INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

SmartStock – Gestão de Stocks

Mestrado em Engenharia de Software

Dissertação efetuada sob orientação de
Professor Doutor Pedro Miguel Faria
Professora Doutora Sara Paiva

Pedro Daniel Matos Barbosa

SmartStock – Gestão de Stocks
Pedro Daniel Matos Barbosa

2019

Escola Superior de Tecnológica e Gestão

INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

Pedro Daniel Matos Barbosa

SmartStock – Gestão de Stocks

Mestrado em Engenharia de Software

Dissertação efetuada sob orientação de

Professor Doutor Pedro Miguel Faria

Professora Doutora Sara Paiva

fevereiro de 2019

AGRADECIMENTOS

Desejo exprimir os meus agradecimentos a todos aqueles que permitiram que esta dissertação se concretizasse.

Em primeiro lugar quero agradecer ao Professor Doutor Pedro Miguel Faria, da Escola Superior de Tecnologia e Gestão de Viana do Castelo, por me ter recebido no seu grupo de trabalho, e desde o início, e perante as minhas dificuldades, acreditado sempre nas minhas capacidades. Ainda a disponibilidade e motivação quando o esmorecimento se instalava e pelo trato correto e científico em todas as fases desta jornada.

Agradeço, de igual forma, à Professora Doutora Sara Paiva, da Escola Superior de Tecnologia e Gestão de Viana do Castelo pela ajuda, pelo profissionalismo e atenção disponibilizada.

Agradeço a todos os meus amigos, em especial ao João Ribeiro a amizade demonstrada, o seu sempre inteiro dispor, e ainda ser um colega inseparável, que está sempre nos bons e menos bons momentos.

À minha família, especialmente à minha esposa e ao meu filho pelos quais me esforço para ser sempre melhor. Aos meus pais, irmão, avó e tio, que sempre me apoiaram ao longo da minha vida. O meu muito obrigado!

RESUMO

Otimizar o controlo dos processos é cada vez mais importante na gestão de eventos. Nos dias de hoje, uma gestão eficaz associada a uma redução nos custos de operação, torna-se uma mais-valia, pois permite às empresas de eventos obterem maior competitividade e crescimento focando-se no que realmente importa.

Todo o evento de sucesso, tem como base uma empresa sólida e organizada. Assim, neste projeto propõe-se apresentar uma solução colaborativa e funcional de gestão de stocks para empresas de eventos, que fará uso de uma aplicação móvel, que irá flexibilizar a gestão dos produtos num armazém e organizá-los de forma eficiente, através de múltiplas funcionalidades otimizadas e integradas nas soluções deste projeto, controlando assim o ciclo de vida do produto, desde a sua entrada no armazém até ao regresso de um evento. Os resultados serão os bens disponibilizados no exato momento em que deles se necessite.

A estratégia adequada equilibrará a compra de produtos segundo as necessidades sentidas na organização.

ABSTRACT

Optimizing process control is increasingly important in an event management. Nowadays, efficient management associated with a reduction in operating costs is an added value, as it enables event companies to achieve greater competitiveness and growth by focusing on what it really matters.

Every successful event is based on a solid and organized company. Thus, this project proposes to present a collaborative and functional stock management solution for event companies, which will make use of a mobile application and a web management platform, that will make product management flexible in a warehouse and organize them in an efficient way, through multiple optimized functionalities and integrated in the solutions of this project, in order to control the life cycle of the product, from its entrance in the warehouse until its return of an event. The result will be the goods made available at the exact moment they are needed.

The appropriate strategy will balance the purchase of products according to the needs felt in the organization.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SIGLAS E ACRÓNIMOS

Termo	Descrição
CSS	Cascading Style Sheets
PHP	PHP: Hypertext Preprocessor
Laravel	Framework PHP
MYSQL	Open-source relational database management system
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
URL	Uniform Resource Locator
Slug	Parte de uma URL legível

1. INTRODUÇÃO

Com esta dissertação pretende-se desenvolver uma aplicação móvel que dê resposta à gestão de stocks em empresas de eventos. De uma forma geral, pretende-se uma solução que permita a um colaborador efetuar o inventário de produtos e acompanhar os movimentos de entrada e saída de material durante a realização dos eventos.

Neste primeiro capítulo é dada a conhecer a motivação e contextualização do problema que conduziu à realização desta dissertação. De seguida, é apresentado o problema e objetivos a que esta dissertação pretende responder.

Por fim, o capítulo termina com a descrição da estrutura do documento.

1.1 Contexto e Motivação

Nos dias de hoje, e com a constante presença das tecnologias torna-se indispensável a utilização das ferramentas disponíveis para a melhoria do decorrer das atividades empresariais. Cada vez mais as empresas têm necessidade de controlar os custos, melhorar o desempenho e preservar o meio ambiente, poupando recursos que serão necessários para as gerações vindouras.

No contexto da atividade profissional do autor desta dissertação, e no seguimento da sua colaboração com uma média empresa do ramo da organização de eventos, este deparou-se com a dificuldade sentida pelos colaboradores da empresa na gestão dos produtos utilizados em cada evento.

Posto isto, e com a perceção que as empresas de eventos têm dificuldades em gerir as entradas e saídas de material e respetivas baixas de stock, torna-se necessário utilizar tecnologia de forma a encontrar uma solução para um problema que persiste durante anos e criar uma forma colaborativa de simplificar um processo difícil e com poucos resultados.

Depois de uma primeira pesquisa à procura de aplicações móvel, chegou-se à conclusão de que existe uma lacuna no mercado e que as aplicações existentes na gestão de stocks são de difícil utilização e demasiado complexas, não

respondendo assim ao problema concreto que se pretende resolver com esta dissertação. Tal como já referido, pretende-se acima de tudo uma aplicação colaborativa que simplifique o processo de gestão e movimentação de stocks.

Esta aplicação será direcionada para o mercado de eventos podendo no futuro ser alargado a outras áreas de negócio.

1.2 Problema

Com o exponencial crescimento do número de eventos de uma empresa e com timings entre estes, muito curtos, torna-se cada vez mais difícil o controlo de stocks.

Gerir stocks em empresas de eventos implica efetuar a gestão do inventário e controlar as entradas e saídas de material para eventos.

O foco do problema recai sobre a necessidade de qualquer elemento dentro de uma empresa, efetuar o levantamento de material sem o controlo necessário e sem uma guia que identifique onde se encontra o produto. Este processo de controlo e movimentação de stocks individual torna difícil a gestão, pois temos o controlo repartido e descentralizado, não tendo um ponto de união de stocks.

Cada colaborador faz o seu próprio controlo, tornando-se num processo moroso, dispendioso e com uma possibilidade de erros crescente.

1.3 Objetivos

Esta dissertação tem como objetivo principal desenvolver uma aplicação móvel colaborativa que permita a gestão de stocks e que efetue o controlo das movimentações de material para um determinado evento.

Pretende-se assim uma aplicação que responda aos seguintes requisitos:

- Simplificar o processo de gestão aos colaboradores de empresas de eventos.
- Possibilitar o registo de produtos e respetivas quantidades.
- Efetuar entradas e baixas de stock.

- Efetuar movimentações de produtos do armazém para os eventos e vice-versa
- Acompanhar as movimentações dos produtos utilizados em cada evento.

No final deste projeto será testada a usabilidade da aplicação e efetuada a avaliação dos resultados obtidos.

1.4 Estrutura da dissertação

Esta dissertação encontra-se dividida por seis capítulos, sendo eles a Introdução (Capítulo 1), Estado da Arte (Capítulo 2), Análise e Especificação de Requisitos (Capítulo 3), Desenvolvimento (Capítulo 4), Avaliação dos Resultados (Capítulo 5) e Conclusão e Trabalho Futuro (Capítulo 6). Pretende-se assim assegurar uma maior clareza e entendimento progressivo do documento.

No capítulo 1 apresenta-se a motivação, contextualização, problema e exposição dos objetivos deste projeto.

O capítulo 2 versa a análise do estado da arte. Começa-se pela análise da prospeção do mercado no que concerne às aplicações existentes, seguindo-se a apresentação de comparações e análises relevantes para o desenvolvimento da aplicação.

O capítulo 3 centra-se no estudo inicial da solução, onde são apresentadas análises e especificados os requisitos, diagramas e fluxogramas, casos de uso, modelo de classes e protótipos da aplicação móvel.

No capítulo 4 apresenta-se as tecnologias utilizadas para o desenvolvimento do projeto e descreve-se de forma pormenorizada a implementação da aplicação.

O capítulo 5 corresponde à análise e apresentação de resultados. Esta análise será efetuada através de testes na aplicação, realizados por colaboradores de empresas de eventos e logística. Pretende-se assim obter feedback de utilizadores reais e identificar possíveis melhorias.

Por fim, no capítulo 6, são expostas as conclusões. Apresenta-se a reflexão do trabalho desenvolvido e perspetivas futuras de desenvolvimento.

2. ESTADO DA ARTE

2.1 Introdução

Sendo o tema deste projeto direcionado para as novas tecnologias, torna-se necessário na fase de levantamento de requisitos fazer uma primeira abordagem dos conceitos relacionados com o tema. Com este estudo, pretende-se apurar quais as melhores práticas de desenvolvimento e implementação para elaborar este projeto de forma eficaz. Este estudo é fundamental, pois ajuda ao entendimento mais profundo do tema, podendo assim adquirir os pressupostos iniciais.

Nesta análise do estado da arte, serão dadas a conhecer algumas aplicações já existentes, fazendo uma comparação entre elas, podendo assim demonstrar algumas vantagens e desvantagens. Serão também referidos alguns artigos relacionados com o tema em questão.

2.2 Aplicações existentes no mercado

Neste subcapítulo serão abordadas algumas aplicações existentes atualmente no mercado e semelhantes à que se pretende desenvolver. Foram instaladas e testadas várias aplicações na área de gestão de stocks, mas apenas as apresentadas abaixo se aproximaram do objetivo pretendido para este projeto.

Stock Controller

Esta solução (Figura 1), desenvolvida pela empresa portuguesa XNR Sisbi, encontra-se disponível na *Play Store* (Sisbi, 2018) na versão 5.1.5. Após análise a esta aplicação conclui-se que esta tem algumas funcionalidades chave para uma aplicação neste ramo, contudo a interação com esta torna-se confusa para o utilizador. É de salientar que a mesma permite a gestão colaborativa e contém uma plataforma de gestão web, no entanto, não foi possível usar pois era necessário adquirir este módulo. A mesma apenas possuí a leitura de produtos através do

barcode, faltando assim a leitura via *QR Code*. Alguns pontos fortes identificados desta aplicação:

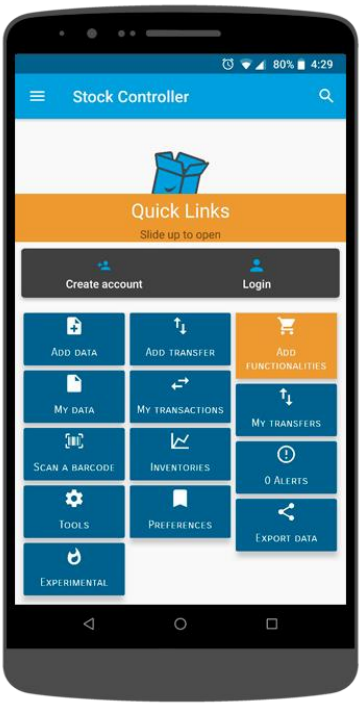

Figura 1 - Aplicação Stock Controller

- Documentos de Entrada e Saída de stock
- Gestão Produtos
- Leitor de *Barcode*
- Gestão colaborativa
- Possibilidade de exportar dados via PDF ou CSV

Stock and Inventory

Esta solução (Figura 2) é desenvolvida pela Chester Software e encontra-se atualmente na versão 2.0.12 disponível na Play Store (Chester SW, 2018). Esta aplicação é bastante completa e focada no tema. Contudo, a gestão de inventário e respetivos documentos são de difícil leitura o que provavelmente fará com que alguns utilizadores desistam da mesma após a primeira utilização.

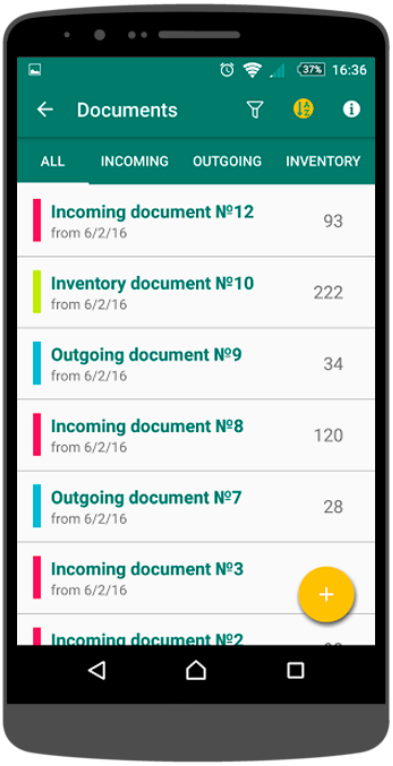

Figura 2 - Aplicação Stock and Inventory

Apesar da mesma ter documentação para o utilizador, nem sempre é fácil entender como este módulo da aplicação funciona. No entanto foram identificados alguns dos pontos fortes:

- Gestão de Produtos
- Documentos de Entrada e Saída de stock
- Possibilidade de exportar os produtos para documento Excel
- Leitor de *QR Code*
- Cópias de Segurança

Mobile Stock

Mobile Stock (Figura 3), é uma aplicação da empresa portuguesa Wave Solutions, que está ligada ao desenvolvimento de aplicações de mobilidade empresarial. Está encontra-se disponível na Play Store (Wave Solutions, 2018a).

Salienta-se a organização dos documentos de entrada e saída de stock e o aspeto cuidado da aplicação. Ao efetuar testes não foi possível a utilização da funcionalidade do *barcode*.

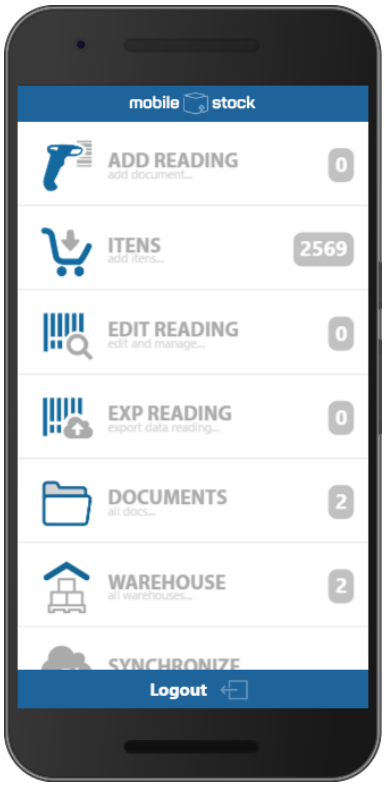

Figura 3 - Aplicação Mobile Stock

Das funcionalidades disponíveis, são de salientar as seguintes:

- Gestão de produtos
- Documentos de Entrada e Saída de stock
- Armazéns

De forma a finalizar a análise destas 3 soluções, foi elaborada uma tabela onde é possível identificar as diferentes funcionalidades de cada solução, pretende-se assim, verificar até que ponto estas respondem aos requisitos identificados para a gestão de stocks em empresa de eventos.

Tabela 1 - Resumo Comparativo de Abordagens Existentes

	Stock Controller	Stock and Inventory Simple	Mobile stock
Inventário	Sim	Sim	Sim
Gestão de Produtos	Sim	Sim	Sim
Categorias de Produtos	Sim	Sim	Sim
Leitor *Barcode*	Sim	Sim	Sim
Leitor *QR Code*	-	Sim	Sim
Gestão Colaborativa	Sim	-	-
Eventos	-	-	-
Movimentação de Produtos	-	-	-
Real Time	-	-	-

(Wave Solutions, 2018b),(Chester SW, 2018),(Sisbi, 2018)

Com esta análise, é possível verificar que as aplicações dispõem de funcionalidades comuns para efetuar a gestão de stocks, no entanto, nenhuma delas preenche os requisitos necessários para acompanhar as movimentações de produtos e apenas uma fornece a possibilidade de efetuar uma gestão colaborativa.

2.3 Tecnologias de desenvolvimento

Devido à constante evolução, existem inúmeras tecnologias para o desenvolvimento de todo o tipo de aplicações. Neste subcapítulo pretende-se estudar e dar a conhecer algumas tecnologias, que sejam úteis à construção da *RESTful API* e aplicação móvel. Pretende-se encontrar tecnologias atuais, ricas em documentação e *open-source*. Sendo assim, e por ordem lógica serão descritas as possíveis tecnologias para o desenvolvimento da *RESTful API*, passando depois para as tecnologias *mobile*.

2.3.1 RESTful API's

Atualmente, é comum existirem aplicações que funcionam exclusivamente pela Internet, estas podem ser utilizadas por diferentes tipos de plataformas e clientes. Por outro lado, temos sistemas que precisam de receber e tratar vários tipos de dados. Dada esta problemática, de comunicação entre clientes e sistemas foi necessário desenvolver uma solução que permitisse a comunicação entre ambos. Ao longo do tempo foram surgindo várias aplicações que ficaram conhecidas com *APIs*.

API

API – *Application Programming Interface,* tem como conceito um conjunto de rotinas e padrões estabelecidos e documentados por uma aplicação, que pretende dar a possibilidade a outras de conhecer as suas funcionalidades, sem precisar de entender os detalhes da implementação. Posto isto, verificamos que as *APIs* permitem uma interoperabilidade entre aplicações.

REST vs RESTful

De forma a seguir um conjunto de princípios para o correto desenvolvimento de uma API, foi analisada a arquitetura *REST (Representation State Transfer)* que é um modelo de arquitetura que foi descrito por Roy Fielding, um dos principais criadores do protocolo *HTTP*, na sua tese de doutoramento. REST é um conjunto

de princípios, que quando aplicados de maneira correta numa aplicação, beneficiam com a arquitetura e padrões da Web. RESTful é a capacidade de determinado sistema aplicar os princípios de *REST* (Roy, 2000).

Potenciais frameworks para o desenvolvimento da RESTful API

Foram estudadas três *frameworks PHP*, *Laravel* (Laravel, 2018)*, Symfony* (Symfony, 2018) e *Lumen* (Lumen, 2018) de forma a identificar vantagens e desvantagens de cada uma.

Como base, as três *frameworks* utilizam *PHP*, linguagem de programação que atua do lado do servidor, tendo sido criada em 1994 por Rasmus Lerdorf primeiramente para o desenvolvimento web. Normalmente, é processada por um interpretador que é instalado em servidores web tais como *apache* e *Nginx*. A flexibilidade do *PHP* contribui para popularidade deste sendo assim utilizado em mais 70% dos servidores web (Usage Statistics, 2018).

De entre muitos dos benefícios de usar *frameworks*, destaca-se o rápido desenvolvimento, organização de código, separação entre lógica e apresentação, boas práticas e um conjunto de ferramentas vastas.

O *Laravel* é um dos mais populares *frameworks* de desenvolvimento de aplicações web. Criado por Taylor Otwell em 2011, foi projetado para facilitar o desenvolvimento da web e segue o padrão *model-view-controller* (MVC). No momento da escrita desta dissertação a versão atual é a 5.8.

Symfony é outro *framework* de desenvolvimento web muito popular. Foi lançado em 2011 e a versão no momento da escrita desta dissertação é a 4.5. Este dispõem de um conjunto de componentes *PHP* reutilizáveis, que permite construir aplicações de alto desempenho.

Lumen é uma *micro-framework PHP* desenvolvida por Taylor Otwell, criador da *framework Laravel*. Desenhada para construir *micro-services* e *API´s* rápidas que beneficiam do poder de *Laravel*, trocando algumas funcionalidades por um aumento de velocidade. Sendo *Lumen* baseado em *Laravel*, herda grande parte das funcionalidades, tais como, *ORM, MVC Pattern*, *routes* simples, autenticação e sessões. Além disto e caso seja necessário é possível mover o projeto *Lumen* para

Laravel de uma forma simples. No momento da escrita deste documento, a última versão do *Lumen* era a 5.7.0.

Comparação entre *frameworks*

Para uma melhor escolha, apresenta-se agora algumas comparações das *frameworks* estudadas.

Tabela 2 - Comparação entre as Frameworks Laravel e Symfony

Fatores de diferenciação	***Laravel***	***Symfony***
Performance	- O tempo médio para carregar um website é de cerca de 60 milissegundos	- O tempo médio para carregar um website é de cerca de 250 milissegundos
Base de dados e Migração	- *Eloquent* - Manual e não requer a definição do campo no modelo	- *Doctrine* - Automático, mas é necessário definir o campo no modelo
Tempalte Engine	*Blade, PHP, Smarty, Twig*	*Twig, PHP*
Base de dados suportadas	*SQLite, MySQL, PostgreSQL, Redis, Microsoft BI, MongoDB*	*Microsoft BI, MySQL, NoSQL, PostgreSQL, CouchDB, DynamoDB, MongoDB, MemcacheDB, GraphDB Membase, GemFire, Oracle, Apache Jackrabbit*
Injeção de dependência	Automático	Manual
Suporte *WebSockets*	Sim (*Laravel WebSockets* e *Ratchet*)	Sim (*Ratchet*)
Curva de aprendizagem	Média	Alta

Principais vantagens da *framework Symfony*

- Múltiplas opções de extensibilidade
- Mais rápido que a maioria dos *frameworks PHP*
- Adaptabilidade e desempenho ideal (solicitação / resposta centrada em HTTP)
- Componentes reutilizáveis que reduzem tempo e custo

Principais vantagens da *framework Laravel*

- Mecanismo avançado de *query builder*
- Migração de dados e gestão geral simples
- Facilidade de *auto-loading* (não é necessária manutenção manual)
- Sistema de autenticação simples

(Gusti, 2018)

Destaca-se desta comparação a performance e a *query builder* do *Laravel* em comparação com *Symfony*, uma vez que estas serão importantes para o desenvolvimento da API.

Na comparação realizada Tabela 3, verifica-se que *Lumen* é uma versão leve de *Laravel* e aparenta ter uma melhor performance, no entanto é espectável que *Lumen* tenha algumas limitações em relação à *framework Laravel*, sendo o seu propósito direcionado para micro serviços.

Tabela 3 - Comparação entre as Frameworks Laravel e Lumen

	Laravel	***Lumen***
Definição	*Framework* web que utiliza a arquitetura de software *MVC*.	*Micro-Framework* para aplicações web que permite criar micro serviços.
Uso	Com base no *PHP*	É uma versão leve do *Laravel*
Plataforma	Suportada em qualquer tipo de sistema operacional ou plataforma	Suportada em qualquer tipo de sistema operacional ou plataforma pois esta é uma derivação do *Laravel*
Pedidos	Suporta menos pedidos por segundo	Consegue lidar com mais pedidos por segundo
Licença	É licenciada sob licença do MIT	É licenciada sob licença do MIT
Tempo de Resposta	Maior	Menor
Packages	Grande variedade	Grande variedade
Suporte *WebSockets*	Sim (*Laravel Websockets* e *Ratchet*)	Sim (*Ratchet*)
Curva de aprendizagem	Média	Média

(educba, 2017)

Em suma, cada aplicação web é diferente e com necessidades diferentes. A escolha final recairá sempre sobre a *framework* que preencha a maior parte dos requisitos para o desenvolvimento do projeto.

2.3.2 WebSockets

WebSockets é uma tecnologia que permite a comunicação bidirecional por canais *full-duplex* através de uma única conexão *TCP (Transmission Control Protocol)* entre o cliente e o servidor em tempo real, resultando assim numa conexão com latências muito reduzidas. (Erkkilä, 2012)

WebSockets vs HTTP

Os *WebSockets* representam uma evolução muito esperada na tecnologia web entre comunicações cliente/servidor. Sendo os pedidos *HTTP* mais convencionais, os *WebSockets* têm como principal diferença não seguirem o método mais convencional e tradicional de pedido-resposta, em vez disso quando existe uma conexão *WebSockets* entre o cliente e o servidor, ambos os *endpoints* podem enviar de forma assíncrona dados de um para o outro, esta conexão permanece aberta e ativa até que uma das partes feche a sua conexão, conforme ilustrado Figura 4.

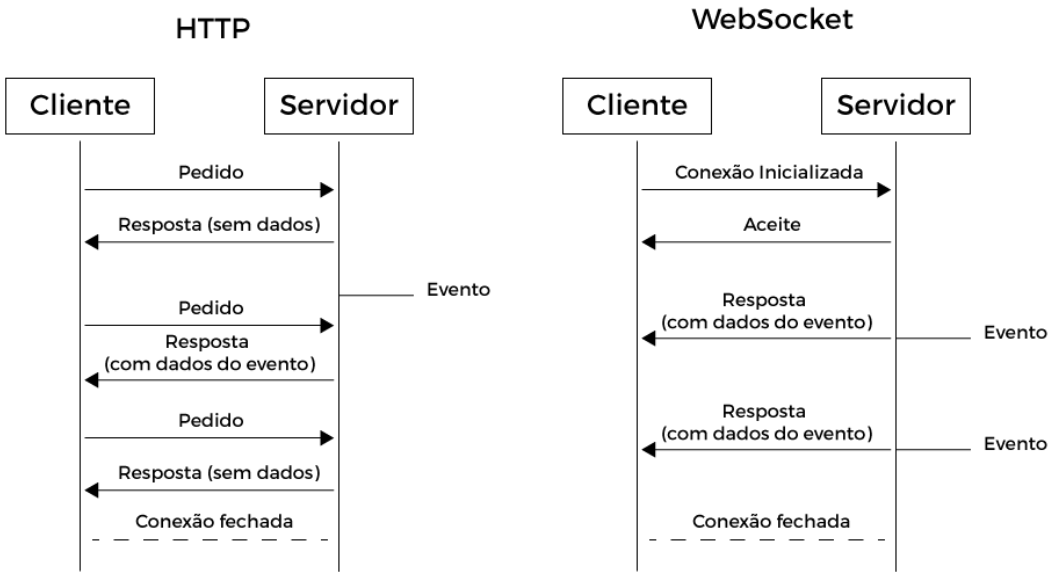

Figura 4 - WebSockets vs Http

A especificação do protocolo *WebSocket* é definido por dois tipos de esquema *URL*, *ws (Web Socket)* e *wss (Secure Web Socket),* para conexões não criptografadas e criptografadas respetivamente. (Mike & Rahman, 2011)

Ratchet vs Laravel Websockets

Ratchet é um package *PHP* para lidar com *WebSockets*. Permite criar aplicações bidirecionais em tempo real entre clientes e servidores através de *Websockets*. Ao contrário do código normalmente utilizado em *PHP*, que são scripts de curta duração, os scripts do *Ratchet* são um pouco diferentes. Cada página web assim que é carregada no lado do cliente, executa no lado do servidor um novo script *PHP*, carrega recursos, executa o código, fecha recursos, envia a saída (*HTML*) para o cliente e fecha a conexão. Esta é a natureza do protocolo HTTP. Já com *Websockets* e como resultado da utilização de *PHP* em *Ratchet*, apenas um script é executado e as conexões perecem abertas. De salientar que não é possível utilizar variáveis globais, pois o contexto do processo de execução não está limitado a uma única conexão. (socketo, 2019)

Laravel WebSockets, desenvolvido por Marcel Pociot da empresa *beyondcode*, é um package para *Laravel* que permite manipular um servidor de *WebSockets*. Este foi desenvolvido sobre o package *Ratchet* e pretende facilitar a integração de *WebStockes* em projetos *Laravel*. *Laravel WebSockets*, dispõem de múltiplos recursos de fácil utilização, dos quais se salienta a utilização de um único servidor de *WebSockets* para diversas aplicações, e um painel de depuração em tempo real, onde é possível analisar o número de conexões e a quantidade de mensagens enviadas e recebidas. (Pociot & Herten, 2018)

Em conclusão, salienta-se a possibilidade de aplicar *WebSockets* em qualquer um dos *frameworks* citados e analisados no subcapítulo 2.3 da presente dissertação. Do estudo feito entre os *packages* supracitados, destaca-se *Laravel WebSockets*, que permite criar uma abstração sobre o *package Ratchet* e assim disponibilizar muitos recursos interessante que levariam muito tempo a ser implementados ao usar *Ratchet*.

2.4 Aplicações Móveis

A quantidade de aplicações móveis existentes tem vindo a crescer de forma geral, como podemos verificar no gráfico da Figura 5:

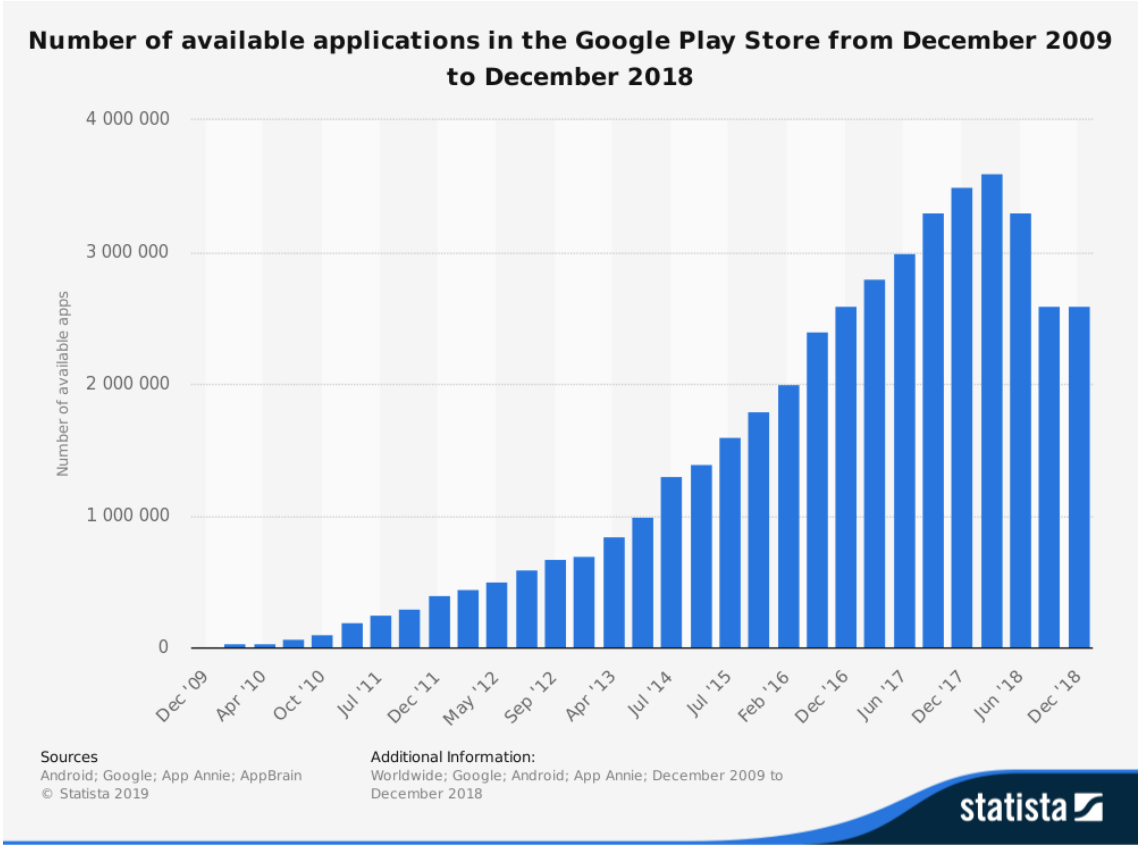

Figura 5 - Número de apps móveis disponíveis no Google Play entre 2009 e 2018

(statista, 2018)

Atualmente, devido à extensa oferta do mercado existem inúmeras possibilidades ao desenvolvimento de aplicações mobile para Android, iOS ou Windows Phone. Com a evolução destas tecnologias sugiram também alternativas ao desenvolvimento de aplicações nativas, sendo estas as aplicações híbridas. Em qualquer tipo de desenvolvimento deve-se sempre ter em conta a limitação de recursos, o seu ecossistema, a experiência do utilizador e a manutenção. Pretende-se agora fazer uma análise a estes dois métodos de desenvolvimento de aplicações.

2.4.1 Aplicações Nativas

As aplicações nativas são programas desenvolvidos a pensar numa plataforma específica, exploram e executam todas as funcionalidades da plataforma. Estes podem ser desenvolvidos para Android, iOS ou Windows Phone. As aplicações, por norma, funcionam de forma prática, simples e otimizada, por sua vez, as estruturações podem ser mais complexas. Na seguinte tabela podemos ver algumas diferenças entre o desenvolvimento para Android nativo e iOS.

Tabela 4 - Android vs iOS Development

Android	IOS
JAVA	SWIFT
Android Studio	XCode
Maior fragmentação (quantidade de aparelhos android superior, dificuldade no preenchimento de requisitos para todos os telemóveis)	Menor fragmentação (quantidade inferior de modelos Iphone, facilidade no preenchimento de requisitos para os vários modelos)
Menor receita com a aplicação publicada	Normalmente consegue-se obter maior receita com a aplicação publicada
Atualmente possui a maior quota de mercado	Atualmente possui a segunda maior quota de mercado
Processo facilitado ao publicar uma aplicação	Processo rígido para poder publicar uma aplicação
Após ser publicada na *Play Store* só pode ser alterada duas horas depois	Após ser publicada na *App Store* só pode ser alterada duas semanas depois
25$ inicias para poder publicar a aplicação na *Play Store*	99$ anuais para poder publicar a aplicação no *App Store*

(Goadrich & P. Rogers, 2011)

Java: As aplicações Android são desenvolvidas em Java, esta é uma linguagem de programação de alto nível e bastante popular, desenvolvida pela Sun

Microsystems posteriormente adquirida em 2009 pela *Oracle Corporation* (Oracle Corporation, 2018).

Swift: Inicialmente as aplicações desenvolvidas para iOS utilizavam a linguagem de programação *Objective C*, atualmente e de forma a facilitar a curva de aprendizagem a Apple desenvolveu o *Swift*. Apesar de ser uma linguagem recente, apresentada na WWDC em 2014, presentemente encontra-se na décima segunda posição do ranking das linguagens de programação mais populares da TIOBE Index (TIOBE, 2018).

XCode: O XCode (Xcode, 2018) é um ambiente de desenvolvimento integrado (IDE), desenvolvido pela Apple em 2003. Atualmente na versão nove, esta ferramenta é essencial para desenvolver aplicações para iOS.

Fragmentação: A fragmentação, no que diz respeito ao desenvolvimento de aplicações para dispositivos digitais, é um grande problema para qualquer *developer*, principalmente para android *developers*. Como citado no artigo publicado pelo *Android Authority* (Android Authority, 2017)*:* Infelizmente, desenvolver para um dispositivo Android significa desenvolver para inúmeros dispositivos Android, isto significa diferentes tamanhos de ecrãs, diferentes *DPIs* e diferentes *aspect ratios*. Para além disso, existe fragmentação (Figura 6) em termos de versões diferentes do sistema Android. Esta situação aplica-se menos ao desenvolvimento para dispositivos iOS, pois como estes apenas são desenvolvidos pela Apple não existem tantos modelos diferentes desta empresa em circulação.

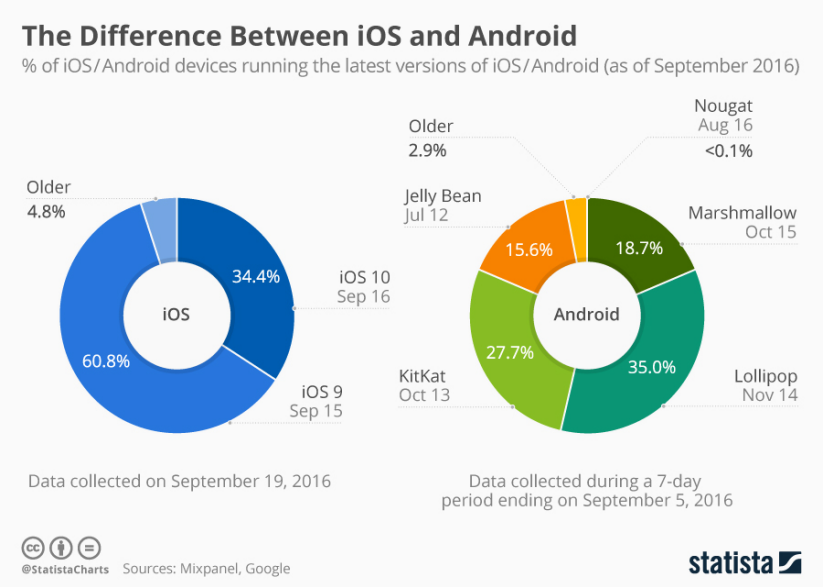

Figura 6 - Diferenças entre versões de dispositivos IOS e Android

(Richter, 2018)

Lucros: Apesar de existirem mais aplicações e downloads realizados na plataforma de distribuição de aplicações da Google, um estudo realizado em 2016 pela *App Annie* (*business intelligence company and analyst firm*), pode-se verificar que existe um maior lucro por parte das aplicações desenvolvidas para iOS. (App Annie, 2016)

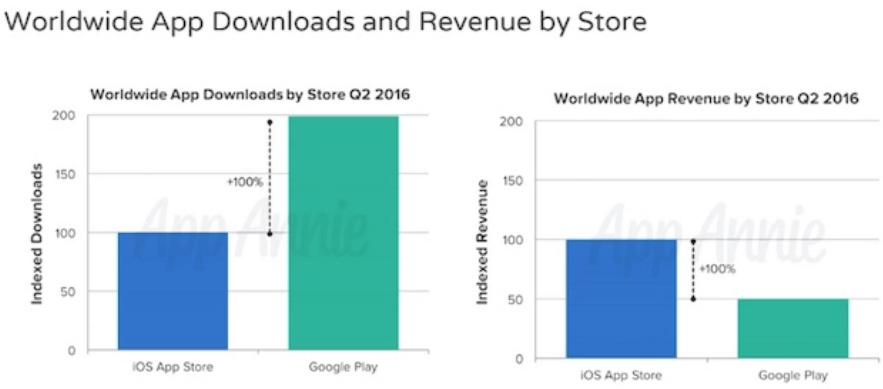

Figura 7 - Downloads e lucros App Store vs Google Play

(App Annie, 2016)

Quota de Mercado: De acordo com a IDC tracker (Figura 8), a Google está bastante à frente no que diz respeito à quota de mercado de dispositivos a circular, podendo-se verificar no gráfico abaixo que este número tem vindo a aumentar ano após ano.

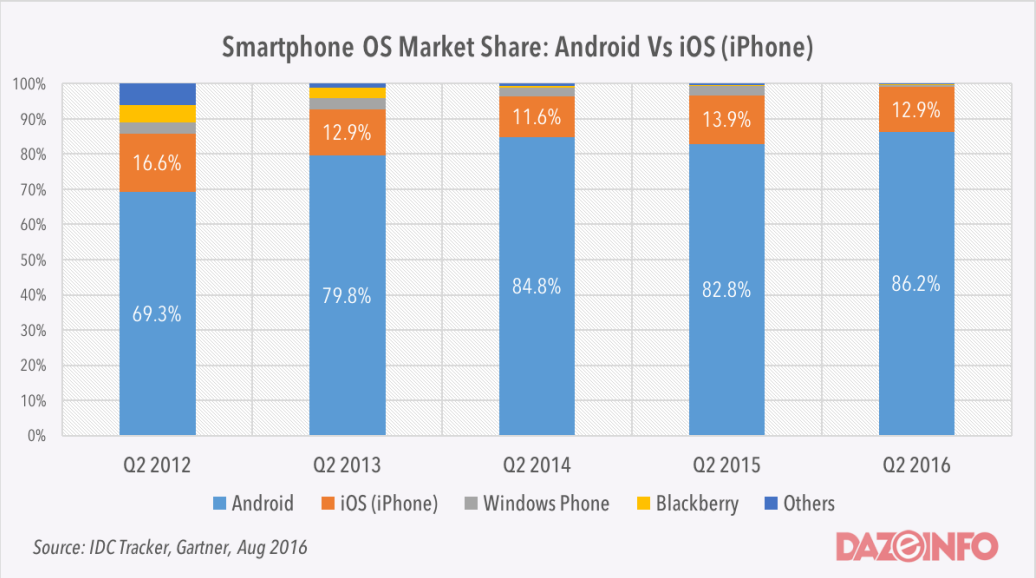

Figura 8 - Quota de mercado: Android vs IOS

(Android Authority, 2017)

Diferenças ao publicar aplicações em diferentes lojas

Existem bastantes diferenças no que concerne à publicação de aplicações, o processo mais trabalhoso é sem dúvida a da Apple, pois inicialmente é necessário estar inscrito no *Apple Developer Program*, de seguida desenvolver a aplicação, só depois é que esta poderá ser enviada para avaliação. No caso desta avaliação ser bem-sucedida, esta é publicada. Este processo é mais facilitado para as aplicações Android, basta desenvolver, adquirir uma conta de distribuição e publicar. A aplicação ficará visível na *Play Store* poucas horas depois desta ser publicada.

Outro fator a realçar é, a alteração de uma aplicação após publicada na *store*, enquanto que, na *Play Store* esta alteração pode ser realizada duas horas após a publicação, na *App Store* apenas duas semanas da mesma ter sido publicada.

Em conclusão, o objetivo principal será poder atingir o maior número de utilizadores e não obter receitas com a aplicação.

2.4.2 Aplicações Híbridas

As aplicações híbridas possibilitam a implementação de uma aplicação que pode ser executada em diferentes plataformas. O desenvolvimento destas permite a redução dos custos e o tempo de implementação, pois não exigem que o programador saiba diferentes linguagens para o desenvolvimento das diferentes plataformas. Existem inúmeras *frameworks* de apoio à criação destas aplicações híbridas, sendo que maioritariamente apoiadas de tecnologias web. Serão agora apresentadas algumas dessas *frameworks*:

React Native

O *React Native* (React Native, 2018) é uma framework *open source* desenvolvida pelo *Facebook* para criar aplicações Android e iOS, esta utiliza *javascript* como linguagem base. Dedica-se a melhorar a eficiência do programador, poupar tempo e dinheiro no desenvolvimento de aplicações. As aplicações desenvolvidas utilizando o *react* são praticamente indistinguíveis das desenvolvidas por linguagens nativas. Uma das grandes vantagens da utilização desta linguagem é a possibilidade da reutilização de componentes em aplicações *web* e *mobile*.

Redux é um container de estado previsível para aplicações *Javascript*, criado por Dan Abramov. Esta resolve problemas de partilha de estado entre componentes, tornando-o unidirecional. O estado do *redux* é divido em quatro partes: *Store*, que é um container de armazenamento do estado geral da aplicação de forma imutável. *Actions*, são informações enviadas da aplicação para a *Store*. Estas são enviadas através de *Actions* Creators, são simples funções que ao serem executadas ativam os *Reducers*. *Reducers*, recebem a informação para ser guardada na *Store*.

Redux Thunk é um *middleware* que permite chamar *actions creators* e em vez de retornarem um *action object*, retornam uma função onde é possível fazer pedidos assíncronos. Esta função recebe o método *dispatch* da *Store* que é utilizado para fazer o *dispatch* regular síncrono, uma vez que a função assíncrona tenha terminado.

Xamarin

Xamarin (Xamarin, 2018) é uma *framework* de desenvolvimento de aplicações híbridas em Android, iOS e Windows Phone, e foi recentemente adquirida pela Microsoft a 24 de fevereiro de 2016 (Matt, 2016). É baseada na linguagem de programação C#, sendo que esta é uma das vantagens principais, pois os programadores que estejam familiarizados com esta linguagem estarão mais à vontade para utilizar esta *framework*.

Ionic

O Ionic (Ionic, 2018) é uma *framework open source* para criar aplicações híbridas, utiliza *Angular* como linguagem base de programação e beneficia de livrarias otimizadas de *HTML*, *CSS* e *Javascript* para dar apoio ao seu desenvolvimento. Uma das principais vantagens é o desenvolvimento rápido, pois através da linha de comando é possível gerar facilmente novas páginas, *providers*, serviços, entre muitos outros.

Tabela 5 - Comparações entre Reactive Native, Xamarin e Ionic

	Reactive Native	Xamarin	Ionic
Linguagem	Javascript	C#	Angular
Open Source	Sim	Não	Sim
Vantagens	Desenvolvimento Rápido	Focado na experiência do utilizador	Desenvolvimento Rápido

	Aplicações desenvolvidas quase indistinguíveis de aplicações Nativas	Performance muito próxima de aplicações Nativas	Fácil aprendizagem
	Utilização de componentes Nativas	Baseada em .net *framework*	Vasta gama de documentação no site oficial
Desvantagens	Algumas limitações a componentes de terceiros	Os tamanhos das aplicações acabam por ser bastante grandes	Não é recomendada para aplicações muito complexas
	Requer alterações diretas no código nativo	Algumas limitações a componentes de terceiros	Segurança por vezes mais fraca do que as aplicações nativas

(Rasmussen, 2018)

Como é possível verificar, o *Xamarin* é tipicamente utilizado por programadores de C#, este oferece uma performance semelhante a aplicações nativas. Relativamente ao *Ionic*, é um *framework* para prototipagem rápida e permite a possibilidade de reutilizar código. Por fim, *React Native* oferece compatibilidade entre plataformas, o que possibilita a criação de aplicação nativas apenas com o uso de *JavaScript*.

2.5 Comparação entre QR Code e Barcode

2.5.1 QR Code

O *QR Code* consiste num código de barras bidimensional que contém informações pré-estabelecidas como páginas web, texto, vídeo ou imagens. A leitura destes é efetuada em grande parte através das câmaras dos *smartphones*. Esta tecnologia tem o potencial de revolucionar a forma como as bibliotecas entregam instruções, conectando assim os clientes à informação. (Coleman, 2011)

2.5.2 Barcode

O *barcode* é provavelmente o mais utilizado e reconhecido mundialmente. Originalmente criado para auxiliar os mercados ajudando estes a aumentar o processo de verificação de saída de produtos. Este é a representação gráfica de uma sequência numérica utilizada para identificar um produto.

Tabela 6 - Diferenças entre QR Code e Barcode

QR Code	Barcode
Bidimensional	Unidimensional
Consegue interpretar códigos desfigurados entre 0 a 30%	Não consegue interpretar códigos desfigurados
Consegue ler códigos na horizontal e vertical	Apenas consegue ler códigos na horizontal
Consegue conter informação acima dos 2000 caracteres	Consegue conter informação entre 20 a 25 caracteres
Utilizado para transmitir e guardar dados de texto, imagens, contactos, entre outros	Maioritariamente utilizado para guardar e transmitir dados associados a produtos

Na Tabela 6, conseguimos identificar inúmeras vantagens da utilização do *QR Code* em relação ao *barcode,* das quais se destaca a interpretação de códigos desfigurados e a leitura de códigos na vertical e horizontal. Estes dois fatores são de extrema importância para o tema, visto que os produtos são reutilizados e as etiquetas ficarão danificadas com o tempo, de igual modo a leitura na vertical e horizontal permite que o produto seja lido quando os produtos estão empilhados.

2.6 Conclusão

Após análise às diferentes soluções existentes, confirma-se a existência de uma lacuna no mercado de uma aplicação que facilite a gestão stocks em empresas de eventos. Todas as empresas querem que o seu negócio cresça e sabe-se, nos

dias de hoje que uma eficiente gestão de stock faz toda a diferença na rentabilidade, na imagem da marca e nos níveis de satisfação dos clientes.

Denota-se uma mais valia para as empresas que adquirirem a aplicação que se pretende desenvolver pois esta permitirá uma gestão colaborativa, um maior controlo sobre os stocks e consequentemente uma diminuição na utilização de recursos quer monetários quer materiais.

Com esta solução pretende-se assim que a interação entre o utilizador e aplicação seja fácil e intuitiva. Estes aspetos farão toda a diferença uma vez que as opções analisadas não conseguem acompanhar os objetivos pretendidos para esta aplicação.

Procura-se assim desenvolver uma aplicação móvel e uma *RESTful API*. A aplicação móvel permitirá efetuar uma melhor gestão de produtos, inventário e movimentos de entrada e saída de produtos para eventos de uma forma colaborativa.

3. ANÁLISE E ESPECIFICAÇÃO DE REQUISITOS

3.1 Introdução

O presente capítulo pretende analisar e dar a conhecer o trabalho desenvolvido a partir do estudo preliminar ao desenvolvimento da aplicação móvel e *RESTful* API. O estudo dos casos de uso, análise de diagramas, modelo de classes da base de dados e protótipos da aplicação móvel ajudarão ao correto desenvolvimento deste projeto.

3.2 Análise e especificação de requisitos

Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e o comportamento do sistema, ou seja, estes ao serem definidos têm de corresponder ao que é pretendido que o sistema faça e como este se deve comportar. Sendo assim serão agora apresentados os mesmos: sistema de registo e login, gestão de produtos, gestão de eventos, movimentos de entrada e saída de produtos nos respetivos eventos, gestão de utilizadores e respetivos cargos e visualização de relatórios finais.

Requisitos não funcionais

Os requisitos não funcionais dizem respeito às características e padrões de qualidade que o sistema deve oferecer, ou seja, são requisitos que não dizem respeito diretamente às funcionalidades do sistema, mas que por sua vez influenciam o comportamento deste. No contexto deste projeto serão apresentados os seguintes: performance, eficiência, desempenho (tempo de resposta e processamento), usabilidade (fácil aprendizagem), interface intuitiva e tolerância a falhas.

3.3 Fluxograma da aplicação móvel

Os fluxogramas são uma mais-valia em qualquer tipo de desenvolvimento de sistemas/programas de computadores, a criação deste, por vezes, pode tornar-se exaustiva, no entanto, tem os seus benefícios, tais como: facilitar a organização de um programa, ajudar em decisões críticas, deteção de possíveis erros inicias e identificar como os utilizadores poderão navegar na aplicação, dito isto, apresenta-se agora o fluxograma desenvolvido para a aplicação móvel:

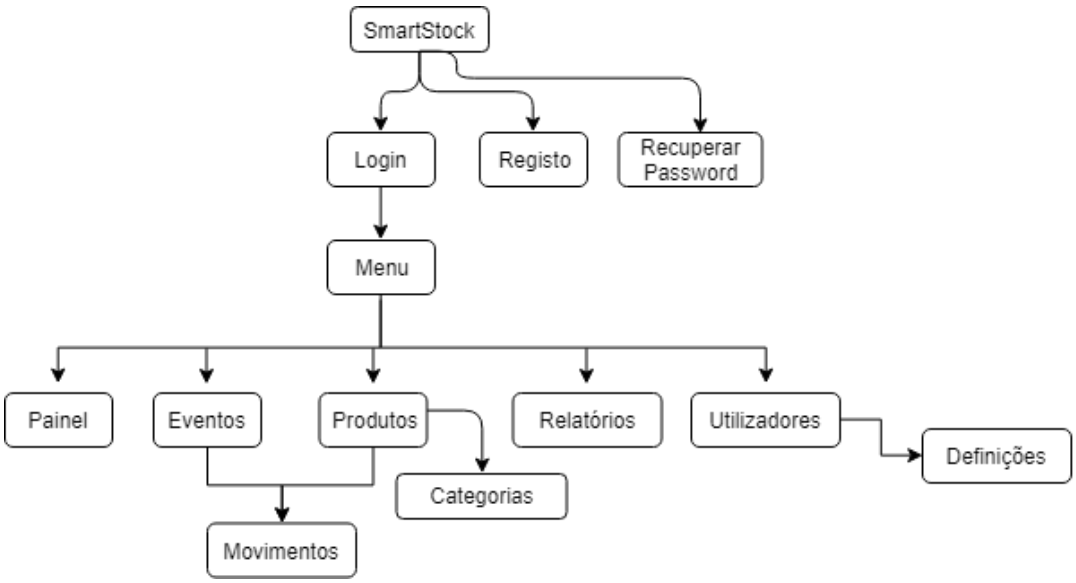

Figura 9 - Fluxograma da aplicação móvel

É possível identificar neste fluxograma, Figura 9, os principais componentes da aplicação móvel e as comunicações realizadas entre eles.

3.4 Casos de uso

Nas seguintes imagens estão especificados os casos de uso da aplicação móvel, estes representarão as unidades funcionais da interação do utilizador e administrador com o sistema, e permitem identificar, clarificar e organizar os requisitos deste.

Na Figura 10 podemos verificar a interação de um utilizador com a aplicação, onde este poderá visualizar, adicionar, editar ou remover e dar a entrada ou baixa de stock de produtos.

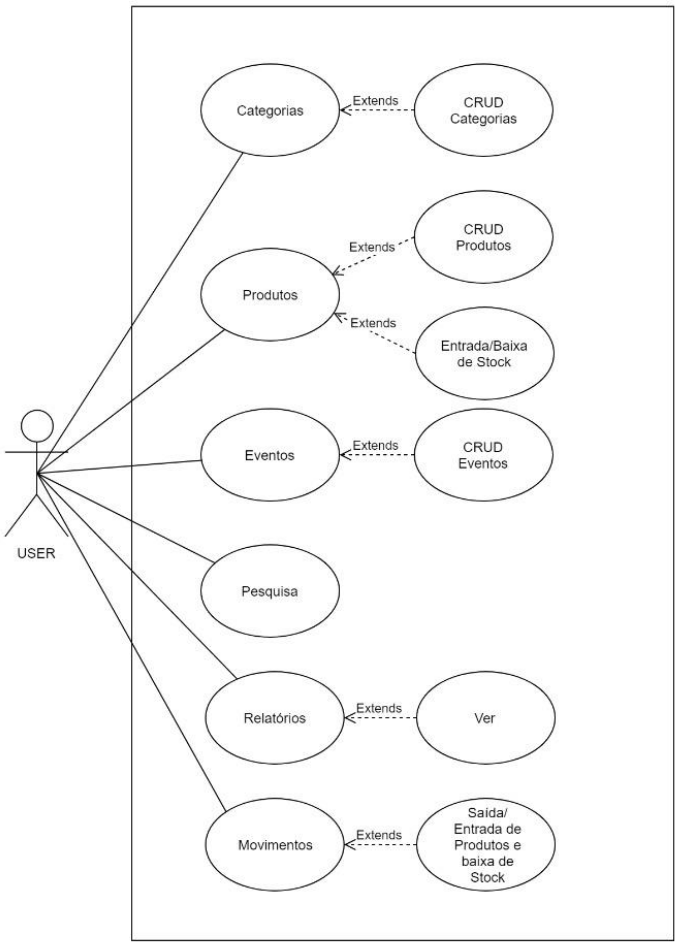

Figura 10 - Caso de Uso da aplicação móvel: Utilizador

Poderá também visualizar, adicionar, editar ou remover categorias, realizar movimentos de entrada ou saída de produtos e baixas de stock num determinado evento. O utilizador pode pesquisar produtos existentes, criar, editar, visualizar e apagar eventos. Por fim, este poderá visualizar os relatórios correspondentes a cada evento.

Na Figura 11 está representada a interação de um administrador com a aplicação:

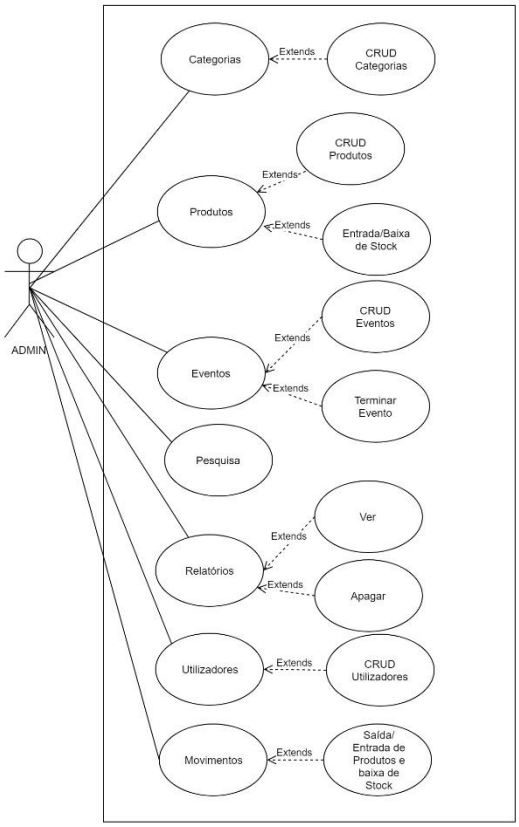

Figura 11 - Caso de Uso da aplicação móvel: Administrador

O administrador pode efetuar as mesmas operações que um utilizador comum, mas com o privilégio de poder terminar eventos, ver ou apagar relatórios e fazer a gestão de utilizadores.

3.5 Modelo de Classes

Antes de dar início ao desenvolvimento em concreto da aplicação, foi desenvolvida uma base de dados recorrendo a *MySQL*.

O *MySQL* é um sistema de gestão de bases de dados relacionais, suporta *SQL*, é *open source* e é um dos *SGBDs* para utilização profissional mais utilizado (conta com mais de 5 milhões de instalações ativas) e o mais conhecido a nível mundial. O *MySQL* foi desenvolvido e é disponibilizado pela empresa *MySQL AB Limited Company*, que atualmente vende um conjunto de serviços e produtos

relacionados com a tecnologia *MySQL* (centroatl, 2016). Optou-se pela escolha de *MySQL* pois permite a integração com todos os *frameworks* descritos neste documento.

Será agora apresentado o modelo de classes e descritas as tabelas com maior relevância para este projeto. Por fim, salienta-se a utilização do soft delete e a sua importância.

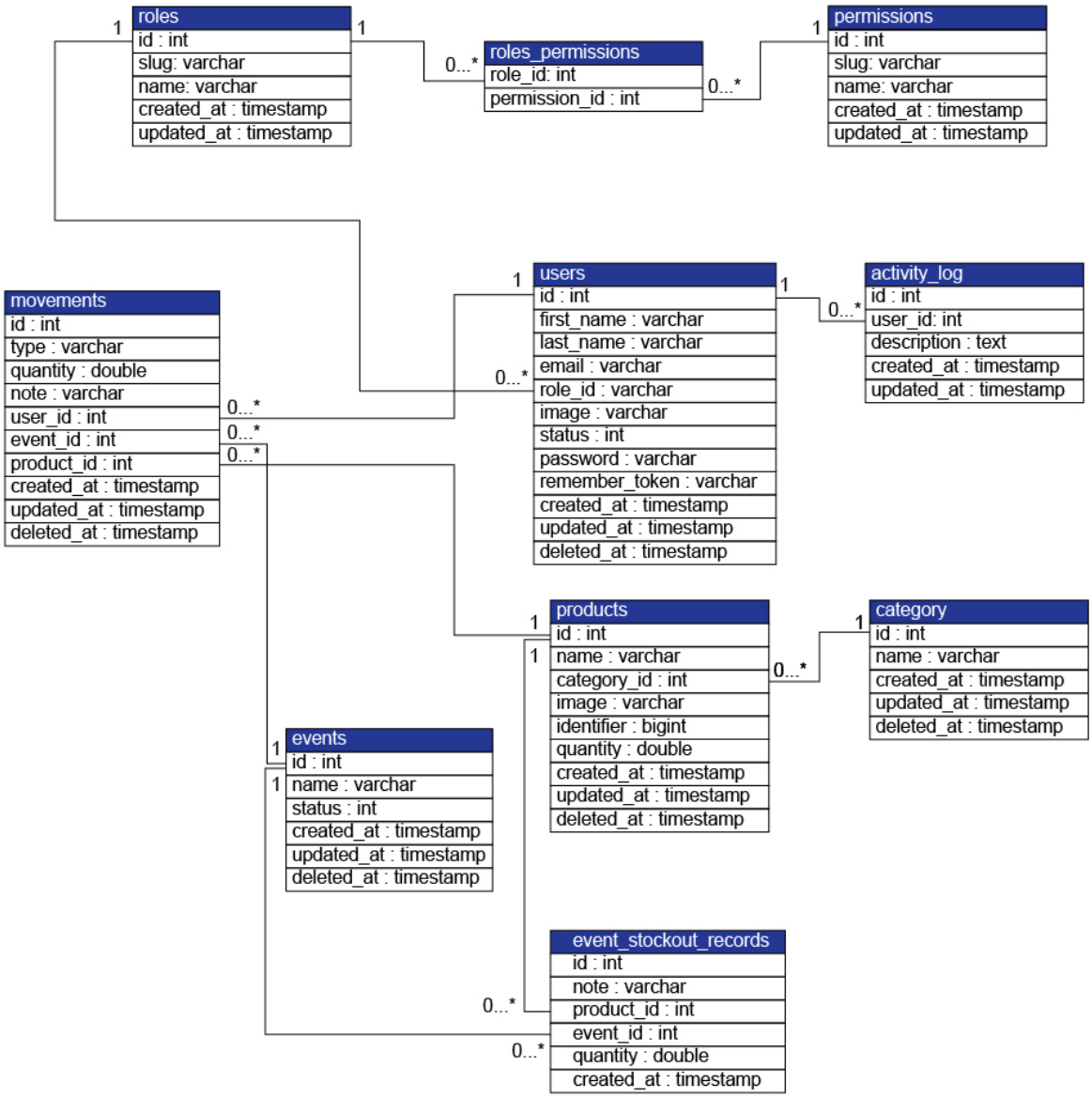

Figura 12 – Modelo de Classes

Descrição dos dados

Users: Tabela responsável por armazenar todos os dados dos utilizadores. Estes dados são imprescindíveis para efetuar a autenticação de um utilizador na aplicação.

Products: Tem como propósito guardar todos os dados relativos a produtos da aplicação.

Events: Esta tabela tem como função garantir que todos os dados necessários aos eventos sejam guardados com sucesso.

Movements: Regista todos os movimentos feitos na aplicação, ou seja, sempre que é dada a entrada ou saída de um produto num determinado evento é esta a tabela responsável por armazenar esses dados.

Events_stockout_records: Regista todos as baixas de stock efetuadas num evento ou diretamente num produto.

Activity_log: Regista a atividade dos utilizadores na aplicação.

Coluna deleted_at e a relação com Soft Delete

Soft Delete permite apagar registos sem os excluir da base de dados, desde que esses dados existam numa tabela onde esta contenha a coluna "*deleted_at*". Assim sendo, o *Soft Delete* insere nesta coluna a data em que o registo foi eliminado pelo utilizador, garantindo assim que estes dados continuem a existir.

Posteriormente, quando estes dados forem novamente consultados este registo não é visto/acessível pelo utilizador, mas continuam a existir na base de dados para futura utilização.

Este método é usado em algumas tabelas deste projeto, como por exemplo, a tabela "*products*" que previne que um produto seja apagado completamente caso este esteja a ser utilizado nos relatórios.

3.6 Protótipo da aplicação móvel

O desenvolvimento de protótipos é um passo fundamental na conceção de qualquer aplicação, pois estes ajudam a identificar e a definir os aspetos mais específicos. Possibilitam a visualização de como poderá ser o produto final, garantem que os requisitos do sistema atendem às necessidades e reduzem futuros erros e riscos do projeto. Podemos agora verificar alguns desses protótipos desenvolvidos nas figuras apresentadas abaixo.

Figura 13 – Protótipo do ecrã de Login e Registo

Na Figura 13, estão representados os primeiros ecrãs da aplicação onde o utilizador poderá efetuar o login utilizando o email e a password (no lado esquerdo).

Do lado direito da figura está representado o registo, neste o utilizador deverá fornecer o primeiro e último nome, email e a escolha da password.

Na Figura 14 pode-se ver representado do lado esquerdo da figura a listagem e pesquisa dos produtos existentes na aplicação.

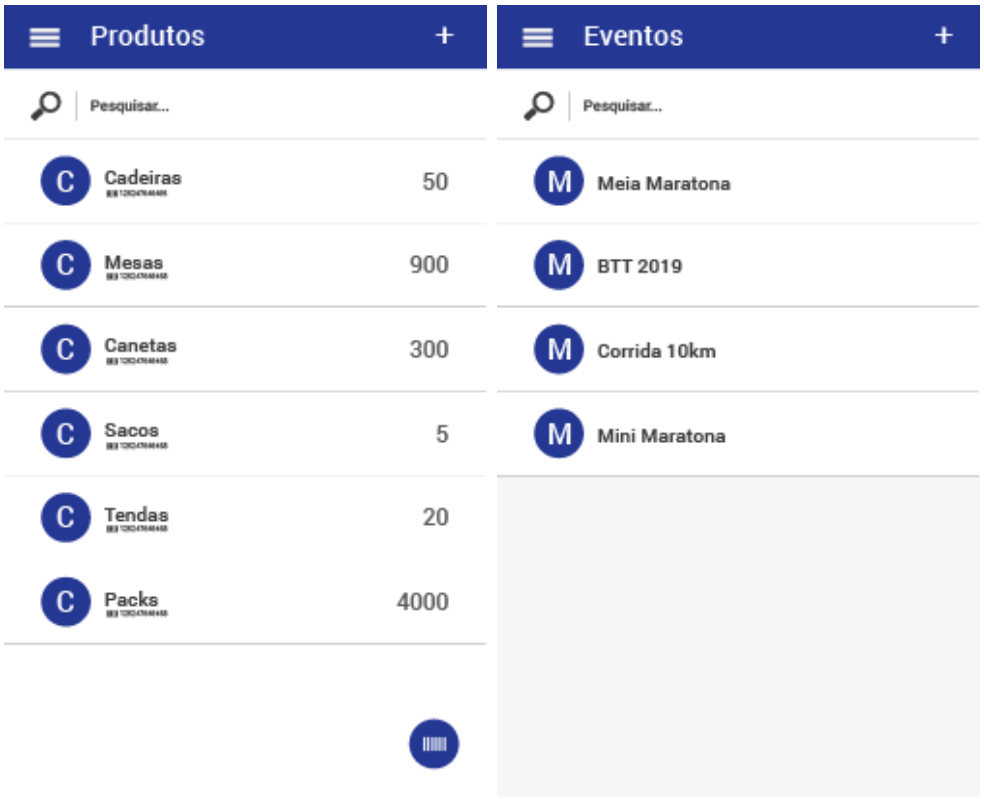

Figura 14 - Protótipo do ecrã de Produtos e Eventos

Ainda na Figura 14, no lado direito está representado o ecrã dos eventos onde consta a listagem dos mesmos e a opção de pesquisa por nome.

Na Figura 15, abaixo apresentada, o ecrã da esquerda tem como propósito dar a possibilidade de saída ou entrada de um produto no evento selecionado.

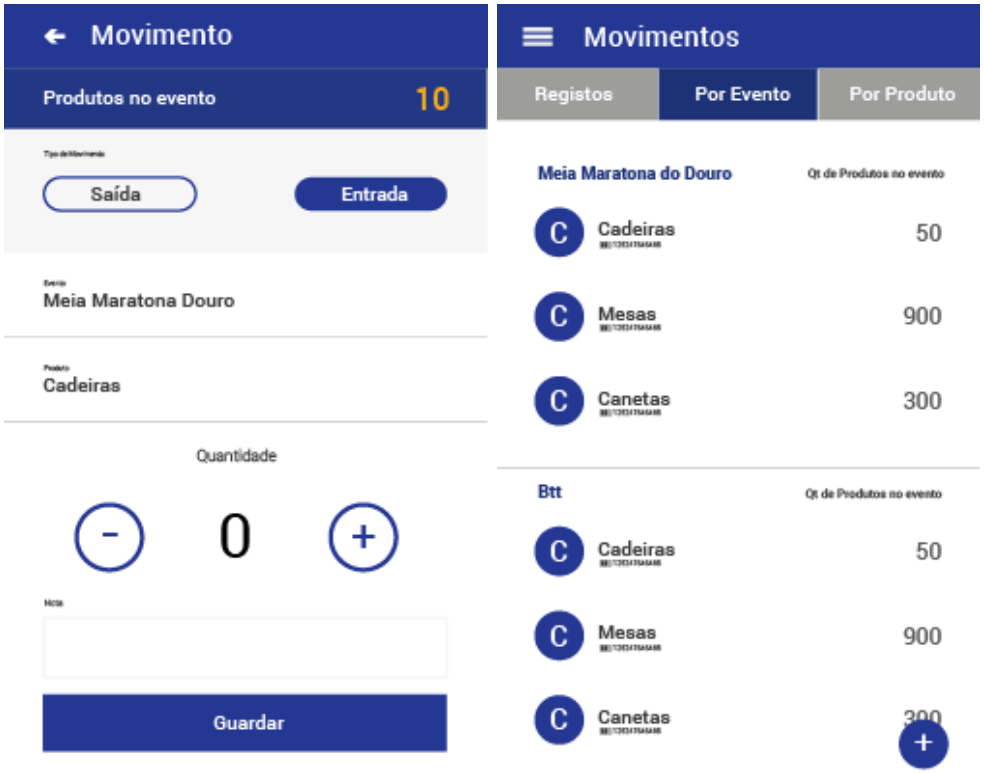

Figura 15 - Protótipo Aplicação Móvel

No lado direito da Figura 15 é possível visualizar a quantidade de produtos que estão a ser utilizados num determinado evento.

3.7 Arquitetura do Sistema

Visão Geral

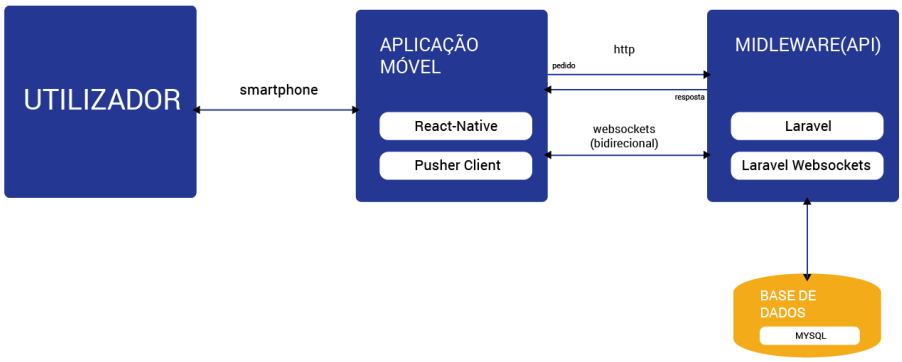

Figura 16 - Arquitetura (Visão Geral)

Na Figura 16 é apresentada a arquitetura do sistema. Destaca-se a comunicação entre a aplicação móvel e o *middleware*, que representam uma parte essencial do sistema. A aplicação móvel, comunica com o *middleware* de duas formas, por *http* que permite uma comunicação pedido-resposta e por *WebSockets* de forma bidirecional e em *real-time*.

Comunicação WebSockets

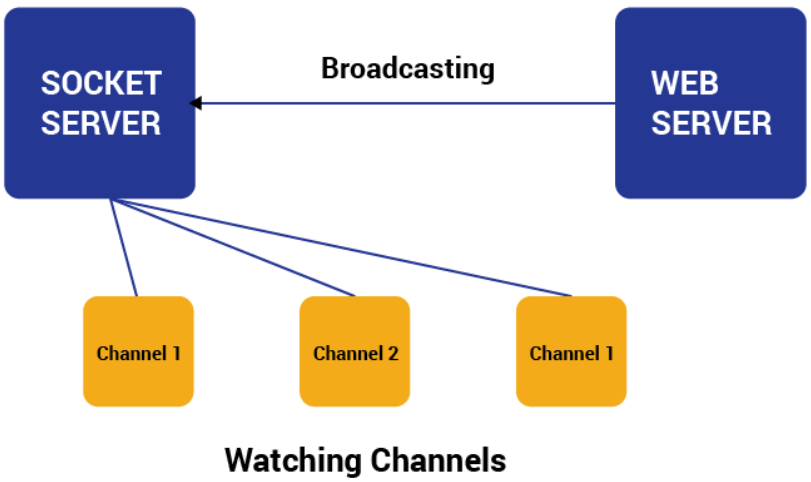

Figura 17 – Arquitetura (WebSockets)

Os componentes principais da comunicação por *WebSockets* podem ser vistos na Figura 17. Em primeiro lugar temos o servidor web, que envia informações sobre as alterações para o servidor *socket*, este manipula a transmissão e distribuição das informações através do protocolo *WebSockets*.

3.8 Conclusão

Em conclusão verifica-se a importância do planeamento para o desenvolvimento com sucesso desta aplicação. Neste capítulo foi possível analisar as especificações de requisitos, casos de uso, modelo de classes, protótipos da aplicação e arquitetura do sistema. Esta análise e planeamento permitirá que o desenvolvimento da aplicação seja mais fluído e bem sucedido. No próximo capítulo será abordado todo o desenvolvimento da aplicação móvel e *RESTful API*.

4. DESENVOLVIMENTO

No presente capítulo dar-se-á a conhecer as tecnologias escolhidas para o desenvolvimento da *RESTful API* e aplicação móvel. Serão apresentados todos os componentes desenvolvidos para este projeto bem como algumas figuras do produto final. Assim sendo, começa-se por apresentar o desenvolvimento da *RESTful API*.

4.1 RESTful API

4.1.1 Tecnologia escolhida para o desenvolvimento da RESTful API

Após o estudo feito das potenciais tecnologias para o desenvolvimento da *RESTful API* no capítulo 2.3 do presente documento, optou-se pelo desenvolvimento recorrendo à *framework* Laravel (Laravel, 2018). Esta disponibiliza inúmeros recursos dos quais se destaca o mecanismo de query builder, que permite efetuar queries à base de dados de uma forma mais simples. As migrações consistem em recriar a base de dados de uma forma automática e por fim, o package Laravel WebSockets que dá a possibilidade de criar um servidor de WebSockets, rico em recursos, e que retira a complexidade da utilização do Ratchet.

Com esta decisão pretende-se aproveitar ao máximo todas as potencialidades da framework para desenvolver uma API segura, rápida e estável.

4.1.2 Endpoints da API

Neste subcapítulo são descritos os *endpoints* principais da API e os respetivos controladores, de forma a demonstrar como este são obtidos da base de dados.

Login

Endpoint: /api/auth/signup/login

Method: POST

Controller: App\Api\V1\Controllers\LoginController@login

A principal função deste *endpoint* é efetuar o login do utilizador. Para tal, no corpo da requisição, deve ser enviado o *email e a password*. Em caso de sucesso, a API irá retornar uma *token* que permitirá ao utilizar navegar na aplicação.

```
1.  return response()
2.              ->json([
3.                  'status' => 'ok',
4.                  'token' => $token,
5.                  'expires_in' => Auth::guard()->factory()->getTTL() * 60
6.              ]);
```

Registo

Endpoint: /api/auth/signup

Method: POST

Controller: App\Api\V1\Controllers\SignUpController@signUp

Neste *endpoint*, é possível efetuar um novo registo de utilizador, para tal é necessário o envio do campo *first_name, last_name, email e password*. O controlador irá criar um registo na base de dados e gerar uma *token* para este novo utilizador.

```
1.  $user = new User($request->all());
2.          if(!$user->save()) {
3.              throw new HttpException(500);
4.          }
5.
6.          if(!Config::get('boilerplate.sign_up.release_token')) {
7.              return response()->json([
8.                  'status' => 'ok'
9.              ], 201);
10.         }
11.
12.         $token = $JWTAuth->fromUser($user);
13.         return response()->json([
14.             'status' => 'ok',
15.             'token' => $token
16.         ], 201);
```

Eventos

Endpoints: /api/events

Methods: POST,GET,DELETE,PUT

Controller: App\Api\V1\Controllers\EventController@

Com o *endpoint /events*, podem ser feitas todas as requisições CRUD, *(create, read, update e delete).* Como exemplo, é possível obter todos os eventos através de um pedido *GET* a este *endpoint*, ou se necessário, um evento em específico através do mesmo *endpoint*, mas com um parâmetro extra, o id /events/5.

```
1.  public function showAllEvents() {
2.          return response()->json(Event::orderBy('id', 'desc')->get());
3.      }
4.
5.  ublic function showOneEvent($id) {
6.          return response()->json(Event::find($id));
7.      }
```

Produtos

Endpoints: /api/products

Methods: POST,GET,DELETE,PUT

Controller: App\Api\V1\Controllers\ProductController@

Tal como no *endpoint /events*, também neste é possível efetuar todas as requisições CRUD. Neste, destaca-se a particularidade de ao ser criado um novo produto, e caso este não receba o identificador, ser gerado um automaticamente pelo sistema. Outro ponto importante é a alteração de stock, onde só pode ocorrer caso o stock seja maior que os produtos que deram saída para o evento.

```
1.  if ($MinStock >= $request->quantity) {
2.
3.      $Product = Product::where('id', '=', $id)->update(array(
4.          'name' => $request->name,
5.          'quantity' => $ProductQuantity - $request->quantity,
6.      ));
7.
8.      $stock_out = DB::table('Products_stock')->insert([
```

```
            'quantity' => $request->quantity,
            'note' => $request->note,
            'type' => 'OUT',
        ]);

        return response()->json('Baixa de stock adicionada com sucesso', 200);

    }
```

```
function generateBarcodeNumber()
{
    $number = mt_rand(1000000000, mt_getrandmax());

    if (barcodeNumberExists($number)) {
        return generateBarcodeNumber();
    }
    return $number;
}

function barcodeNumberExists($number)
{
    return Item::where('identifier', '=', $number)->exists();
}
```

Movimentos

Endpoints: /api/movements

Methods: POST,GET,DELETE,PUT

Controller: App\Api\V1\Controllers\MovementController@

Este é o *endpoint* mais importante da aplicação, pois irá tratar todos os movimentos de saída e entrada de produtos nos eventos.

Destaca-se no controlador deste endpoint a função *showAllEventsWithMovementsByProduct($id),* que retorna todos os eventos com movimentos de um determinado produto e *getAvailableStockByProduct*($id), que retorna o stock disponível para saída com base nos movimentos que foram efetuados para um produto específico.

```php
public function showAllEventsWithMovementsByProduct($id=null)
    {

        $eventsWithMissignStock = DB::select('
        SELECT m.id, COALESCE(m.missing_stock,0) as missing_stock ,e.id as event_id,m.Product_id,e.name
        FROM events e
        LEFT JOIN movements m
        ON e.id = m.event_id and m.Product_id= $id
                AND m.id IN
                (
                    SELECT MAX(m.id)
                    FROM movements m
                    GROUP BY m.Product_id, m.event_id
                )');


        return response()->json(array('data' => $eventsWithMissignStock));


    }
```

```php
public function getAvailableStockByProduct($id)
    {

        $missing = DB::select('
        SELECT id, COALESCE(SUM(missing_stock),0) as missing_stock ,Product_id
        FROM movements
        WHERE Product_id=$id
        AND id IN
        (
            SELECT MAX(id)
            FROM movements
            GROUP BY Product_id, event_id
        )');

        $StockProduct = DB::table('Products')
            ->where('id', $id)
            ->sum('quantity');

        $availableStock= $StockProduct - $missing[0]->missing_stock;



        return response()->json(array('available' => $availableStock, 'stock' => $StockProduct));

    }
```

4.3 Aplicação Móvel

4.3.1 Tecnologias escolhidas para o desenvolvimento móvel

Para o desenvolvimento da aplicação móvel foi elegida a *framework* React Native pela possibilidade de reutilizar código entre plataformas e escrever uma aplicação robusta e segura. De forma a manter o controlo do estado da aplicação e ter a possibilidade de fazer pedidos assíncronos à API foram utilizados o Redux e o Redux Thunk.

4.3.2 Componentes

Serão agora apresentadas as componentes desenvolvidas para a aplicação móvel, estas são acompanhadas por partes relevantes da programação.

Registo, login e recuperação de password do utilizador

Para o desenvolvimento desta aplicação móvel, foi criado inicialmente o processo de registo do utilizador como ilustra a Figura 19. Para o poder efetuar deve ser colocado o primeiro e último nome, email e escolhida uma password pessoal de acesso à aplicação. Ao clicar no botão registar, será automaticamente enviado um email aos administradores, a informar o registo. Um dos administradores através do seu acesso pode validar este utilizador. Após a validação do administrador, o utilizador recebe um email de confirmação para aceder à aplicação SmartStock.

Após a receção do email, o utilizador pode efetuar o login (Figura 18) através da validação das suas credenciais, e assim utilizar todas a funcionalidades da aplicação.

Existe a possibilidade de recuperação da password no caso de necessidade por parte do utilizador, para tal, deve clicar no botão recuperar password, a aplicação vai solicitar o email e envia uma nova password de acesso.

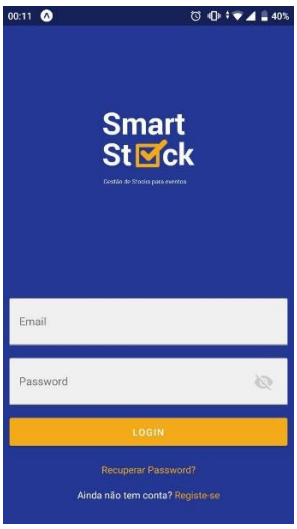

Figura 18 - Login

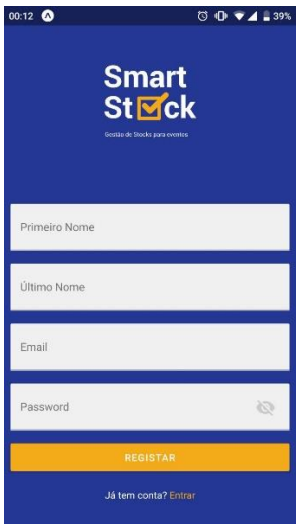

Figura 19 - Registo

No que diz respeito ao desenvolvimento, destaca-se a resposta ao pedido de login. Numa primeira fase, é disparada uma *action* ao carregar no botão Login. De seguida é efetuado o pedido à API com os dados introduzidos pelo utilizador. Caso a resposta seja bem-sucedida, esta irá retornar uma *token* que é armazenada pela aplicação no sistema de *AsyncStorage*. Esta *token*, servirá para o utilizador conseguir navegar pelos ecrãs da aplicação. De seguida, são dados exemplos da implementação.

```
1.  const mapDispatchToProps = (dispatch) => {
2.    return{
3.      login: (creds) => dispatch(login(creds))
4.    }
5.  }
```

```
1.  export const login = (credentials) => {
2.     return (dispatch, getState) => {
3.
4.           fetch('auth/login', {
5.             method: 'POST',
6.             body: JSON.stringify(credentials),
7.             headers:{
8.               'Content-Type': 'application/json'
9.             }
10.          })
11.          .then(res => res.json())
12.          .then(response => {
13.            console.log(response)
14.            dispatch({type:'LOGIN_SUCCESS'})
15.            if(response.token) {
```

```
16.             dispatch(storeToken(response.token))
17.                 NavigationService.navigate('Products', {});
18.              }
19.             })
20.            .catch(error =>{
21.              dispatch({type:'LOGIN_ERROR',err})
22.              console.error(error)});
23.
24.     }
25.  }
```

Menu e painel principal

O painel principal da Figura 20, apresenta as informações gerais e os últimos movimentos de saída e entrada de produtos no evento. É possível encontrar três separadores “produtos”, “eventos” e “em uso” que permitem ao utilizador saber de uma forma global o que está a ser utilizado e disponível no momento.

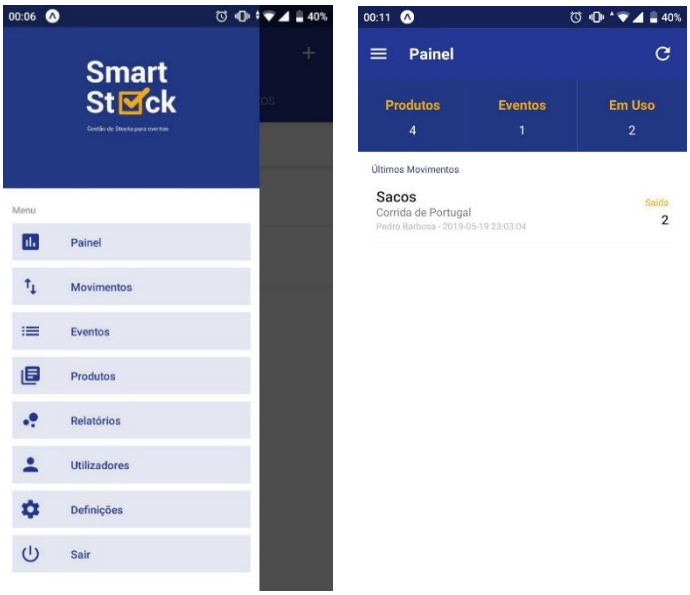

Figura 20 - Painel

Pode ainda aceder ao menu onde estão disponíveis todas as opções para navegar entre ecrãs, este menu está disponível em todos os ecrãs principais da aplicação.

Produtos

Na Figura 21 é apresentada uma *listview* com todos os produtos, o stock e *barcode/QR Code*. A *listview* tem um campo de pesquisa que serve de filtro, podendo ser efetuada através do *barcode/QR Code* ou nome, de forma a fornecer ao utilizador uma resposta mais célere. Neste mesmo ecrã pode ser acionado o scan do *barcode*/*QR Code* através de um botão no canto inferior direito do ecrã. Ao acionar o scan, a camara inicia a captura do barcode/QR Code e automaticamente apresenta o respetivo produto.

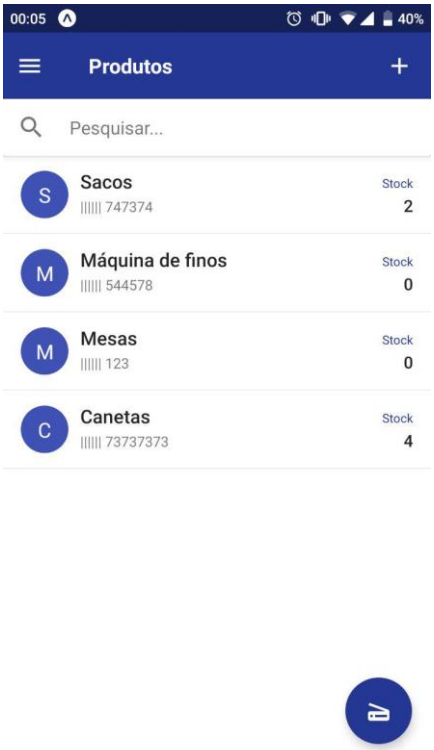

Figura 21 - Produtos

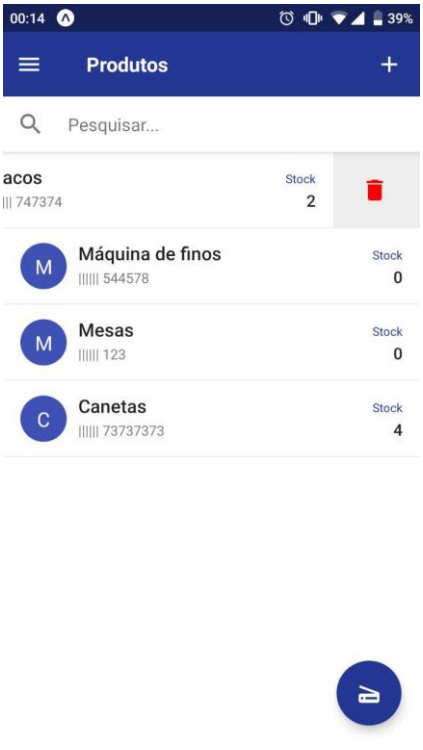

Figura 22 - Opções para Apagar/Editar produto

Ao deslizar o item na *listview* (Figura 22) é possível apagar ou editar o mesmo, se deslizar para o lado esquerdo é possível selecionar a opção apagar, e para o lado direito editar. No caso de se pretender ver os detalhes, deve clicar sobre o mesmo. Este comportamento será o mesmo em todas as *listviews* da aplicação caso as opções estejam disponíveis.

No ecrã com os detalhes do produto (Figura 22) é possível consultar as existências em stock e o identificador do produto. Neste, encontramos também no canto superior direito, um botão que direciona para o ecrã de entradas / baixas de stock com o símbolo (+).

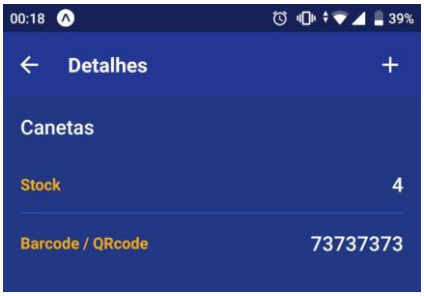

Figura 23 - Detalhes do Produto

Figura 24- Entrada e Baixa de Stock

Ao ser direcionado (Figura 24), é apresentado o stock atual, duas opções para selecionar o tipo de movimento que se pretende fazer, que pode ser de entrada ou baixa de stock, e um campo para a quantidade. Ao guardar o movimento, o ecrã é recarregado e apresenta o stock atualizado.

Regressando aos produtos (Figura 21), encontramos no canto superior direito um botão com o sinal (+) que direciona o utilizador para um novo ecrã (Figura 25). Neste, é possível adicionar um novo produto onde deve ser preenchido o nome, *barcode/QR code*, quantidade e opcionalmente pode fazer o upload de uma foto. O campo *barcode/QR code* tem a opção scan que lê o código e preenche automaticamente o campo, evitando que o utilizador tenha de o fazer manualmente. No caso de não preencher este campo, o sistema atribui automaticamente um

código, por isso é importante que, no caso do produto já estar etiquetado, reutilizar o mesmo.

No que diz respeito ao campo da quantidade, o utilizador tem duas opções, o mostrador numérico preenchido manualmente e/ou dois botões que permitem adicionar ou subtrair a quantidade de produtos. Este campo repetir-se-á nos restantes ecrãs.

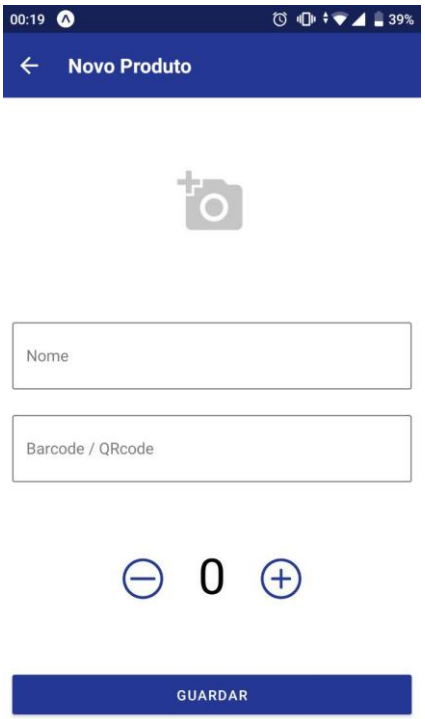

Figura 25 - Adicionar Produto

Figura 26 - Editar Produto

A edição do produto (Figura 26) é semelhante ao processo de adicionar, com a exceção do campo quantidade que deverá ser feito no ecrã de entradas e baixas de stocks.

Eventos

Ao selecionar o menu eventos, é apresentado um ecrã (Figura 27) com a lista de eventos disponíveis. Neste ecrã é possível efetuar a pesquisa de eventos através do nome.

Ao adicionar um novo evento deve-se preencher o campo nome, data do evento e carregar no botão guardar.

Figura 27 – Eventos

Caso seja necessário editar o evento, deve deslizar para a direita o item correspondente ao evento no ecrã eventos, clicar, fazer a edição necessária e por fim guardar.

Para visualizar os detalhes deve clicar sobre o evento pretendido no ecrã eventos. Além dos detalhes, aqui encontra-se uma opção para terminar, que será contextualizada na apresentação dos ecrãs relativos aos relatórios.

Movimentos

O ecrã dos movimentos está dividido em duas *tabs* (Figura 28). A primeira tab “em uso”, refere-se aos produtos utilizados num determinado evento. Nesta tab é possível filtrar os registos por nome de produto ou evento.

A segunda *tab* “registos” apresenta todos os movimentos de entrada e saída.

Figura 28 - Movimentos - Registos

Para efetuar um movimento (Figura 29) de entrada ou saída de produtos no evento, existe um botão no canto inferior direito do ecrã (Figura 28) com o sinal (+), que direciona para o ecrã de eventos. O utilizador deve selecionar um dos eventos e após a seleção, é direcionado para o ecrã produtos. É feita a seleção do produto, e com estes dois pré-requisitos, pode então dar entrada e saída do produto já no ecrã movimento. Aqui, existem dois botões para selecionar o tipo de movimento, que pode ser de entrada ou saída. Ainda neste ecrã, e caso seja necessário é possível alterar o evento ou produto, sem regressar ao ecrã movimentos, basta clicar sobre uma das opções pretendidas. O utilizador seleciona a quantidade e tem opção de colocar uma nota relativa a este movimento. Depois de submetido o

movimento, o ecrã é recarregado e é atualizado, no topo, o número de produtos no evento.

Figura 29 - Movimento

Para efetuar este movimento, é disparada uma *action* sempre que o utilizador carrega no botão guardar. Esta envia um pedido à API com o movimento pretendido. De seguida, e em caso de sucesso, será devolvida uma resposta em *json* para a seleção efetuada com o total de produtos no evento, dando assim a possibilidade de o utilizador a confirmar.

De seguida é apresentado o código relevante para descrever este processo.

```
1.  const mapDispatchToprops = dispatch => {
2.      return {
3.          setFalse: () => dispatch(setFalse()),
4.          addMovement: movement => dispatch(addMovement(movement)),
5.          dialogAction: bool => dispatch(dialogAction(bool)),
6.      };
7.  };
```

```
export const addMovement = (movement) => {
    return(dispatch, getState) => {

        console.reportErrorsAsExceptions = false;

        token = getState().auth.token;

        if (token) {
            fetch('http://localhost:8000/api/movements?token=' + token, {
                method: 'POST',
                body: JSON.stringify(movement),
                headers: {
                    'Content-Type': 'application/json'
                }
            })
            .then(res => {
                if(res.ok) {
                    return res.json()
                }else{
                    throw res.json();
                }
            })
            .then(response => {
                //NavigationService.navigate('Goods', {});
                dispatch({type:"ADD_MOVEMENT_SUCCESS"})
                console.log(response)
            })
            .catch(error => {
                dispatch({type:"ADD_MOVEMENT_ERROR", error:error})
            });

        }
        else {
            NavigationService.navigate('Login', {});
        }


    }
}
```

Relatórios

Para que seja possível apresentar os relatórios (Figura 30) o evento deve estar concluído, para tal, o utilizador deve clicar no menu eventos, selecionar o evento pretendido e escolher a opção “concluir evento”. Concluído o evento, todos os produtos que estão em falta são dados como baixa de stock. Após terminado, o evento não pode ser reaberto.

O ecrã relatórios é constituído por duas *tabs*: baixas de stock e registos. Na primeira *tab* é possível visualizar as baixas de produtos por evento, na segunda *tab* registos pode-se visualizar todos os movimentos de entrada e saída efetuados nos eventos.

Figura 30 - Relatórios - Visão Global

Os relatórios nesta fase são importantes porque permitem dar a conhecer à empresa as necessidades presentes e futuras que têm de material, ter conhecimento por evento do desgaste que o material sofre, necessidade de futuras aquisições e com isto ter sempre o conhecimento real de tudo o que entra e sai para cada evento e evitar gastos desnecessários em duplicação de produtos.

Utilizadores

Mediante as permissões que o administrador concede aos utilizadores, estes poderão ter acesso a algumas funcioanlidades, como é o caso dos utilizadores por

aprovar/bloquear. Quando o administrador acede à aplicação consegue consultar os utilizadores ativos e os por aprovar/bloqueados (Figura 31), só após dada a autorização através da aprovação ou desbloqueio, o utilizador poderá começar a navegar.

Figura 31 - Utilizadores

O administrador pode também adicionar e editar utilizadores, e atribuir o cargo pretendido.

Definições

Para fazer alterações nos dados pessoais e definições da aplicação, tais como, ativar ou desativar notificações, envio de emails, alteração de password, entre outros, está predefinido no menu a opção definições que permite ao utilizador um maior controlo e autogestão na aplicação.

4.4 Implementação e comunicação com o servidor de WebSockets

Como referido anteriormente neste documento, foi utilizado o package *Laravel Websockets* como meio de transmissão de dados em tempo real entre a aplicação móvel e a API. De seguida são apresentadas algumas terminologias de *WebSockets* e descrito um exemplo da implementação.

Channel/Canal

Canal ou *channel* é um elemento organizacional de *Websockets*. São utilizados canais para distribuir informações para os utilizadores que efetuaram uma subscrição do mesmo.

Event/Evento

Um evento serve para identificar uma ação pretendida. No exemplo que será demostrado de seguida é possível verificar que o evento é "*DeleteProduct*", que será enviado para os utilizadores que tenham subscrito o canal "*Product*".

Broadcasting

De forma a ajudar a criar interfaces de utilizador em tempo real, *Laravel* facilita a "transmissão" de eventos por uma conexão *Websockets*. Fazer *Broadcasting* de eventos com o *Laravel* permite compartilhar os mesmos nomes de eventos entre o código do lado do servidor e a aplicação *javascript* do lado do cliente.

Implementação

Depois de efetuadas as configurações iniciais do package *Laravel WebSockets*, é necessário criar um evento no *Laravel*, recorrendo ao comando *php artisan make:event DeleteProduct.*

De seguida é necessário implementar nesta classe um *Contract* com o nome de *ShouldBroadcast*, este *Contract* é uma interface de serviços core do *Laravel* que permite transmitir dados de uma forma imediata para um servidor *Websockets*.

```php
class DeleteProduct implements ShouldBroadcast
{
    use Dispatchable, InteractsWithSockets, SerializesModels;

    public $product;

    /**
     * Create a new event instance.
     *
     * @return void
     */
    public function __construct(Product $product)
    {
        $this->product = $product;
    }

    /**
     * Get the channels the event should broadcast on.
     *
     * @return \Illuminate\Broadcasting\Channel|array
     */
    public function broadcastOn()
    {
        return new Channel('Product');
    }

    public function broadcastAs()
    {
        return 'product.deleted';
    }
}
```

Como é possível verificar no código descrito em cima, encontramos dois métodos: *broadcastOn*(), responsável por criar o canal, utilizado para fazer o "*subscribe*" do utilizador e transmitir os dados para o servidor *Websockets* e o método *broadcastAs*, que identifica o evento no momento da transmissão.

Por fim, é utilizado este novo evento no controlador do produto.

```php
public function delete($id)
    {
        ...
            $product->delete();
            event(new DeleteProduct($product));

            return response()->json('Eliminado com sucesso.', 200);
        }
```

Do lado da aplicação e sempre que um utilizador eliminar um produto é disparado um evento para todos os utilizadores que tenham feito "*subscribe*" do canal "*Product*".

4.5 Conclusão

Para uma correta elaboração dos componentes da aplicação móvel e *RESTful API*, verificou-se a extrema utilidade e importância do planeamento para conseguir um desenvolvimento eficaz. Com a elaboração dos protótipos, definição dos requisitos e com a correta escolha de *frameworks*/bibliotecas, foi possível atingir os objetivos principais desta aplicação e deste projeto, e assim elaborar uma aplicação de gestão de stocks para eventos que simplifica o processo de gestão e permita às empresas poupar recursos.

5. AVALIAÇÃO DOS RESULTADOS OBTIDOS

Neste capítulo serão descritos os processos e métodos de avaliação levados acabo após a conclusão do sistema desenvolvido. Estes testes foram elaborados com o objetivo de avaliar os requisitos definidos ao longo do projeto, são de extrema importância pois avaliam o sistema e ajudarão a definir caso necessário melhorias e trabalho futuro que possa vir a ser realizado.

5.1 Cenários

De forma a criar um caso real onde esta aplicação fosse útil e contextualizando assim os inquiridos, foram desenvolvidos os seguintes cenários de uso:

Cenário 1

Durante este fim de semana decorre o evento “Maratona de Viana”, para tal é necessário fazer toda a gestão deste evento. O responsável pela logística já fez o inventário dos seguintes produtos: 50 Cadeiras; 1000 Águas; 20 Tendas; 100 Grades; 120 Canetas; 5 toldos; Neste momento é necessário adicionar ao inventário os produtos que ainda estão por registar no armazém, para tal, um colaborador da equipa de logística, efetua a verificação e adiciona ao inventário 200 t-shirts, 5 PCs e efetua o registo do respetivo barcode.

Com o inventário atualizado, o colaborador já pode efetuar a saída de produtos que irá necessitar para o evento “Maratona de Viana”. Após identificar as necessidades, este efetua a saída dos seguintes produtos: 3 PCs, 100 t-shirt, 20 cadeiras e 100 Águas. De forma a acelerar o processo o colaborador utiliza o leitor de barcodes para rápida deteção do mesmo. Ao efetuar o carregamento dos produtos, este verifica que uma das cadeiras está partida e efetua a baixa deste produto identificando o motivo.

No dia que antecede o evento, verificou-se a necessidade de utilizar toldos para o sol, dada a inexistência de árvores no local de meta. Para tal o colaborador

verifica no inventário se existe o produto e se este tem disponibilidade. Após verificação, confirma a disponibilidade e dá saída de 5 toldos.

No decorrer do evento o colaborador verifica que restam poucas garrafas de água nos postos de abastecimento, sendo necessário fazer a reposição. O colaborador verifica a disponibilidade de stock e efetua a saída de 50 Águas para o evento.

No fim do evento, o colaborador efetua a entrada dos seguintes produtos: 2 PCs, 2 t-shirt, 20 cadeiras(1 das cadeiras precisa de substituir o encosto, é necessário deixar a indicação no momento de entrada) e 5 Águas; De seguida efetua a verificação dos produtos que ainda estão no evento e depara-se com as seguintes falhas: 145 Águas, que não deram entrada porque foram consumidas; 98 t-shirts; 5 toldos; e 1PC. Após efetuar a procura do pc, não foi possível encontrá-lo. Depois da revisão dos produtos em falta, o responsável da logística decide efetuar o fecho do evento e solicita ao seu colaborador que faça a reposição das águas utilizadas. Para tal, o colaborador verifica a quantidade de águas consumidas no evento e procede à entrada de stock.

Cenário 2

Durante este fim de semana decorre o evento “Maratona de Viana”, para tal é necessário fazer toda a gestão deste evento. O responsável pela logística já fez o inventário dos seguintes produtos: 50 Cadeiras; 1000 Águas; 20 Tendas; 100 Grades; 120 Canetas; 5 toldos; Neste momento é necessário adicionar ao inventário os produtos que ainda estão por registar no armazém, para tal, o colaborador da equipa de logística, efetua a verificação e adiciona ao inventário t-shirts, PCs e efetua o registo do respetivo barcode deixando a quantidade a zero.

Em simultâneo outro colaborador após efetuar a contagem dos produtos disponibilizados pelo colega, atualiza o stock para as seguintes quantidades: 200 t-shirts, 5 pcs.

Com o inventário atualizado, o colaborador já pode efetuar a saída de produtos que irá necessitar para o evento “Maratona de Viana”. Após identificar as necessidades, este efetua a saída dos seguintes produtos: 3 PCs, 100 t-shirt, 20 cadeiras e 100 Águas. De forma a acelerar o processo o colaborador utiliza o leitor

de barcodes para rápida deteção do mesmo. Ao efetuar o carregamento dos produtos, este verifica que uma das cadeiras está partida e efetua a baixa deste produto identificando o motivo.

No mesmo momento em que o utilizador está a dar saída do produto cadeiras, outro colaborador que está a efetuar o mesmo movimento, verifica em tempo real que a disponibilidade foi atualizada (de 50 para 30 cadeiras).

No dia que antecede o evento, verificou-se a necessidade de utilizar toldos para o sol, dada a inexistência de árvores no local de meta. Para tal o colaborador verifica no inventário se existe o produto e se este tem disponibilidade. Após verificação, confirma a disponibilidade e dá saída de 5 toldos.

No decorrer do evento o colaborador verifica que restam poucas garrafas de água nos postos de abastecimento, sendo necessário fazer a reposição. O colaborador verifica a disponibilidade de stock e efetua a saída de 50 Águas para o evento.

No fim do evento, o colaborador efetua a entrada dos seguintes produtos: 2 PCs, 2 t-shirt, 20 cadeiras(1 das cadeiras precisa de substituir o encosto, é necessário deixar a indicação no momento de entrada) e 5 Águas; De seguida efetua a verificação dos produtos que ainda estão no evento e depara-se com as seguintes falhas: 145 Águas, que não deram entrada porque foram consumidas; e 1PC. Após efetuar a procura do PC, não foi possível encontrá-lo. Depois da revisão dos produtos em falta, o responsável da logística decide efetuar o fecho do evento e solicita ao seu colaborador que faça a reposição das águas utilizadas. Para tal, o colaborador verifica a quantidade de águas consumidas no evento e procede à entrada de stock.

5.2 Metodologia de avaliação SUS

O SUS

System Usability Scale (SUS), criado em 1986 por John Brooke, *é um instrumento autoadministrado e amplamente utilizado para a avaliação da usabilidade de uma ampla gama de produtos e interfaces de utilizador. O valor*

principal do SUS é fornecer uma única pontuação de referência para a visão dos participantes sobre a usabilidade de um produto ou serviço (Martins, Rosa, Silva, & Rocha, 2015). Esta será a metodologia de avaliação de usabilidade à aplicação móvel desenvolvida.

Questões

O SUS utiliza 10 simples questões para avaliar a usabilidade do sistema, sendo elas:

1. Acho que gostaria de utilizar este produto com frequência.
2. Considerei o produto mais complexo do que necessário.
3. Achei o produto fácil de utilizar.
4. Acho que necessitaria de ajuda de um técnico para conseguir utilizar este produto.
5. Considerei que as várias funcionalidades deste produto estavam bem integradas.
6. Achei que este produto tinha muitas inconsistências.
7. Suponho que a maioria das pessoas aprenderia a utilizar rapidamente este produto.
8. Considerei o produto muito complicado de utilizar.
9. Senti-me muito confiante a utilizar este produto.
10. Tive que aprender muito antes de conseguir lidar com este produto.

A tradução para português das 10 questões acima foi obtida através do artigo "*European Portuguese validation of the System Usability Scale*" (Martins et al., 2015).

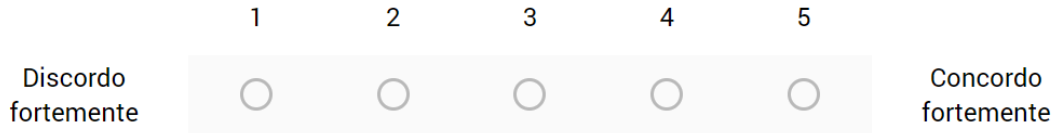

Figura 32 - Formato das respostas ao SUS

As respostas a este questionário são realizadas numa escala de 1 (discordo fortemente) a 5 (concordo fortemente), como se pode verificar na figura acima.

Pontuação dos Resultados

Após a realização do questionário as pontuações serão obtidas da seguinte forma:

- Para números ímpares: subtrair 1 valor à resposta do utilizador.
- Para números pares: subtrair 5 valores ao valor da resposta do utilizador.
- Isto permitirá escalar todos os valores de 0 a 4 (sendo 4 a resposta mais positiva)
- Somar as repostas convertidas para cada utilizador e multiplicar o total por 2.5. Isto converte o intervalo de valores possíveis de 0 a 100, em vez de 0 a 40.

Uma pontuação do SUS acima de 68 será considerada acima da média e qualquer resultado abaixo de 68 estará abaixo da média (Sauro, 2011).

Resultados

Após a análise e interpretação dos resultados, obteve-se um valor de 94,5 pontos na escala de SUS, em que se concluí que esta aplicação está acima da média, cumprindo assim com os requisitos do SUS. Foram inquiririas 10 pessoas das quais 40% encontra-se na faixa etária entre 24 e 30 anos, 40% entre 31 e 40 anos e 20% entre 41 e 54 anos de idade; 20% com o Ensino Secundário, 60% Licenciados e 20% com o Mestrado.

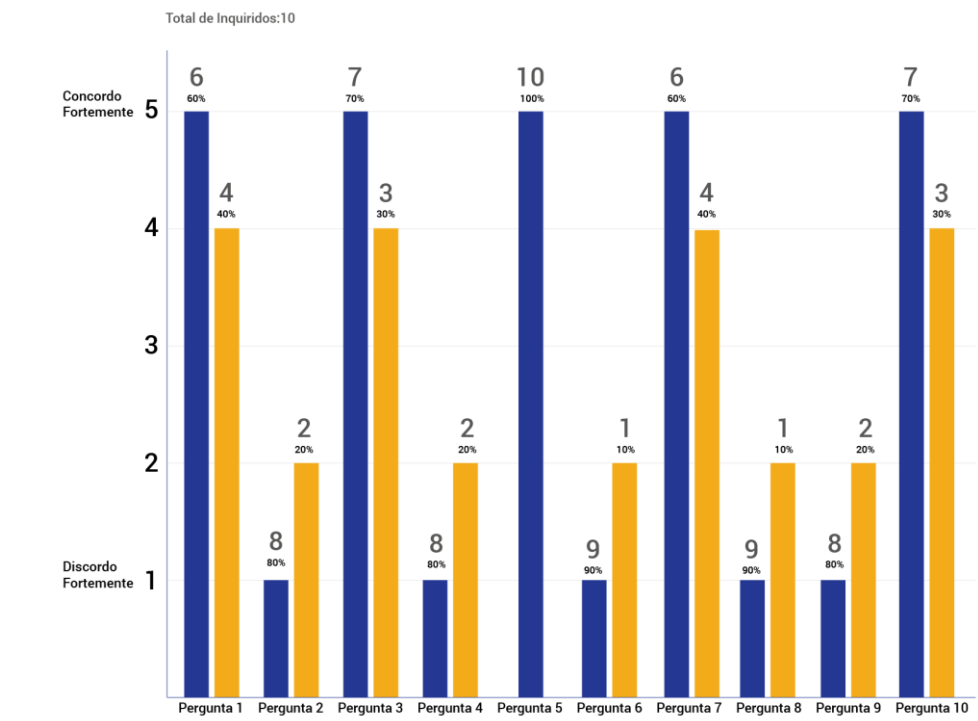

Figura 33 - Gráfico de Resultados SUS

Na Figura 33 estão representados os resultados obtidos às 10 questões do SUS, onde podemos verificar que no eixo do x estão representadas as 10 questões e no eixo do y está representada a escala do SUS que é compreendida entre 1(discordo fortemente) e 5(concordo fortemente).

Serão agora analisadas mais ao pormenor algumas destas questões colocadas aos inquiridos:

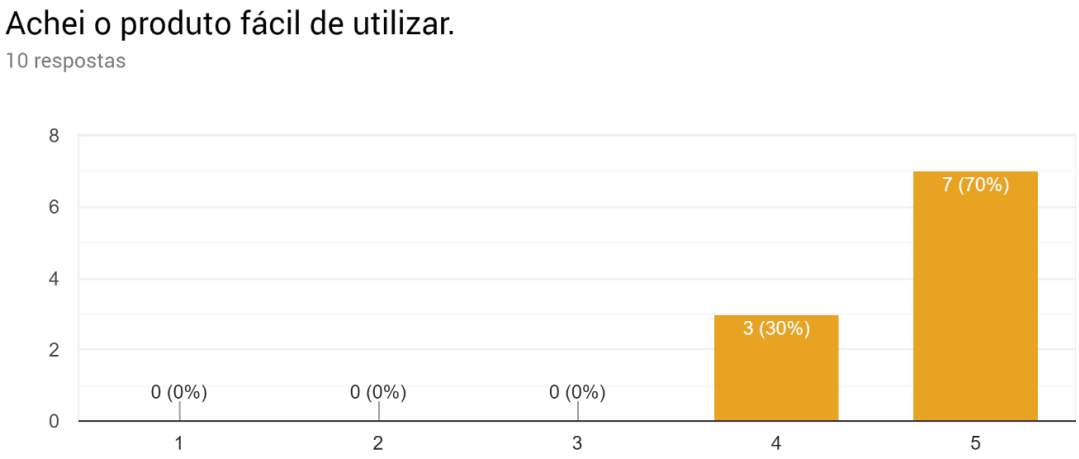

Figura 34 - Resultado pergunta 3

A figura acima apresentada representa os resultados obtidos à pergunta número 3 do questionário SUS, pelo que é possível analisar que os inquiridos acharam a aplicação acessível, correspondendo assim, a um dos objetivos desta aplicação.

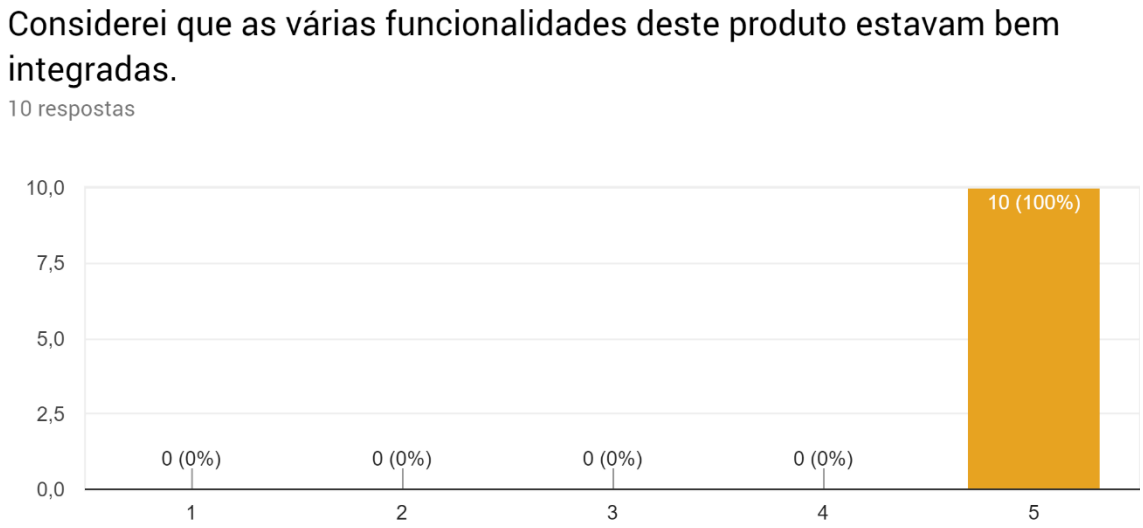

Figura 35 - Resultado pergunta 5

Pode-se concluir através da Figura 35 que as funcionalidades da aplicação foram bem definidas e implementadas.

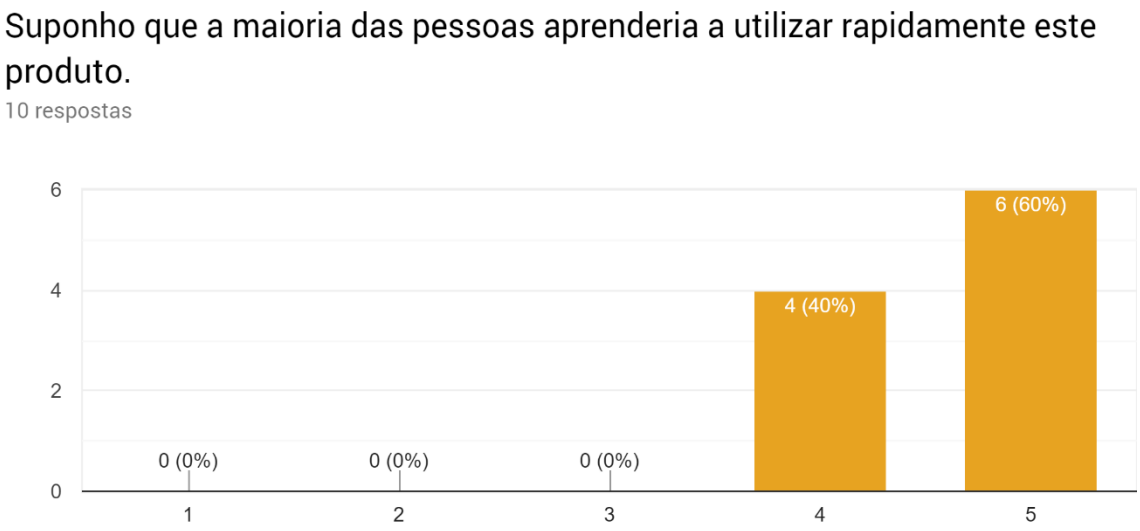

Figura 36 - Resultado pergunta 7

Analisando a Figura 36 pode-se verificar que a aplicação corresponde a um dos requisitos não funcionais proposto no capítulo 3 deste documento, sendo este a usabilidade (fácil aprendizagem).

5.3 Testes

Os testes que se seguem têm como objetivo validar os requisitos funcionais já definidos neste documento, estes são de extrema importância pois ajudarão a entender se a aplicação funciona corretamente e se corresponde aos objetivos propostos para esta dissertação, ajudará também, caso necessário, a definir trabalho futuro a ser realizado.

Questionário

Enumerar-se-á agora as 14 questões colocadas aos inquiridos:

1. Conseguiu criar um evento?
2. Foi possível visualizar os eventos disponíveis?
3. Conseguiu adicionar um produto?
4. A Aplicação permitiu registar o *QR Code/barcode* associado ao produto utilizando a camera do *smartphone*?
5. Foi possível adicionar uma foto ao produto?
6. Conseguiu efetuar um movimento de entrada/saída de um produto no evento criado?
7. Conseguiu dar entrada/baixa de stock de um produto?
8. Ao efetuar movimentos num evento utilizou o QR Code/Barcode para encontrar o produto pretendido?
9. Foi possível verificar quais os produtos utilizados no evento?
10. Conseguiu terminar o evento?
11. Conseguiu visualizar os relatórios dos eventos na aplicação?
12. Quando o produto foi adicionado, alterado ou eliminado por um colaborador conseguiu visualizar o mesmo em *realtime*?
13. Ao efetuar o movimento conseguiu verificar a atualização de stock ou a remoção do produto em *realtime*?
14. Conseguiu efetuar o *logout* da sua conta na aplicação?

Resultados

Este segundo questionário foi realizado aos mesmos inquiridos anteriormente descritos no questionário SUS. Estas questões eram de resposta “Sim” ou Não”. Das perguntas comuns aos dois questionários, em que se obteve 90.72% de repostas positivas e 9.28% de resposta negativas, é possível verificar o correto funcionamento da Aplicação nas tarefas propostas aos utilizadores descritas no cenário 1. Foram também efetuadas duas questões sobre a utilização da aplicação em real time, cenário 2, da qual se obteve os seguintes resultados de um uni verso de 6 pessoas: 66.7% dos inquiridos respondeu positivamente e 33,3% negativamente.

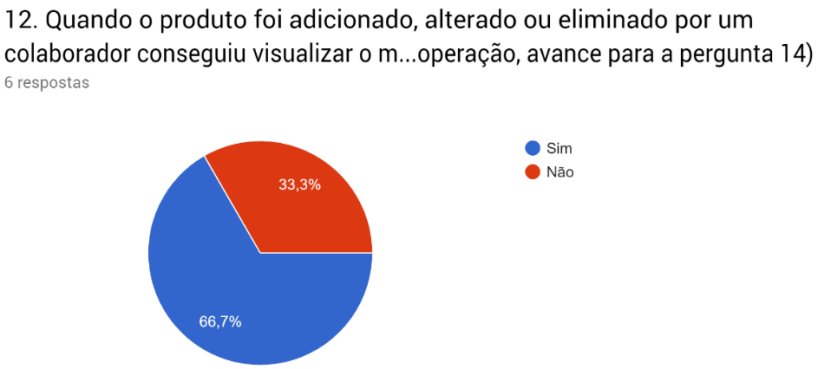

Figura 37 - Pergunta 12

Em análise verificou-se que 67,7% dos utilizadores consegui efetuar a conexão ao servidor *WebStocks* e efetuar as operações propostas no cenário 2.

Na questão 13 verificou-se 83,3% de respostas positivas e 16,7% negativas.

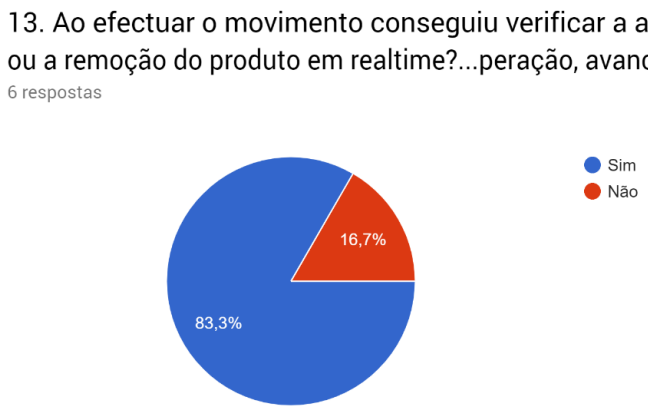

Figura 38 - Pergunta 13

Nesta questão foi possível verificar que a maioria dos utilizadores conseguiu estabelecer a conexão ao servidor *WebStocks* e efetuar as operações com sucesso.

Após análise destas duas questões, foi possível verificar que a implantação do servidor de *WebStocks* foi realizada com sucesso.

Sugestões e Comentários

Juntamente com os questionários acima descritos foi colocada uma pergunta de resposta aberta aos inquiridos “Por favor, forneça comentários/sugestões acerca da Aplicação que experimentou.”, apresenta-se agora estas sugestões e comentários na tabela que se segue:

Tabela 7 - Sugestões e Comentários

1	Boa aplicação para empresas do ramo. Aplicação User Friendly.
2	Util, fácil de utilizar, não foi necessária ajuda para realizar as tarefas propostas.
3	Após terminar o evento, na View Relatórios, seria importante a implementação de um Botão ou Lista, para a reposição de Stock. Exemplo... Repor Stock: -> reposição do stock gasto neste evento; -> reposição de stock personalizada (aqui aparecer formulário para meter os números a dar update, pois pode ser diferente dos números que se deu baixa durante o evento A funcionalidade de dar entrada e baixa através do QR Code é muito prático e funciona muito bem.
4	Aplicação ótima para a logística de um evento.
5	A utilização do barcode facilita no momento de efetuar as saídas/entradas de produtos para o evento. Sugestão: Dividir os produtos por armazéns.
6	Aplicação bem estruturada e de fácil utilização. Como sugestão: Estudar a possibilidade de gerar uma guia de transporte.

Serão tomadas em consideração as sugestões/comentários descritos pelos inquiridos.

6. CONCLUSÃO E TRABALHO FUTURO

Neste projeto assumiu-se o desenvolvimento de uma aplicação móvel colaborativa, ligada a uma *API* e a um servidor de *WebStocks* de forma a possibilitar a troca de informação em tempo real.

Numa primeira fase, foi possível aprender e desenvolver as competências necessárias para a utilização de várias tecnologias, destacando o *Laravel* e o *React Native*.

O estudo do Estado da Arte permitiu observar soluções tecnológicas essenciais para o desenvolvimento da aplicação bem como, efetuar uma análise de mercado no que diz respeito às aplicações de gestão de stocks.

Foi realizado o levantamento dos requisitos da aplicação e definida a arquitetura da mesma, que possibilitou e facilitou a criação da aplicação, diminuiu a margem de erros e proporcionou uma maior rapidez na elaboração.

Das tecnologias encontradas, foram analisadas as mais relevantes e populares, de acordo com o que se pretendia responder nesta dissertação. Após esta análise foi feita a comparação das várias soluções com o intuito de perceber quais poderiam ser utilizadas na aplicação de gestão de stocks para eventos.

Com a análise dos requisitos e arquitetura do sistema foi possível definir quais as funcionalidades a serem implementadas e identificar os elementos relevantes para a mesma.

A segunda fase desta dissertação engloba o desenvolvimento dos componentes, os testes à aplicação e correção de erros detetados.

Foram realizados e analisados testes de usabilidade e de requisitos funcionais com utilizadores reais, de forma a validar a aplicação desenvolvida. Com o decorrer da dissertação e passando pelos vários pontos de estudo, foi possível chegar ao objetivo final.

O propósito desta dissertação foi criar uma aplicação móvel para munir os colaboradores de empresas de eventos com uma ferramenta que permitisse efetuar uma gestão de stocks de forma colaborativa, simples e intuitiva, de modo a resolver um processo que até então era complexo e com poucos resultados.

Num futuro próximo pretende-se criar um sistema de gestão web, de forma a abranger o máximo de utilizadores deste tipo de empresas, quer por facilidade de acesso quer por uma questão de operacionalização. Na aplicação móvel, dividir os produtos por armazéns, estudar a possibilidade de criar guias de transporte e melhorar as funcionalidades existentes.

REFERÊNCIAS

Android Authority. (2017). Developing for Android vs developing for iOS – in 5 rounds. Obtido de http://www.androidauthority.com/developing-for-android-vs-ios-697304/

App Annie. (2016). Site Oficial App Annie. Obtido de https://www.appannie.com/en/

centroatl. (2016). Mysql. Obtido de http://www.centroatl.pt/titulos/tecnologias/imagens/excerto-e-book-ca-oguiapraticodomysql.pdf

Chester SW. (2018, Julho). Stock and Inventory Simple. Obtido de https://play.google.com/store/apps/details?id=com.stockmanagment.next.app&hl=en_US

Coleman, J. (2011). QR Codes: What Are They and Why Should You Care? Obtido de https://newprairiepress.org/culsproceedings/vol1/iss1/3/

educba. (2017). Laravel vs Lumen Comparison Table. Obtido de https://www.educba.com/laravel-vs-lumen/

Erkkilä, J.-P. (2012). *WebSocket Security Analysis*. Obtido de https://juerkkil.iki.fi/files/writings/websocket2012.pdf

Goadrich, M., & P. Rogers, M. (2011). *Smart Smartphone Development: IOS versus Android*.

Gusti, Y. (2018). Symfony vs Laravel, analysis of top PHP frameworks. Obtido de https://thinkmobiles.com/blog/symfony-vs-laravel/

Ionic. (2018). Site Oficial Ionic. Obtido de https://ionicframework.com/

Laravel. (2018). Site Oficial Laravel. Obtido de https://laravel.com/

Lumen. (2018). Site Oficial Lumen. Obtido de https://lumen.laravel.com/

Martins, A. I., Rosa, A. F., Silva, A., & Rocha, N. P. (2015). European Portuguese Validation of the System Usability Scale (SUS).

Matt, W. (2016). Microsoft just made a brilliant acquisition by grabbing a hot startup called Xamarin. *Business Insider*.

Mike, P., & Rahman, S. (2011). *Remote Data Visualization through WebSockets*.

Oracle Corporation. (2018). Site Oficial Oracle Corporation. Obtido de https://www.oracle.com/pt/index.html/

Pociot, M., & Herten, F. V. (2018). Introducing laravel-websockets, an easy to use WebSocket server implemented in PHP. Obtido 24 de Junho de 2019, de Freek.dev website: https://freek.dev/1228-introducing-laravel-websockets-an-easy-to-use-websocket-server-implemented-in-php

Rasmussen, C. (2018). Hybrid vs Native app development.

React Native. (2018). React Native. Obtido de Repositório React Native website: https://facebook.github.io/react-native/

Richter, F. (2018). Infographic: What Sets iOS Apart From Android. Obtido 27 de Fevereiro de 2019, de Statista Infographics website: https://www.statista.com/chart/5930/adoption-of-ios-and-android-versions/

Roy, F. (2000). Architectural Styles and the Design of Network-based Software Architectures. Obtido 24 de Julho de 2019, de https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Sauro, J. (2011). The System Usability Scale.

Sisbi. (2018, Julho). Stock Controller. Obtido de https://www.xnr-sisbi.com/stock-portal/

socketo. (2019). Ratchet WebSockets for PHP. Obtido 24 de Junho de 2019, de http://socketo.me/docs/flow

statista. (2018). Google Play Store: Number of apps 2018. Obtido 27 de Fevereiro de 2019, de Statista website: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

Symfony. (2018). Site Oficial Symfony. Obtido de https://symfony.com/

TIOBE. (2018). Site Oficial TIOBE Index. Obtido de https://www.tiobe.com/tiobe-index/

Usage Statistics. (2018). Usage Statistics and Market Share of PHP for Websites. Obtido 26 de Fevereiro de 2019, de Usage Statistics and Market Share of PHP for Websites website: https://w3techs.com/technologies/details/pl-php/all/all

Wave Solutions. (2018a, Julho). Mobile Stock. Obtido de https://play.google.com/store/apps/details?id=com.wavesolutions.mobilestock&hl=pt_PT

Wave Solutions. (2018b, Julho). Site Mobile Stock. Obtido de http://mobilestock.pt/

Xamarin. (2018). Site Oficial Xamarin. Obtido de https://visualstudio.microsoft.com/xamarin/

Xcode. (2018). Site Oficial Xcode. Obtido de https://developer.apple.com/xcode/