançar internamente.

É importante agora compreender, na perspetiva das Operações da DHL Express, quanto tempo se perde, em média, nestes casos de cartas de porte inválidas. Analisando o tráfego e volume de encomendas processadas, verificou-se a perda de cerca de 30-35 segundos, entre a falha na leitura, o desvio da mercadoria, a colocação de uma nova etiqueta, e colocação no início da cinta de processamento.

Este processo pode ser analisado ao detalhe através do diagrama apresentado abaixo.

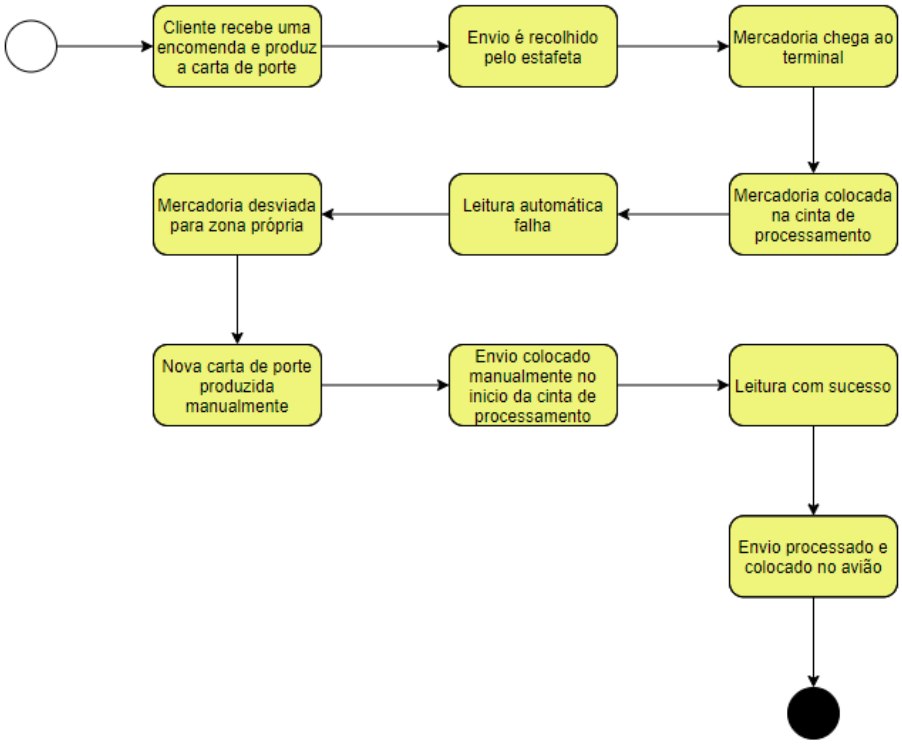


Figura 31 - BPD do processo atual em caso de falha de leitura automatizada.

É importante aqui perceber que a grande perda de tempo não ocorre com a criação/impressão de uma nova carta de porte, mas sim no processo como um todo. Ou seja, todas as suas etapas, coletivamente, criam este impacto negativo na operação.

Utilizando a aplicação desenvolvida, irá haver um impacto positivo neste aspeto. Como já mencionado anteriormente, o objetivo passa por retificar o problema na origem, neste caso, o cliente. Quando o envio inválido é detetado, a equipa de informática deverá contactar imediatamente o cliente e resolver este tema o mais rápido possível, enquanto a equipa das operações deve tomar medidas preventivas, para que este envio "inválido" não tenha consequências negativas no processamento.

Estas medidas preventivas passam por criar uma nova carta de porte, que já esteja totalmente válida e, assim que a encomenda chegue ao terminal, a mesma seja novamente etiquetada e colocada na cinta de processamento.

Assim, com esta alteração de processo, podemos apresentar o seguinte diagrama:

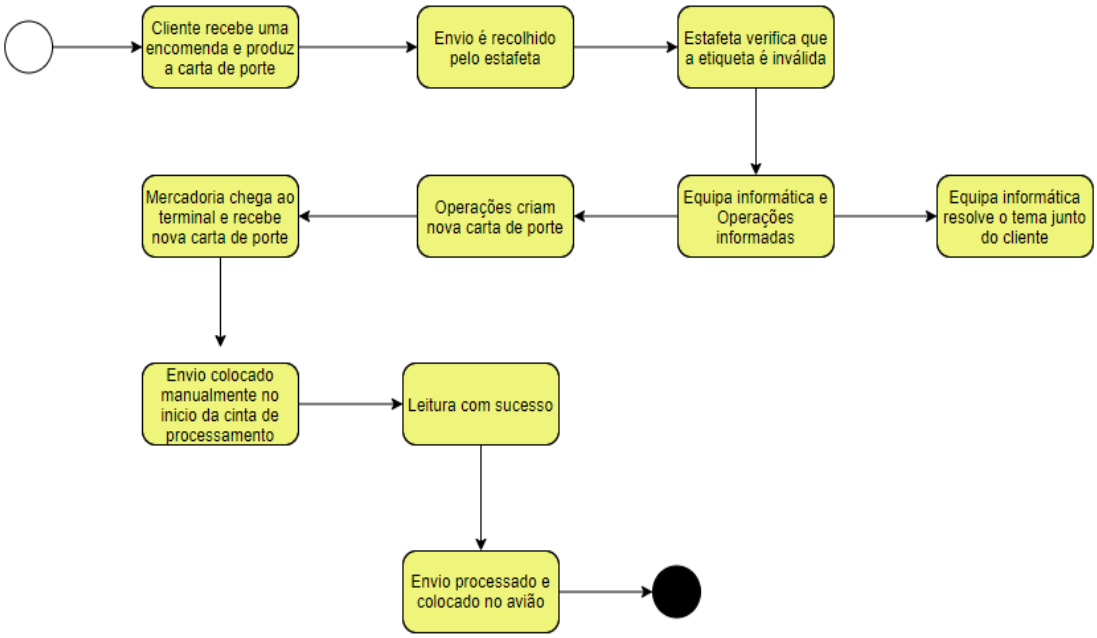


Figura 32 - BPD do processo com a deteção da invalidade da carta de porte na origem, com recurso à aplicação desenvolvida.

Com esta alteração existe um tempo gasto estimado de apenas 10-15 segundos. Ou seja, comparativamente à solução inicial, existe uma poupança de cerca de 20-25 segundos desde que esta falha é notada até ao momento da sua correção.

Olhando agora um pouco para valores e números propriamente ditos, esta diferença estimada de cerca de 20 segundos acaba por proporcionar um acréscimo significativo no rendimento e eficiência na operação da DHL Express. Vamos supor que a DHL Express processa, numa semana, 100 mil envios pelos seus diversos terminais em Portugal, e que, dessas 100 mil envios, apenas 0.1% apresentam falhas nos códigos de barras. Esta percentagem equivale a 100 envios. Vamos estimar por baixo e assumir que apenas são perdidos 30 segundos da operação, por envio, com este problema. Nesse caso, estamos a falar de 3 mil segundos, ou seja, 50 minutos.

Analogamente, seguindo a mesma lógica de cálculos, o tempo despendido com a utilização desta aplicação, seria apenas de 16 minutos e 40 segundos, o que revela uma melhoria de eficiência de 33 minutos e 20 segundos.

Numa operação onde o cerne do negócio é o serviço expresso, uma diferença de 33 minutos pode e por vezes significa a diferença entre garantir um serviço de entrega em qualquer ponto do mundo em 24 horas, ou, a falha desse compromisso.

7

# 7. Conclusões e trabalho futuro

Este projeto assumiu como objetivo fulcral a melhoria de um processo já existente na DHL Express. O conhecimento adquirido ao longo deste Mestrado, bem como da Licenciatura permitiram-me desenvolver uma solução que fosse capaz de suprimir uma necessidade real da empresa. Para além das competências técnicas necessárias envolvidas na programação lógica da aplicação, foi também necessário efetuar o levantamento da situação atual e as dificuldades que a mesma causava.

Trata-se de uma solução que irá efetivamente entrar em produção nas operações diárias da DHL Express. Acompanhará a qualquer instante qualquer estafeta, estando sempre à distância de um clique do seu dispositivo móvel. Esta solução irá assim trazer vantagens operacionais estratégicas que anteriormente comprometeriam o negócio da empresa e o compromisso desta para com os clientes. Assim, a implementação desta solução irá otimizar todo o negócio como um todo, estando as suas vantagens claramente apresentadas para os diversos elementos envolvidos, seja o cliente, o funcionário, ou gestor.

Em suma, o desenvolvimento desta aplicação permitiu-me adquirir novos e mais aprofundados conhecimentos, não só da linguagem de programação Android, na vertente de bibliotecas existente, algoritmos e extensões, mas também *skills* de gestão de tarefas, organização de trabalho e implementação de metodologias em ambiente empresarial.

Com o desfecho e finalização da aplicação proposta a desenvolver, devemos agora debruçar-nos sobre as ilações que podemos retirar de todo este projeto.

Inicialmente foi proposto que se deveria criar e implementar uma aplicação que fosse capaz de identificar e sinalizar cartas de porte, cujos códigos de barra se encontrassem inválidos, segundo os parâmetros utilizados pela DHL. Devemos questionar-nos se a aplicação cumpre efetivamente essa funcionalidade e a resposta é sim. A aplicação é totalmente capaz de reconhecer formatos de códigos de barra que não pertençam ao leque de possíveis proporções que a DHL reconheça como *standard*.

É também importante compreender toda a jornada, metodologia, planeamento e abordagem seguida. Esta aplicação sem dúvida segue uma abordagem *Lean,* cujo foco central reside na eliminação a todo o custo do desperdício de trabalho e cujos princípios fundamentais deste acabam por não só contribuir para a implementação da aplicação em si, mas também por se estender para uma nova visão: mais eficiência na DHL. Ao conseguirmos eliminar uma falha logo no momento da sua geração, no preciso momento em que ela surge - a sua origem - estamos a otimizar todo um processo, independentemente do número de etapas ou tarefas que este tenha. Ou seja, nesta vertente, o *Lean* acaba por voltar um pouco à sua raiz, de onde efetivamente surgiu: uma linha de montagem automóvel, agora "convertida" para uma linha de processamento de encomendas.

Durante a apresentação do capítulo 6, onde são descritos todos os requisitos funcionais e os requisitos não funcionais da aplicação, ficou claro que, através da definição de prioridades para os requisitos estipulados, nem tudo tinha a mesma importância no que toca à sua implementação. Alguns por falta de contexto, outros porque não representam necessidade imediatas, mas sim funcionalidades que trariam valor acrescentado num futuro próximo. Assim, estas funcionalidades irão convergir para uma versão 2.0 da aplicação, abordando já o tema que recai sobre desenvolvimentos e trabalho futuro.

Como apresentado anteriormente, as cartas de porte da DHL Express são constituídas por três códigos de barras. O foco desta aplicação reside no primeiro código de barras que é o utilizado para codificar o número de uma carta de porte. Ainda assim, existem mais dois, cada um responsável por transmissão de informação para a rede DHL. Apesar de não ser um problema de primeira instância, estes códigos de barras também podem ser problemáticos no que toca a leituras automatizadas. Futuramente, a abordagem a seguir aqui poderá ser a de dar ao utilizador a opção de selecionar que código de barras em concreto este deseja validar, disponibilizando essa opção no menu de lançamento da câmara do dispositivo. De forma a ter uma aplicação o mais versátil e vanguardista possível, dados os atuais avanços tecnológicos neste campo e dada a sua facilidade de utilização, estas funcionalidades serão implementados através de tecnologias de reconhecimento de caracteres, vulgarmente conhecido como OCR, ou *Optical Character Recognition*.

Posto isto, pode então surgir ainda a questão de "porquê utilizar esta via?" e não uma outra abordagem? Todo este desenvolvimento foi feito com uma abordagem, pensamento e metodologia *Lean*. Um dos princípios chave desta abordagem desenvolvido e aprofundado no capítulo 3 é a questão da eficiência e da eliminação do desperdício a qualquer custo. Podemos ter uma visão extremamente eficiente ao notarmos que toda a aplicação já é executada com recurso à câmara do dispositivo do utilizador. Ou seja, a câmara é um dos pontos fundamentais já desenvolvidos que permite que a aplicação opere conforme é suposto. Assim, pode ser feito um reaproveitamento do que já foi implementado, para suprir eventualmente esta necessidade futura, devendo apenas diferir nos processamentos lógicos de validação, que, desta vez, serão feitos analisando caracteres, e não proporções de códigos de barra, capturados pela câmara.

Um outro aspeto que pode ser considerado como um eventual trabalho futuro, será recorrer à validação OCR através da obtenção de informação acerca dos códigos de barras recorrendo a *Web services*. Mas de que forma uma implementação de invocações a *Web services* melhorará a aplicação em si? A API da DHL disponibiliza uma funcionalidade conhecida como *addressValidate*. Esse serviço permite validar se os dados inseridos estão conforme, no sentido de existir um mapeamento correto entre código-postal e cidade inserido pelo utilizador. Ao implementar esta funcionalidade, não só teríamos uma aplicação capaz de validar códigos de barra e as suas proporções como seria também uma aplicação capaz de detetar dados inválidos logo na sua origem, o que ao fim ao cabo, iria apenas trazer vantagens à operação da empresa evitando atrasos por dados incorretos.

8

# 8. Bibliografia


Ahmad, M. O., Markkula, J., & Oivo, M. (2013). Kanban in software development: A systematic literature review. *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*, 9–16. https://doi.org/10.1109/SEAA.2013.28

Balsamiq. (n.d.). *Quick and Easy Wireframing Tool*. https://balsamiq.com/wireframes/

Barcodes Inc. (n.d.). *Advantages of Barcodes*. Retrieved October 17, 2020, from https://www.barcodesinc.com/articles/advantages-of-barcodes.htm

Codeacademy. (n.d.). *Learn Java*. Retrieved July 11, 2020, from https://www.codecademy.com/learn/learn-java

Coltman, T., Gattorna, J., & Whiting, S. (2010). Realigning service operations strategy at DHL express. *Interfaces*, *40*(3), 175–183. https://doi.org/10.1287/inte.1100.0491

Computerweekly. (2002). *Write once, run anywhere?* https://www.computerweekly.com/feature/Write-once-run-anywhere

Console, F. (n.d.). *Authentication*. Retrieved October 27, 2020, from https://console.firebase.google.com/u/0/project/dhl-ocr-label-scanner/authentication/providers

Curt Hibbs, S. J. & M. S. (2009). The Art of Lean Software Development: A Practical and Incremental Approach. In *O`Reilly*.

Deutsche Post DHL Group. (n.d.-a). *CO- AND ADDITIONAL BRANDING BUSINESS UNITS*. Retrieved September 8, 2020, from https://www.dpdhl-brands.com/dhl/en/guides/co-and-additional-branding/business-units.html

Deutsche Post DHL Group. (n.d.-b). *History 1969 to present*. Retrieved September 8, 2020, from http://wap.dhl.com/info/history.html

Developers, G. (2015). *Introducing Bar Code and QR Code Detection in the Google Vision APIs (100 Days of Google Dev)*. https://www.youtube.com/watch?v=kAoDW3gm8FM&feature=emb_title

DiMarzio, J. F. (2016). Beginning Android® Programming with Android Studio. *Beginning Android® Programming with Android Studio*. https://doi.org/10.1002/9781119419334

Ferreira, K. (2020). *Como escolher um método para priorização do seu Backlog*. Como Escolher Um Método Para Priorização Do Seu Backlog. https://medium.com/@katlenaferreira/como-escolher-um-método-para-priorização-do-seu-backlog-d50b729ac6e9

Firebase, G. (n.d.). *Uma plataforma de desenvolvimento de apps abrangente*. Retrieved October 27, 2020, from https://firebase.google.com/

Google. (n.d.). *Barcode Detection with the Mobile Vision API*. Retrieved September 17, 2020, from https://codelabs.developers.google.com/codelabs/bar-codes/#0

Google. (2017). *Barcode API Overview*. https://developers.google.com/vision/android/barcodes-overview

Google. (2020). *Barcode Scanning*. https://developers.google.com/ml-kit/vision/barcode-scanning

Grande Consumo. (2019). *DHL Express abre primeira loja própria em Portugal*. https://grandeconsumo.com/dhl-express-abre-primeira-loja-propria-em-portugal/#.X1lKMOeSmUk

Guévin, M. (n.d.). *Os 8 Principais Métodos de Gerenciamento de Projetos, Abordagens e Técnicas*. Nutcache. https://www.nutcache.com/pt-br/blog/os-8-principais-metodos-de-gerenciamento-de-projetos-abordagens-tecnicas/

Jackson, W. (2017). *Android Apps for Absolute Beginners*. https://doi.org/10.1007/978-1-4842-2268-3

Kaur, A. (2018). App Review: Trello. *Journal of Hospital Librarianship*, *18*(1), 95–101. https://doi.org/10.1080/15323269.2018.1400840

Kumar, A. (2018). *Mastering Firebase for Android*.

Marcela, A. (2019). *DHL Express abre primeira loja em Portugal. E tem mais duas previstas este ano*. https://www.dinheirovivo.pt/empresas/dhl-express-abre-primeira-loja-em-portugal-e-tem-mais-duas-previstas-este-ano/

Moroney, L. (2015). *Barcode Detection in Google Play services*. https://android-developers.googleblog.com/2015/08/barcode-detection-in-google-play.html

Moroney, L. (2017). The Definitive Guide to Firebase. In *The Definitive Guide to Firebase*. https://doi.org/10.1007/978-1-4842-2943-9

Ostergaard, K. (2016). Applying Kanban principles to electronic resource acquisitions with Trello. *Journal of Electronic Resources Librarianship*, *28*(1), 48–52. https://doi.org/10.1080/1941126X.2016.1130464

Pires, R. (2019). *Aprenda a usar a técnica MoSCoW nos projetos da sua agência!* Aprenda a Usar a Técnica MoSCoW Nos Projetos Da Sua Agência! https://rockcontent.com/br/blog/metodo-moscow/

Poppendieck, B. M., & Poppendieck, T. (2003). Lean software development: an agile toolkit Software Development Managers. In *Computer* (Vol. 36, Issue 8). https://doi.org/10.1109/MC.2003.1220585

Poppendieck, M. (2010). Lean Software Development Principles. *October*, 1–2. https://doi.org/10.1201/EBK1439817568-12

Poppendieck, M. (2016). Lean software development. *The Routledge*

*Companion to Lean Management*, 392–402. https://doi.org/10.4324/9781315686899

Prabhu, R. (2019). *Java Programming Basics*. https://www.geeksforgeeks.org/java-programming-basics/

Pro, S. (2019). *React Native vs iOS/Android Native: what is the best choice for mobile development today?* https://medium.com/@smartym.pro/react-native-vs-ios-android-native-what-is-the-best-choice-for-mobile-development-today-67cde42fc8bc

Programiz. (n.d.). *Learn Java Programming*. Retrieved July 11, 2020, from https://www.programiz.com/java-programming

Reis, J. (2019a). *DHL Express abre primeira loja em Portugal. E tem mais duas previstas este ano*. https://www.dinheirovivo.pt/empresas/dhl-express-abre-primeira-loja-em-portugal-e-tem-mais-duas-previstas-este-ano/

Reis, J. (2019b). *DHL Express simplifica serviço GoGreen*. https://www.logisticaetransporteshoje.com/transportes/dhl-express-simplifica-servico-gogreen/

Transportes, H. L. &. (2019). *DHL Express simplifica serviço GoGreen*. Jorge, Victor. https://www.logisticaetransporteshoje.com/transportes/dhl-express-simplifica-servico-gogreen/

Trnka, D. (2018). *Mobile App Development: React Native vs Native (iOS, Android)*. https://medium.com/mop-developers/mobile-app-development-react-native-vs-native-ios-android-49c5c168045b

Wojtek Kalicinski. (2020). *Criar uma IU responsiva com o ConstraintLayout*. https://developer.android.com/training/constraint-layout