**Universidade de Aveiro**

**Ano 2016**

Departamento de Eletrónica, Telecomunicações e Informática

**João Pedro Oliveira Pedrosa**

# Loja de retalho do futuro

**Universidade de Aveiro**

**Ano 2016**

Departamento de Eletrónica, Telecomunicações e Informática

**João Pedro Oliveira Pedrosa**

# Loja de retalho do futuro

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistemas de Informação, realizada sob a orientação científica do Doutor Joaquim Arnaldo Carvalho Martins, Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e pelo Eng. Ricardo Jorge Costa Machado, Project Management Officer da empresa Ubiwhere.

**o júri**

presidente

Professor Doutor José Manuel Matos Moreira
Professor Auxiliar da Universidade de Aveiro

vogais

Professor Doutor Fernando Joaquim Lopes Moreira
Professor Associado, Departamento de Inovação, Ciência e Tecnologia da Universidade Portucalense

Professor Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

Em primeiro lugar gostaria de agradecer ao meu orientador Doutor Joaquim Arnaldo Carvalho Martins, pelo seu apoio e pela disponibilidade sempre demonstrada para a realização desta dissertação.

Seguidamente agradecer ao Rui Costa e ao Nuno Ribeiro por me terem dado a oportunidade de realizar este projeto na Ubiwhere, onde tive sempre todos os recursos necessários para a realização do mesmo. Um agradecimento especial ao Ricardo Machado e ao Bruno Silva que estiverem sempre disponíveis para me dar toda a ajuda e apoio em todos os campos que ia necessitando, foram fundamentais para a conclusão desta dissertação.

**agradecimentos**

Agradecer a todos os professores com quem tive a oportunidade de me cruzar até aos dias de hoje, desde a pré-primária até hoje que foram uma fatia importante e ajudaram a tornar no que sou hoje.

De seguida gostaria de agradecer à minha família, ao meu pai, à minha mãe e à minha irmã, que estiveram sempre presentes para me auxiliar ao longo não só do desenvolvimento desta dissertação mas em todo o meu percurso na universidade. Sem eles não seria possível estar onde estou hoje. Além disso, agradecer também ao resto da minha família, especialmente à minha tia Irene e primas Diana e Marisa, que sempre se preocuparam ao longo deste percurso.

Gostava também de agradecer a todos os meus amigos que sempre estiveram lá para me apoiar e ajudar em todas as dificuldades. Não querendo individualizar, quero agradecer especialmente ao João Couto, Hugo, André, Cristina, Bruno Marques, Cristiana, Diana, Fani, Rui, Soraia, os meus amigos de longa data que sempre estiverem lá. Ao Tiago, Rafael Coimbra, Hugo Correia, Rita, Costa, Renato, Valente Margarida, Adriana e Ivo, amigos que a universidade me deu e sempre me apoiaram e ajudaram em tudo. E ainda a todas as pessoas que não individualizei mas sabem que estou muito grato, quer sejam pessoas com quem tive a felicidade de me cruzar ao longo destes anos pela universidade, quer a todas as pessoas com quem me cruzo sempre no Lanheiro e na Ilha que são grandes amigos e foram fundamentais nesta fase (Telma, Nuno, Soraia, Fábio, Acácio, David, Tiago, Vanessa, Wilson e todos os outros que me esqueço de referir mas sabem que são importantes).

Um agradecimento especial também ao Artur Francisco que ao longo deste percurso universitário sempre me apoiou e esteve sempre lá para me ajudar. E ainda ao César Couto e à Placidina por terem disponibilizado o espaço para a concretização do meu caso de uso.

Quando se trata de agradecimentos as palavras são sempre poucas, por isso mesmo, simplesmente obrigado a todos!

**palavras-chave** Smartphone, consumidores, retalhistas, realidade aumentada, indoor-location, beacons, Android, padrão de compra, promoções

**resumo**

O mundo em que vivemos atualmente está em constante revolução devido à evolução tecnológica que se tem vindo a verificar nos últimos anos. O que é agora para nós normal e natural neste mundo, daqui a uns anos pode já nem o ser. E um dos mercados que tem sido mais afetado com esta explosão a nível tecnológico tem sido o mercado do retalho.

Com o aparecimento das lojas *online* e compras através da internet em que entregam os produtos em nossa casa, são cada vez menos as pessoas que se deslocam a lojas físicas para efetuar as suas compras, o que pode levar a um possível desaparecimento das lojas físicas num futuro não muito distante.

Assim sendo, as lojas têm de evoluir com a tecnologia e aproveitar os recursos que esta fornece, e nada melhor para isso que utilizar os aparelhos tecnológicos, que qualquer consumidor tem em sua posse, o seu *smartphone*. As lojas físicas têm de fazer com que os seus consumidores se desloquem à loja, e nada melhor para isso que conhecer bem o consumidor, dando-lhe uma experiência de compra que não poderá ter fazendo compras *online* e oferecendo-lhe promoções exclusivas só porque este se deslocou à loja.

E a chave para desbloquear este problema é o *smartphone*, com ele é possível criar uma aplicação que crie uma nova experiência de compra para o cliente utilizando tecnologias como *geofencing*, realidade aumentada, *indoor location* e *beacons*. E em simultâneo vá recolhendo métricas que ajudem o dono da loja a conhecer melhor o seu cliente, podendo assim oferecer promoções e descontos exclusivos.

A aplicação foi desenvolvida em *Android* e o *backend* foi desenvolvido em *Django*, tendo sido testada num caso de uso real.

**keywords** Smartphone, consumers, retails, augmented reality, indoor-location, beacons, Android, buying pattern, promotions

**abstract**

The world we live in today is in constant revolution due to the technological evolution that has taken place in recent years. What is now normal and natural for us may no longer be, in a few years. And one of the markets that has been most affected by this explosion in technology is the retail market.

With the emergence of online stores and shopping over the internet in delivering the products in our home, less and less people, go to brick and mortar stores to make their purchases, which can lead to a possible disappearance of brick and mortar stores in a near future.

Thus stores must evolve with technology and take advantage of the its resources. And nothing better for that, than using the technological devices that almost all consumers already have in their possession, the smartphone. Brick and mortar stores have to make their consumers moving to the store, and for that, they must know them in order to provide an unique shopping experience, that cannot be achieved with online shopping. For instance, they can offer them exclusive promotions, just because they went to the store.

The key to unlock this problem is the smartphone. It allows to create a new and innovative shopping experience for the customer using technologies such as geofencing, augmented reality, indoor location and beacons. It also allows to collect data regarding consumers behaviour, that can help the shop owners to get to know them better. Thus being able to offer promotions and exclusive discounts.

The application was developed for Android and the backend was developed in Django, having been tested in a real use case.

# Conteúdo

# Índice Figuras

# Acrónimos

**NFC** – Near Field Communication

**API** – Application Programming Interface

**SDK** - Software Development Kit

**GPS** - Global Positioning System

**RFID** - Radio-frequency identification

**SO** – Sistema Operativo

**SMS** – Short Message Service

**NDK** - Native Development Kit

**UI** – User Interface

**SESDK** – Shopping Experience SDK

# 1 Introdução

Vivemos numa era em que os avanços tecnológicos se vão acontecendo de uma maneira tão rápida que se torna difícil todos os sectores acompanharem os mesmos. Cada vez mais no nosso dia-a-dia vemos novas ferramentas e funcionalidades a serem implementadas para atrair o cliente a deslocar-se à loja e efetuar uma compra. Estamos numa fase em que os donos das lojas querem que os seus clientes aprendam novos conceitos sobre o produto que estão a comprar, podendo desta forma fazer escolhas e receber ofertas personalizadas.

Esta customização que está a acontecer na loja física deve-se ao facto de cada vez mais pessoas efetuarem as suas compras *online* e se deslocarem à loja física apenas para experimentar o próprio produto ou para comparar preços. Com as pessoas a utilizarem a loja física somente para comparação e experimentação dos produtos, neste sentido os retalhistas têm de se adaptar e arranjar novas soluções para atrair os clientes à sua loja.

Aqui encontra-se um ponto fundamental: como atrair os clientes a irem à loja? É neste ponto que tem de se começar a prever como será a loja do futuro. O que irá querer o cliente na loja do futuro para se possa deslocar até ela? E como pode o dono duma loja saber o que o cliente quer? Esta é a outra pergunta crítica que se coloca quando falamos de como será a loja do futuro. O dono da loja terá de conhecer o seu cliente e saber o que ele poderá querer naquele momento baseando-se no sei histórico de compras entre outras métricas.

É neste sentido que se enquadra esta dissertação, englobada numa ideia geral que é a "*Retail Store of the Future*" um projeto da Ubiwhere [1], a empresa na qual se desenvolveu o trabalho conducente a esta dissertação no âmbito do Mestrado em Sistemas de Informação.

## 1.1 Descrição da Aplicação

A aplicação móvel tem o nome de "Shopping Experience" e tem como principal objetivo tentar juntar o "melhor dos dois mundos" relativamente ao setor do retalho. É pretendido que a aplicação crie uma experiência de compra inovadora para o consumidor recorrendo ao uso de tecnologias como o NFC, *geofencing*, realidade aumentada, *indoor location,* entre outras. Contudo é também pretendido que, enquanto o consumidor faz uso desta aplicação no seu processo de compra, a aplicação vá recolhendo dados e métricas que ajudem no futuro o retalhista a conhecer melhor o seu cliente, padrões de compra e ainda outras tendências de compra sazonais.

O que se pretende como produto final é criar uma aplicação que cumpra todos os requisitos supra citados e que tenha um *design* e *user experience* do agrado do consumidor. É ainda crucial que a aplicação seja transparente para o consumidor e que este não sinta que a sua privacidade seja ameaçada

## 1.2 Motivação e Objetivo Geral

A ideia surgiu uma vez que existe um enorme número de soluções específicas para cada uma das áreas (consumidor e retalhista), no entanto, não existem soluções que tentem tirar proveito e juntar essas duas áreas numa só aplicação. Além disso, muitas soluções requerem a instalação e uso de muito hardware externo (antenas, balanças, câmaras), enquanto que com esta solução precisamos simplesmente do *smartphone* do consumidor. Se o retalhista quiser aumentar a experiência de compra do consumidor pode ainda incluir *beacons,* visto que é o único dispositivo físico que vamos necessitar nesta aplicação.

Com o desenvolvimento desta aplicação pretende-se sobretudo que esta cumpra os seguintes objetivos:

- Conseguir atrair mais consumidores a deslocarem-se à loja física;
- Facilitar o processo de compra para o consumidor;
- Fornecer ao retalhista dados suficientes para conseguir garantir no futuro ofertas personalizadas ao consumidor/cliente;
- Possibilidade de integração de outras API's ou dados que ajudem a conhecer melhor o consumidor e o seu padrão de compra.

## 1.3 Público-alvo

O desenvolvimento desta aplicação, tem como intuito abranger duas entidades:

- **Consumidor** – A pessoa que vai usar e tirar maior proveito da aplicação. Será a pessoa a quem os dados e métricas serão recolhidos enquanto esta se encontra no processo de compra.

- **Retalhista** – É a pessoa dona ou responsável pelo estabelecimento/loja na qual a aplicação está associada. Será a pessoa que quer que os consumidores usem a aplicação para poder ficar a conhecer melhor os seus clientes.

Esta dissertação irá focar principalmente a criação da aplicação e a recolha dos dados. Não vai abordar nem efetuar a análise, nem o estudo dos dados que vão ser recebidos. Vai ser apenas ser realizado um sistema onde os dados irão ser armazenados de uma forma estruturada para futura análise.

## 1.4 Funcionalidades Gerais

A aplicação a ser desenvolvida contará com duas vertentes, consumidor e retalhista, no entanto, neste tópico, apenas serão enumeradas as funcionalidades gerais do lado do consumidor. Do lado do retalhista, vamos ter acesso a alguns serviços externos que irão enriquecer a fonte de dados, porque podem ser dados ou informações que influenciarão o deslocamento do consumidor até à loja ou ao processo de compra. É necessário ter esses dados em atenção, portanto as funcionalidades gerais da aplicação serão as seguintes:

- Login e registo através de redes sociais (*Facebook* ou *Twitter*);
- Efetuar uma lista de compras prévia com os produtos da loja;
- Receção de alertas de promoções ou alertas com a sua proximidade da loja;
- Receção de alertas de promoções ou alertas ao entrar numa zona específica da loja;
- Ver a sua localização em tempo real dentro da loja;
- Ver localização dos produtos que quer comprar dentro da loja;
- Possibilidade de fazer check-in e check-out na loja através de NFC;
- Efetuar pagamentos através de NFC e Bitcoins;
- Ver detalhes de produtos através de NFC e realidade aumentada;
- Localizar os produtos a comprar através de realidade aumentada.

# 2 Estado da Arte

Estamos na era de mudança e impulsionamento na área das tecnologias, e neste aspeto, o ramo do retalho não é exceção. Quando já era abordada a extinção certa das lojas físicas, devido aos avanços tecnológicas e ao comércio *online*, o sector do retalho decidiu beneficiar desses mesmos avanços tecnológicos e implementá-los para atrair o consumidor à sua loja física.

"*The future of retail will be about choice, consistency and customization*" [2]. Como a frase anterior demonstra, a loja de retalho do futuro será sobre escolha, consistência e customização, sendo que o cliente irá querer ofertas personalizadas e direcionadas para si quando se desloca à loja, assim como poder comparar o produto que pretende comprar com outros: saber de onde veio, como foi feito, entre muitos outros campos relevantes. Por fim, pretende ter consistência nas ofertas e vantagens em comprar na loja física em vez da loja *online*.

O tema geral desta dissertação será na área do retalho, que é por sinal um dos setores mais competitivos na Europa. Dados recolhidos no ano de 2010 mostram que este setor criou um valor acrescentado na ordem dos 451 biliões de euros, o que demonstra a importância deste setor e motivo pelo qual vale a pena inovar e apostar neste [3]. Os consumidores são a melhor forma de análise que as empresas têm para prever tendências de mercado, ajustarem os preços nas diversas áreas, adquirirem e distribuírem os produtos desejados pelo cliente. As lojas vão ter de se adaptar aos desejos e expectativas se pretendem continuar a ter clientes e a sua fidelidade.

A melhor forma de um retalhista conseguir personalizar as ofertas para o seu cliente é conhecendo o mesmo, até porque o próprio cliente do futuro vai ser completamente diferente do cliente atual. Uma das tendências que já está a acontecer nesta mudança de estilo de compra do cliente é a tecnologia fazer parte do processo de compra.

Já existem várias empresas a tentar explorar este mercado de recolher métricas para poder personalizar ofertas para o seu consumidor. Tanto em Portugal como no estrageiro é uma área que está em grande expansão. Contudo o principal problema com estas soluções criadas pelas empresas é que são bastante dispersas, ou seja, nenhuma delas oferece uma abrangência total que satisfaça as duas vertentes necessárias a ter em conta neste sector, o retalhista e o consumidor. De forma isolada já existem soluções bastante bem desenvolvidas e embutidas no mercado, sobretudo no mercado americano, como a *Aisle411* [4] que já tem parcerias com a *Toysrus* [5] e a *Walgreens* [6]. Contudo a *Aisle411* aposta sobretudo na experiência de compra por parte do consumidor tirando partido dos *smartphones* ou *tablets* como um auxiliar durante o processo de compra de forma a criar essa experiência, deixando assim de parte a recolha de dados relativos ao consumidor. Estes dados são fundamentais para o retalhista conseguir adaptar a sua loja aos desejos deste. A solução apresentada pela *Aislee411* tira partido dos dados obtidos através do *smartphone* e dos seus sensores, alcance e força de sinal de *wireless* e informação proveniente de outros dispositivos que se encontrem distribuídos pela loja. Um dos exemplos são os *beacons*, que além de conterem um sistema bastante desenvolvido e funcional de realidade aumentada e *indoor location*.

Em Portugal uma das empresas que se tem destacado mais no processo de reconhecimento do cliente é a *Movvo* [7].

A *Movvo* é uma empresa sedeada no Porto e tem como principal objetivo ajudar os donos de lojas ou centros comerciais a conhecerem melhor os seus consumidores, fazendo uso de recolha de dados quando estes estão no processo de compra na respetiva loja. Utilizam os *smarphones* e antenas dos consumidores para recolha de algumas métricas que ajudem a reconhecer os comportamentos deste. Usam também *Cloud Computing* e *Big Data* para armazenar e processar os dados e graças a esses algoritmos conseguem obter informação relevante. Por fim, uma *dashboard* onde os seus clientes podem ver todas a métricas recolhidas e os resultados obtidos. Alguns dos resultados que a *Movvo* oferece com esta solução são a possibilidade de conhecer os caminhos mais usados pelo consumidor dentro da loja, de identificar as caixas de pagamento mais eficientes, de perceber quais os ecrãs dentro da loja que atraem mais atenção, de comparar dados da mesma loja em localizações diferentes e ainda de perceber a resposta dos consumidores a promoções em tempo real. É uma solução inovadora em Portugal, contudo, esta solução só traz vantagens exclusivamente para o dono da loja que assim pode ficar a conhecer melhor os seus clientes, sendo que o consumidor não tem vantagens com este serviço.

Outra das empresas que está a tentar apostar neste mercado é a *SwiftIQ* [8], contudo com uma abordagem ligeiramente diferente. O objetivo da *SwiftIQ* é ajudar os donos das lojas a tirar valor dos dados que contêm, ou seja, fornece mecanismos e algoritmos para onde podemos enviar os nossos dados e recolher informações relevantes sobre os nossos consumidores. Para ajudar este processo é ainda disponibilizada uma API que facilita a obtenção de resultados a partir dos nossos dados.

Por fim, temos ainda a empresa *RetailNext* [9], que é provavelmente a que tira mais proveito de todas as métricas que podem ser recolhidas numa loja. Tem várias fontes de recolha de dados, como por exemplo, câmaras, dispositivos *wifi* e *bluetooth*, sistemas de *staff*, pagamentos através de cartões, informações meteorológicas, entre muitas outras. No fim da recolha dessas informações tem várias formas de as poder utilizar como visualizar as informações, numa *dashboard*, numa aplicação móvel, prever análise, entre muitas outras.

Todas as empresas mencionadas focam-se em recolher informações para os donos das lojas conseguirem reconhecer os seus consumidores e tirar mais valor e proveito dos mesmos. Contudo, nenhuma destas empresas cria ou oferece vantagens para o lado do consumidor, ou seja, o consumidor até pode estar disposto a fornecer mais informações relativas a si desde que graças a isso consiga ter ofertas personalizadas ou uma experiência de compra diferente. Porém, existe também uma empresa portuguesa que tenta criar esse novo processo de compra para o consumidor, a *Xhockware* [10].

A *Xhockware* criou uma aplicação que torna o utilizador autónomo dentro da loja onde pode fazer a sua lista de compras, ver o seu histórico de compras, entre muitas outras funcionalidades. Porém, esta aplicação serve apenas de auxiliar no processo de compra e não engloba grande parte do cenário futurista que poderemos ter nas lojas.

É possível observar que já existe bastante exploração neste mercado, contudo as empresas analisadas focam-se só numa das vertentes: ou na recolha de dados para o dono da loja conseguir analisar ou na experiência de compra para o utilizador. Mas é possível tirar proveito destas duas vertentes de uma só maneira, sem envolver custo extra na aquisição de equipamento. O consumidor do futuro vai estar equipado de um aparelho que tanto pode recolher as métricas que são precisas como criar uma nova experiência de compra, o *smartphone*.

Os tempos em que os telemóveis serviam para fazer chamadas e mandar mensagens já lá vai. Hoje em dia, existem os *smartphones*, dispositivos que fazem exatamente o mesmo que os antigos telemóveis, mas vêm equipados com características e funcionalidades que permitem que os utilizadores os utilizem em praticamente todos os processos do seu quotidiano. Os *smartphones* atuais já vêm equipados com ligação à internet, seja através de *wireless* ou dados móveis, um grande número de sensores (giroscópios, acelerómetros, magnetómetro, entre outros, dependendo da marca e modelo), GPS, NFC, *bluetooth*, entre muitas outras funcionalidades. Através dessas características e dispositivos que vêm embutidos nos *smartphones*, temos uma ferramenta bastante poderosa e que acompanha por norma o cliente enquanto este se encontra no processo de compra. A cada ano que passa os *smarphones* vão incorporando mais funcionalidades, o que pode ser bastante interessante, já que com a mesma ferramenta podemos encontrar cada vez mais informação.

"*The future retail experience will be supported by a number of transformative technologies that leverage and complement consumers' mobile devices. These technologies are already enabling retailers to create richer experiences for consumer through their mobile devices*" [11]. Como é possível verificar pela citação anterior, o futuro da experiência de retalho irá basear-se numa experiência de compra totalmente nova, envolvendo algo pessoal do cliente, o seu *smartphone*. Hoje em dia é raro existir uma pessoa sem *smartphone*, e sendo um objeto bastante pessoal do cliente, os retalhistas veem aí uma boa oportunidade para melhorar a experiência de compra e consumo do cliente. Segundo um estudo de 2012 referenciado no mesmo artigo da citação anterior, foi descoberto que 43% dos utilizadores que usam *smartphones* instalaram a aplicação móvel das suas lojas de retalhos, mas somente 14% desses é que acharam que a aplicação foi útil para os ajudar a fazer uma compra. Através deste estudo, podemos concluir que a utilização de aplicações nas lojas por parte dos consumidores terá cada vez uma aceitação maior. Contudo, estas têm de criar uma experiência de compra melhorada se quiserem cativar e ajudar o cliente no seu processo de compra.

Conseguimos assim analisar a grande influência que os *smartphones* virão a ter na área do retalho, mas em termos monetários ainda não conseguimos ter uma análise concreta. A *Delloite* [12] fez recentemente um estudo prevendo qual a influência que os *smartphones* podem ter na área do retalho, esses mesmos resultados podem ser vistos na imagem abaixo. [13]

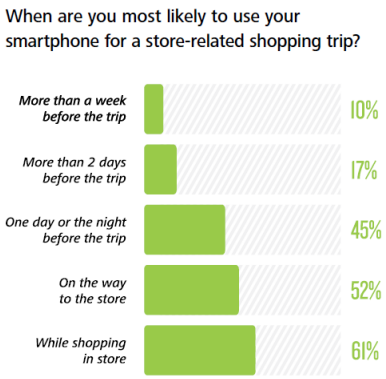



**Figura 1 - Influência dos *smartphones* no retalho** [13]

Como podemos comprovar, a área dos *smartphones* irá influenciar o retalho, tanto ao nível da experiência do consumidor como ao nível de transações. No fim de se saber que as aplicações das lojas de retalho irão ter um papel fulcral na loja do futuro, falta perceber o que é que o cliente deseja encontrar na aplicação para se sentir atraído a usá-la no seu processo de compra. "*Retailers should consider developing new or improved apps that provide key tools and information to help and influence shoppers ate each stage of the decision-marking process – especially while in the store*" [13]. A frase anterior foi citada do mesmo artigo da Deloitte, e demonstra o que os retalhistas têm de ter em atenção quando desenvolvem uma aplicação para os seus consumidores. Ainda no seguimento deste estudo, foi questionado aos consumidores quando é que é mais comum utilizarem o *smartphone* quando vão a caminho duma loja para efetuar compras. Os resultados deste estudo estão referidos na imagem abaixo, que evidencia perfeitamente a preferência dos clientes na utilização do *smartphone* em relação à loja.

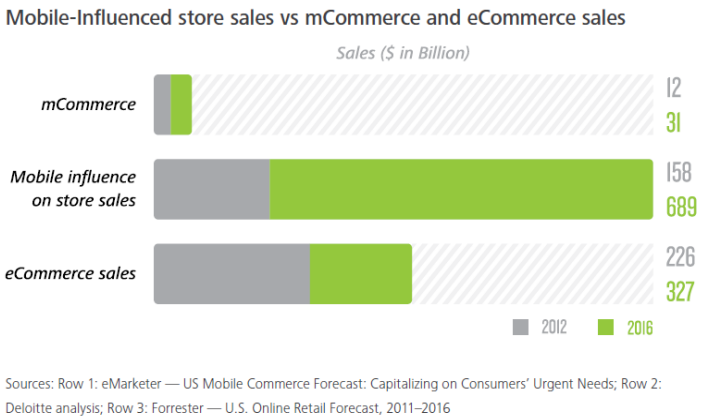



**Figura 2 - Influência do *smartphone* no processo de compra na loja** [13]

Através destes resultados é possível observar que o cliente utiliza mais o seu *smartphone* no *eCommerce*. No entanto, segundo as previsões deste mesmo estudo, já no próximo ano este aspeto vai mudar. O consumidor irá utilizar mais o *smartphone* dentro da loja e enquanto efetua o seu processo de compra. Podemos então concluir que neste momento, mesmo sem existirem aplicações que interliguem a experiência de compra numa aplicação, o cliente já inclui o seu *smartphone* no seu processo de compra.

Como já foi referido, os *smartphones* podem melhorar a experiência de compra do utilizador e aumentar a sua vontade de se deslocar à loja física. Neste sentido, podemos abordar outra perspetiva, como é que os *smartphones* podem ajudar o retalhista. Além da resposta que já falamos em cima (aumento na receita), os retalhistas podem tirar muito mais vantagens se os seus clientes utilizarem os *smartphones* no processo de compra. Enquanto os clientes estão nesse processo interagindo com toda a experiência de compra criada através da aplicação, podem ir recolhendo métricas que no futuro ajudarão a conhecer padrões de compra e a conhecer melhor quem é o seu cliente.

Outra questão que se coloca é, como é que recolher padrões e conhecer melhor o cliente vai ajudar melhor o retalhista? A resposta é simples, reconhecendo padrões de compra. O retalhista sabe o que o cliente gosta e quer comprar, e conhecendo melhor o seu cliente pode oferecer-*lhe loyality rewards* ou mesmo ofertas e promoções especializadas, aumentando assim a vontade dos consumidores em se deslocarem à loja física.

Para finalizar o tópico relativo aos *smartphones*, temos de ter em atenção qual é o mais comum e usual. Em termos de modelos e marcas variam muito, mas em relação ao SO existem basicamente dois concorrentes no mercado: o *Android* [14] e o *iOS* [15]. Assim sendo, a aplicação será desenvolvida para *Android*, pois tem maior cota de mercado a nível mundial (nos EUA a sua cota está praticamente 50/50) e assim, em termos de utilizadores, é possível ter uma maior abrangência. Outra das causas que apontou para esta direção foi o facto de alguns dos serviços que serão usados ainda não terem compatibilidade com *iOS* (como o caso do *NFC*), *Windows Phone* [16] e outros sistemas.

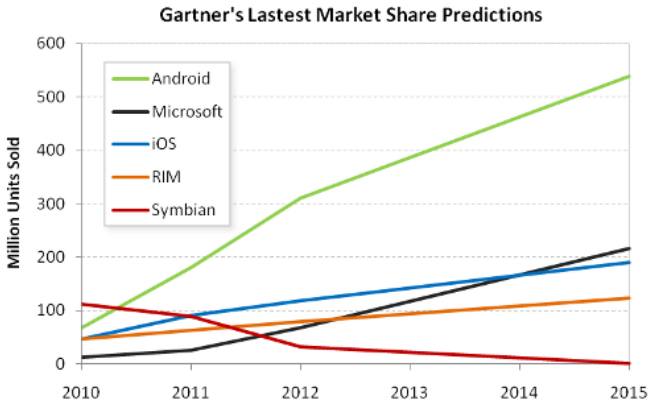



**Figura 3 - Cota de mercado dos principais SO na indústria móvel**
[98]

## 2.1 Objetivos Específicos

Através da análise feita nos tópicos acima, evidencia-se que a experiência de compra do cliente irá fazer certamente parte da loja do futuro, por isso, o tema proposto para esta dissertação é desenvolver uma aplicação que crie o cenário futurista da experiência de compra na loja do futuro para o cliente, e ao mesmo tempo, recolher métricas que no futuro poderão ajudar o gerente da loja a conhecer padrões no seu cliente.

Neste sentido, os objetivos específicos propostos para esta dissertação são os seguintes:

- O cliente deverá ver uma lista de produtos e conseguir efetuar a sua *shopping list.*
- Integração de um serviço de *Indoor Maps* para ser possível ao cliente saber a sua localização na loja, localização dos produtos que procura e ainda para o gerente da loja saber os caminhos mais utilizados pelos seus clientes.
- Integração da tecnologia NFC para fazer *check-in* e *check-out* na loja, efetuar pagamentos, aderir a promoções e ainda digitalizar produtos para obter os seus detalhes
- Utilizar *geofencing* e *beacons* para chamar os utilizadores à loja quando passam perto de uma e lançar promoções dentro da própria loja.
- Integração de realidade aumentada para ser possível ao cliente visualizar promoções na loja e informações dos produtos.
- Integração de *Mobile Payments.*
- Integração de redes sociais (*Facebook*, *Twitter*) para efetuar o login e guardar informação relevante do cliente.
- Integração API's que ajudem a conhecer o padrão de compra dos clientes.
- Recolher informações dos sensores disponíveis no telemóvel.
- Desenvolver uma API.

Enumerados os objetivos mais específicos que farão parte do processo de desenvolvimento desta aplicação, seguidamente serão explicadas em detalhe as tecnologias ou bibliotecas que estão idealizadas para o desenvolvimento da aplicação acima referida.

## 2.2 NFC

Uma das tecnologias a incorporar neste projeto será o NFC. "*Near Field Communication or NFC is an emerging technology for electronic devices which allows them to communicate with each other by simply touching or bringing them very close to each other*" [17]. O NFC é uma tecnologia que nos permite comunicar, transmitir ou receber dados somente encostando o nosso *smartphone* a outro *smartphone* (NFC Beam) ou a uma NFC Tag. Esta tecnologia é recente mas já começa a ser usada por muitos sectores, por exemplo para pagamentos e validações de dados. Também já se começa a observar esta tecnologia aplicada ao sector do retalho. "*Different applications for NFC were named and described, of which five were identified as deploy able for the use in retail stores: Payment, the download of information, loyalty applications, rebate coupons, and product information*" [18]. As vantagens de incluir uma tecnologia como esta numa aplicação destinada à área do retalho são muitas, facilita o processo de escolha e de compra do cliente, poupando tempo e não só.

Numa área mais técnica, a NFC Tag é composta por uma antena e por um circuito integrado, sendo que não precisa de nenhuma bateria, porque consome a energia do *smartphone* quando este se encosta a ela. Quando ativada, envia a informação em si armazenada para o *smartphone*, sendo que o máximo de distância para ser possível comunicar através de NFC são 10cm e com uma taxa máxima de transmissão de dados de 424Kb/s.

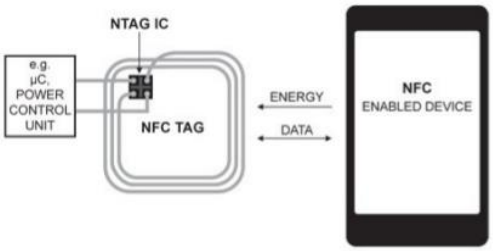



**Figura 4 - Processo de leitura da NFC Tag** [99]

Já existem algumas empresas a apostar na exploração destas tecnologias noutros mercados e cenários diferentes de onde ela é normalmente utilizada. Uma das empresas que está a iniciar essa inovação e mais virada para o sector do retalho é a *TPG Rewards* [19]. O objetivo desta empresa é incluir tags NFC nas embalagens dos produtos que compramos diariamente, para que se consiga obter mais informações ou promoções dos mesmos. Com esta abordagem, o usual código de barras irá deixar de ser usado e irá dar lugar ao NFC, que oferece mais vantagens e funcionalidades. Outra das empresas que explora este mercado é a *Cloud Tags* [20], que tem incluído tags NFC nos seus produtos, dando assim liberdade ao cliente para saber mais informações sobre os produtos que está a comprar. Já conseguiram fechar parceria com cerca de 10 companhias nos EUA, onde já têm o seu serviço a funcionar.

## 2.3 Beacons

Os *beacons* são dispositivos *bluetooth* que servem para transmitir mensagens ou solicitar ações a outros dispositivos. O alcance destes dispositivos depende muito do modelo e do próprio dispositivo *bluetooth*, mas pode variar entre um raio de 30 até 70 metros, o que para estes dispositivos bastante pequenos e portáteis é uma vantagem.

O modo de funcionamento é simples, o *beacon* deteta a proximidade de outros dispositivos e transmite um número identificador único, que é então recebido pelo *smartphone*. Posteriormente, o *smartphone* pode fazer diversas ações como enviar notificações ou mudar algum estado. De referir ainda que o *beacon* só emite informação, nunca recebe informação proveniente do *smartphone,* estando, por isso, assegurada a privacidade e segurança para o utilizador.


**Figura 5 - Esquema de funcionamento dos beacons**

Os *beacons* são uma tecnologia recente que começa agora a ser conhecida, sendo usada essencialmente em campanhas de marketing ou para enviar promoções. Os *beacons* são dispositivos que se conectam com um *smartphone* via *Bluetooth* e podem desempenhar ações quando esses *smartphones* entram ou saem de um raio definido de x metros. Os *beacons* já são usados por grandes cadeias de lojas nos EUA, por exemplo a *Target* [21], para enviar promoções aos seus clientes dentro da loja. Este envio de notificações com promoções pode aumentar o impulso de compra por parte do cliente.

Segundo um estudo conduzido pela *SWIRL* [22] e mencionado no artigo "*The Storefront of the Future*", 60% dos clientes abrem as notificações enviadas pelos *beacons* e 73% dizem que com estas ofertas ficam mais tentados a efetuar a compra do produto [23]. Através desta informação conseguimos perceber que os *beacons* podem ter um grande papel em atrair e impulsionar os clientes a efetuar uma compra dentro da loja. E neste caso estamos a falar do envio de notificações com ofertas espontâneas. E se se analisar a taxa de abertura das notificações ou mesmo de conversão em compra para ofertas especializadas para os consumidores, estes valores poderiam ser bastante mais elevados.


**Figura 6 - Resultados relacionados com a influência e abertura de notificações enviadas através de *beacons*** [100]

Uma das empresas que se encontra mais desenvolvida neste âmbito dos *beacons* é a *Estimote* [24]. Esta empresa tem a sua própria gama de *beacons* e ainda fornece um SDK de desenvolvimento para os mesmos. Ao contrário das outras empresas desta área, a *Estimote* fornece também um SDK de *indoor location* que irá ser abordado num tópico mais em baixo. Os *beacons* são conectados com os dipositivos móveis via Bluetooth e são alimentados por uma bateria, tendo uma duração de vida de cerca de 3 anos e um alcance de cerca de 70 metros. Além disso, vem ainda com alguns sensores como acelerómetros e sensores de temperatura, que podemos usar para recolher informação extra ou acionar ações consoante os parâmetros dos mesmos.


**Figura 7 - Esquema de funcionamento dos Estimote Beacons** [101]

## 2.4 Geofencing

A tecnologia *geofencing* pode ser definida como a definição de um limite (por norma em metros) para uma área. É uma tecnologia que utiliza o GPS ou RFID para definir fronteiras geográficas. A área que fica abrangida é a área de atuação do *geofencing* que pode definir ações a efetuar quando alguém entra, sai ou está um determinado tempo naquela área.

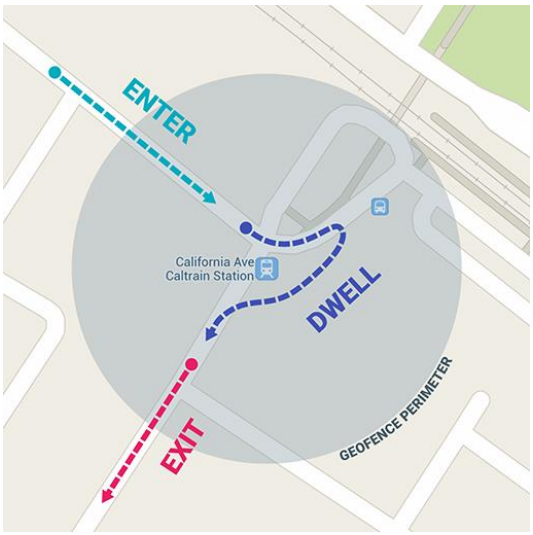



**Figura 8 - Funcionamento do geofencing** [102]

Esta tecnologia já é usada um pouco por todo o mundo principalmente por espaços comerciais, para notificarem as pessoas que passam perto das suas lojas físicas acerca de promoções e outros alertas.

"*Geofencing, which is starting to appear on the latest generation of phones, allows the user to configure some specific actions when he enters a particular perimeter*" [25]. Como a frase anterior indica, o *geofencing,* assim como os *beacons,* é uma tecnologia que está a ficar na moda devido ao seu potencial de atrair os clientes até à loja. Esta tecnologia pode parecer semelhante aos *beacons,* mas é ligeiramente diferente. Enquanto os *beacons* são mais direcionados para ofertas dentro da própria loja, ou seja, quando o cliente já vai com intenções de efetuar uma compra, com o *geofencing* conseguimos enviar notificações a pessoas que estejam perto da loja e atraí-las a entrar na mesma para efetuar uma compra.

As ofertas exclusivas ou ofertas quando os consumidores entram numa determinada área ou secção, é um dos grandes atrativos destas novas tecnologias. A oportunidade de enviar notificações ou alertas com contexto, ou seja, quando o consumidor está no local onde pode proceder logo a compra e não quando nem sequer está na loja, pode aumentar em muito a conversão em compra por parte dos consumidores. "*The rise of geofencing and other location-based solutions capable of identifying individuals who are in a given area also opens up opportunities to personalize the store shopping experience."* [2]

Uma das empresas que se tem distinguido no uso desta tecnologia e na forma de criar soluções com a mesma é a *Plot Projects* [26]. Basicamente, a *Plot Projects* foca-se em possibilitar que aplicações enviem notificações através de *geofencing*.

Permite ainda a integração com *iBeacons* (*beacons* desenvolvidos pela Apple) na sua solução, contudo, o seu objetivo principal é aumentar o número de consumidores a entrarem na loja e a comprar produtos através do envio de notificações. Segundo um estudo realizado pelos mesmos, as notificações *geofencing* apresentam uma taxa de conversão (cliente que entra na loja e acaba por efetuar uma compra), 39% superior às notificações normais Este aspeto, poderá estar relacionado com o facto de estas notificações serem enviadas quando o consumidor se encontra próximo da loja, fazendo com que este fique mais tentado a entrar na loja quando recebe uma notificação do seu interesse. Também as aplicações que usam esta tecnologia apresentam um aumento de 200% na sua taxa de utilização, segundo a *Plot Projects*.

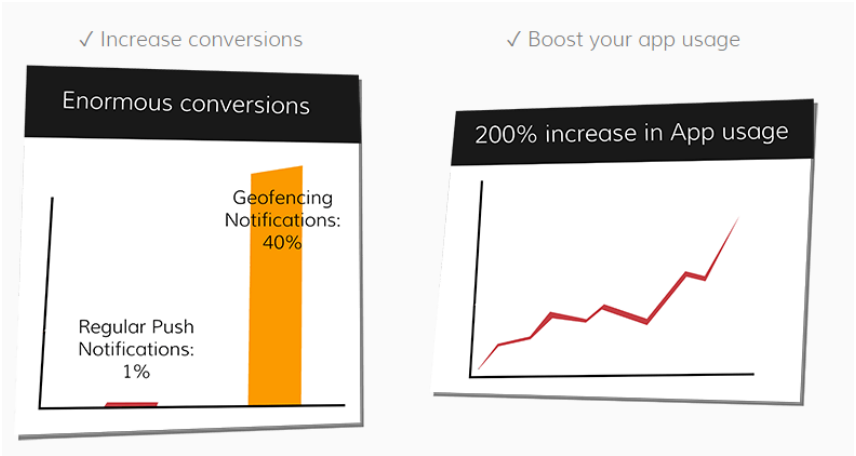



**Figura 9 - Estatísticas associadas a notificações de *geofencing*** [26]

Contudo, existem outras empresas que já exploram bastante este mercado de promoção baseado na localização dos consumidores. No Reino Unido, a *L'Oreal* [27] e a *Starbucks* [28] já fizeram um estudo [29] de mercado nesta área, enviando SMS para os seus clientes oferecendo descontos de 50% nos seus produtos se eles entrassem ou passassem a um determinado raio de uma das suas lojas. No entanto, o estudo não detalha qual foi a taxa de conversão (número de SMS's enviados/compras efetuadas), sendo que seriam resultados bastante interessantes a ter em conta.

## 2.5 Mobile Payments

Hoje em dia sabemos que é inevitável que, no futuro, sejam feitos pagamentos automáticos através do telemóvel, seja através de *wallets* digitais, cartões digitais ou moedas virtuais. E é aqui que entram na história os *bitcoins* [30], a moeda digital, mais conhecida e usada neste momento no mundo digital. "*Bitcoin is voluntary digital currency that can be transferred peer-to-peer over the Internet. The open-source cryptographic program secures electronic transactions without the need for a third party, like a bank or PayPal.*" [31]

A problemática associada a este tipo de moeda é a aceitação por parte das pessoas, principalmente ao nível da segurança. Dentro desta abordagem, foi realizado um estudo em que se questionava se já tinham conhecimento deste tipo de moeda e se estariam dispostos a usá-la no futuro. O estudo foi feito pela empresa *PwC* e podemos ver os resultados do mesmo na imagem abaixo. [32]

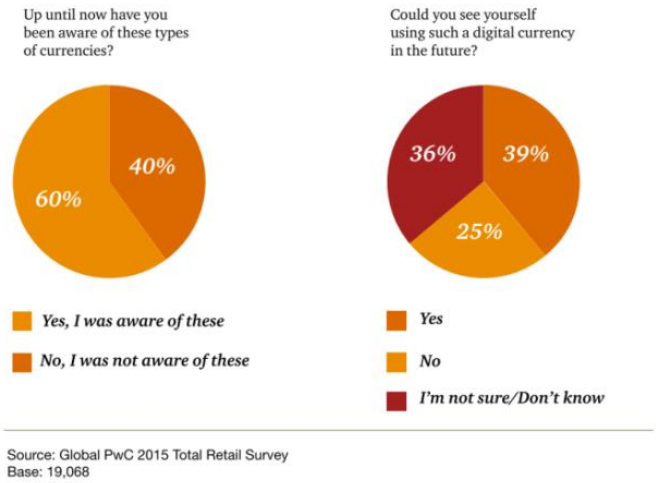



**Figura 10 - Estudo efetuado pela PwC relativamente ao conhecimento e uso das moedas virtuais** [32]

Analisando esta informação, conseguimos observar que 40% das pessoas ainda não tinha conhecimento da existência de moedas virtuais e que estas já eram usadas nos dias de hoje. Contudo, quando questionando se no futuro estariam dispostas a usá-las, apenas 39% respondeu que sim, enquanto que os restantes 61% não têm a certeza se iriam usar este tipo de moeda, muito provavelmente por acharem que não é segura e por não ser algo palpável.

Sabemos que atualmente a moeda virtual e as *wallets* virtuais já estão a ser bastante usadas em alguns países, mas o principal obstáculo a ultrapassar para esta se afirmar é conseguir chegar às pessoas de uma forma segura de efetuar pagamento. Quanto à ligação entre a moeda virtual e loja do futuro, é certo que as lojas terão de estar equipadas para aceitar este tipo de pagamento, para não perderem parte dos seus consumidores.

Contudo, e tendo em conta o receio das pessoas em usar a moeda digital, também vai ser integrado na aplicação um método de pagamento mobile, o *Stripe* [34]. Neste campo já existem enumeras soluções, contudo em Portugal as soluções não são assim tantas, limitando-se ao *PayPal*, *BrainTree* (*Beta Version*) [35] e o *Stripe* (*Beta Version*). O *Stripe* é visto por muitos como o substituto do *Paypal*, devido à sua simplicidade de processo e por ter taxas ligeiramente mais baixas que o *PayPal*. Devido à sua simplicidade de processos e para ser possível ter uma alternativa aos bitcoins, o *Stripe* irá ser também integrado nesta aplicação.

Umas das vantagens do uso desta tecnologia, é nunca transmitimos informação sensível do consumidor, especialmente informações do cartão de crédito. O utilizador pode inserir os seus dados do cartão de crédito na aplicação, e posteriormente estes dados são validados para confirmar se o cartão é válido. Se for, é criado um *token*, e por fim esses *token* (com um tempo limitado de vida) são enviados para o *backend* e lá é feita a transação, o que fornece outro nível de segurança, pois a informação sensível não é transmitida entre aplicação e *backend*.

Uma das possíveis soluções a utilizar seria o *Android Pay* [33], que tem um bom suporte ao desenvolvimento e é bastante rápido e intuitivo no processo de pagamento. Contudo, este ainda não se encontra muito difundido pela Europa. Se este começar a ser mais utilizado, o *Stripe* também suporta integração com o *Android Pay*, o que irá facilitar no processo de inclusão.

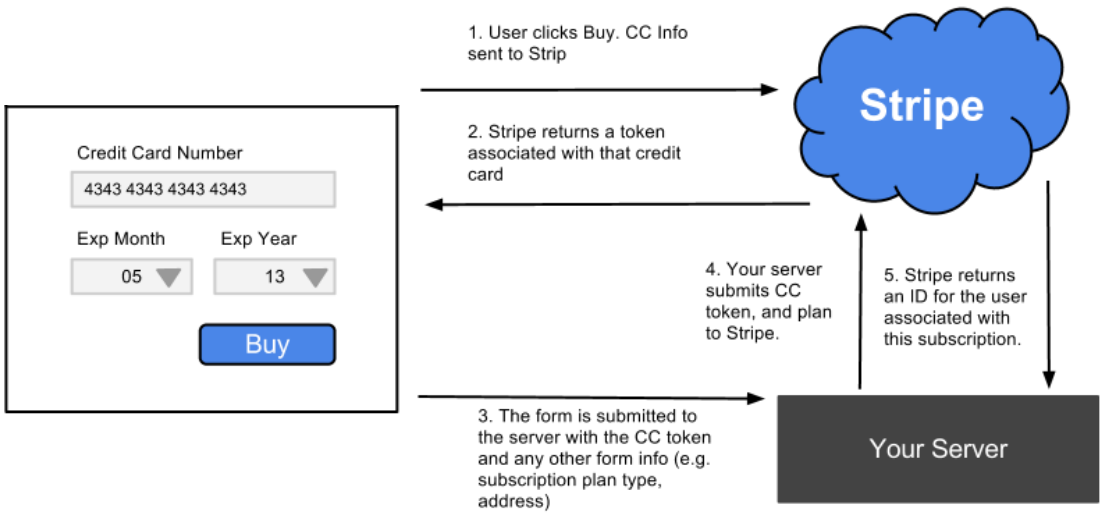



**Figura 11 - Modo de funcionamento de um pagamento feito através do Stripe** [103]

## 2.6 Indoor Maps

O conceito de conhecermos a nossa localização em tempo real não é novo. Com a explosão de uso dos *smartphones* por cada vez mais aplicações recorrem ao uso da localização do utilizador para dar mais valor e precisão às suas aplicações. Contudo, estes dados são obtidos através do GPS ou *wireless* incluídos no *smartphone* e ainda contêm algumas limitações, sobretudo quando falamos em espaços fechados.

É praticamente impossível nos dias de hoje conseguir obter um sinal fidedigno da nossa localização em espaços fechados através de GPS, e se falarmos em edifícios com múltiplos andares, torna-se mesmo impossível. No seguimento deste problema começaram a surgir soluções alternativas para ser possível que o utilizador saiba a sua localização atual num espaço fechado podendo, por exemplo, conhecer a sua localização relativamente ao produto que procura na loja.

Uma das soluções que surgiu está relacionada com o impulsionamento dos *beacons*. Os *beacons* são uma tecnologia bastante recente e em grande expansão, sendo dispositivos que se interligam com o *smartphone* e têm um determinado alcance, podendo desempenhar ações consoante uma pessoa entre ou saia desse alcance. Sabendo que um utilizador está num determinado raio, com vários *beacons* e uma certa calibração, é possível identificar a posição da pessoa num espaço fechado. É usando esta abordagem que uma das maiores empresas relacionadas com os *beacons*, a *Estimote* [24], desenvolveu um sistema de localização *indoor*. O problema com esta solução é que além do custo elevado que é necessário efetuar para os *beacons* (são sempre precisos no mínimo 4 *beacons* para este sistema, sendo que quantos mais *beacons* mais precisão teremos), também tem o problema de só ser possível saber a posição num espaço em relação aos *beacons* que estão lá colados e não a nossa posição associada a um mapa real.

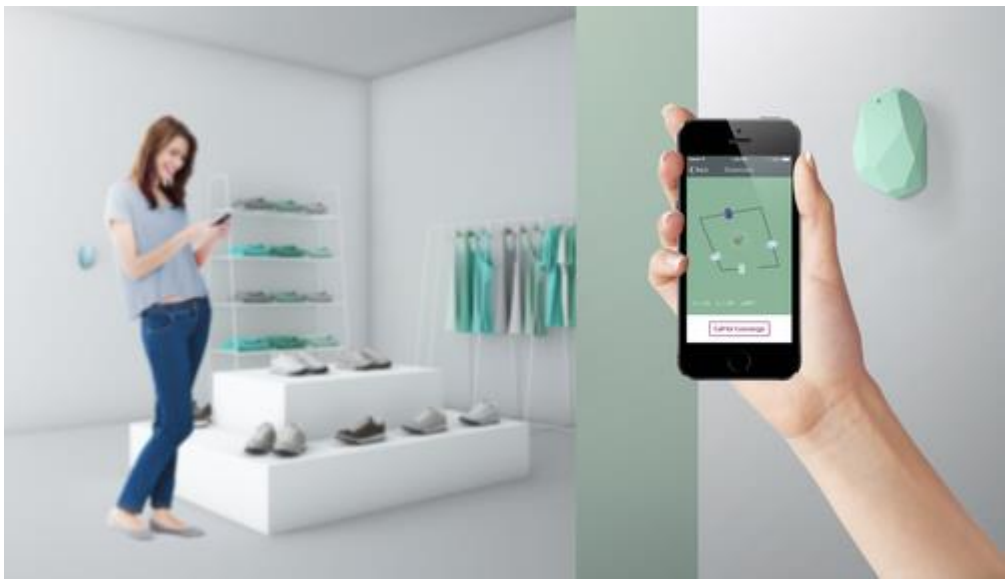


**Figura 12 - Indoor location Estimote** [101]

Para integrar esta funcionalidade na aplicação, iremos recorrer a bibliotecas que já existam e analisar qual se enquadra melhor na aplicação. Todas elas têm as suas vantagens e desvantagens, e é isso que vai ser retratado de seguida.

Inicialmente existe a biblioteca *RedPin* [36] que é uma biblioteca *Open Source* para localização indoor baseada em *fingerprint*. A *fingerprint* não é mais que criar uma identificação única, como que uma impressão digital, do edifício, neste caso através da força do sinal *wireless*. Esta biblioteca não fornece as coordenadas geográficas mas sim identificadores como o número da secção onde está ou o corredor. Ela consegue determinar a *fingerprint* através de diferentes alcances obtidos através de dispositivos wireless, ou seja, quantos mais dispositivos *wireless* existirem no espaço em que nos encontramos mais apurada será a nossa localização. Contudo, o principal problema desta solução é que a nossa localização não vai sendo atualizada à medida que nos vamos movendo, ou seja, ela consegue-nos fornecer sempre a nossa localização se formos fazendo atualizações, mas a nossa localização não é atualizada em tempo real.

Outro dos serviços que existe nesta área é o *Indoor Atlas* [37] que também é um sistema de *indoor map* que usa *magnetic fields* para determinar a posição atual do utilizador.

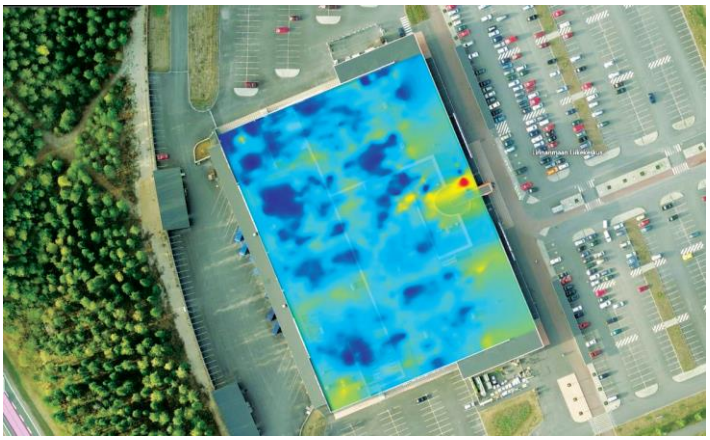


**Figura 13 - Exemplo de campo magnético de um edifício** [104]

Cada edifício tem um campo magnético distinto (causado pelos metais, cimento e outros materiais usados na sua construção) e é ai que está a modo de funcionamento desta solução. Enquanto que em outras soluções, como localização através da ligação *wifi* ou *beacons* estes materiais causam distúrbios e levam a uma menor precisão no resultado final, na localização através do campo magnético, quanto maior as perturbações melhor, pois criam uma pegada magnética única e distinta. A maioria das empresas com patentes neste campo da localização magnética afirma que esta consegue ter um erro na localização do utilizador de apenas 1 a 2 metros.

| Technology | Accuracy | Infrastructure | Set-Up /Costs | Maintenance | Power Source | Developer Environment |
|---|---|---|---|---|---|---|
| Magnetic Positioning | < 6 feet | Software/ Cloud | API / minimal | Crowdsource | None | iOS/Android |
| WiFi | 30 – 300 feet | Hardware/Soft-ware | Triangulation / $$$ | Remap - Fingerprinting | Electricity/ Battery | Android only |
| BLE | 6 – 100 feet (no blue dot) | Hardware/ Software | Configurable/ $$ | Device Management | Battery (Degrades) | iOS/Android |
| PDR (pedestrian dead reckoning) | N/A - Complementary | Software/Cloud | API / minimal | Crowdsource | None | iOS/Android |
| Other (cameras, LED, sound) | Variable | Hardware/ Software | Mesh coverage / $$$ | Device Man-agement | Varies | iOS/Android |

**Figura 14 - Comparação de tecnologias utilizadas para a localização indoor** [104]

Na imagem acima podemos observar uma comparação em termos de precisão, infraestrutura, custos, manutenção, entre outros campos, das várias tecnologias utilizadas neste momento para localização indoor. E como podemos analisar, a localização através de campo magnético é a que tem a melhor precisão e é das únicas que não precisa de hardware externo para funcionar na perfeição, aliado a uma integração relativamente fácil.

Esta solução começa por ser necessário de criar uma *venue*, ou seja, selecionar o edifício em que queremos pôr o mapa indoor e posteriormente fazer *upload* de uma imagem respetiva à planta do edifício. Depois de fazermos o *upload* da planta, segue-se a fase de calibração e de efetuar o *upload* dos dados para um serviço *cloud* da própria plataforma. Nesta fase, é necessário traçar todos os trajetos que são possíveis realizar dentro do edifício e efetuá-los usando o telemóvel nesse mesmo momento (nesta parte estamos a calibrar os dados relativamente ao campo magnético do edifício). No fim deste processo, somos capazes de ter acesso à nossa localização em tempo real dentro do edifício. A grande desvantagem desta solução é a calibração e carregamento dos dados, pois para edifícios muito grandes irá demorar algum tempo até este processo estar concluído. Contudo, é um processo que só é necessário efetuar uma vez. O *Indoor Atlas* fornece um SDK em que podemos criar gratuitamente um número ilimitado de *venues* e podemos ter até 1000 utilizadores a utilizar por mês.

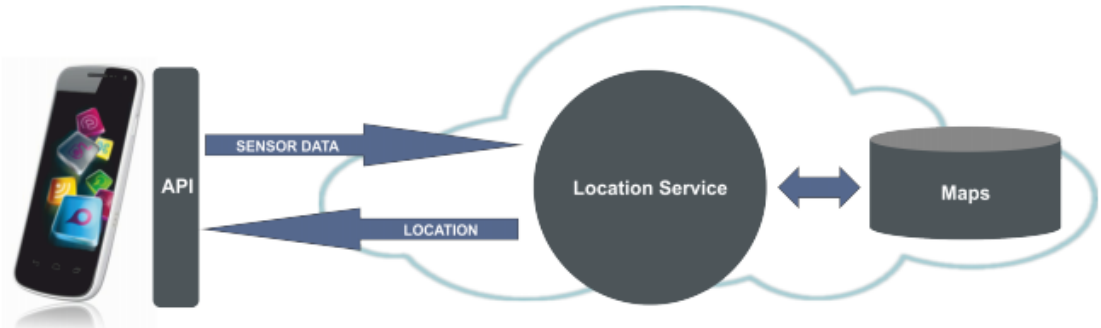



**Figura 15 - Serviço de localização Indoor Atlas** [105]

Outra solução possível é a *Aisle411* [4], já referida anteriormente. Além de ter a sua própria aplicação ainda fornece um SDK e API para ser possível desenvolver as suas próprias aplicações. Contudo, o SDK ainda só está disponível em versão beta e todas as funcionalidades (*List Search API, Maps SDK, Geo Context SDK*) só se encontram disponíveis para lojas que já se encontrem incluídas nos sistemas da Aisle411, o que é um obstáculo neste momento. Sendo possivelmente a ferramenta mais poderosa e mais utilizada no mercado, a não possibilidade de inserção de novas *venues* torna-se num entrave.

Por fim, temos a solução talvez mais bem estruturada em termos de design e com mais funcionalidades, a *Meridian Apps* [38]. Esta aplicação é, de todas as que referi, a que apresenta mais funcionalidades, incluindo direções, para sabermos onde virar numa localização indoor. Em termos de plataforma também oferece bastantes ferramentas como o *Meridian Editor*, *Mappping and self-guided wayfinding*, *Aruba Location Service* e *SDK*.

Através da *Meridian Platform* podemos também enviar *push notification* baseado na localização do consumidor em várias zonas da loja, associando determinadas campanhas e ações a certos *beacons* dispersos pela loja.


**Figura 16 - Aspeto do *indoor map* da Meridian Apps** [106]

No fim de analisadas estas 4 soluções, as que parecem dispor de todas as funcionalidades um design apelativo e não exigem um grande esforço financeiro, são a *Indoor Atlas* e a *Meridian Apps*. Contudo, até à fase de desenvolvimento alguma destas pode demonstrar algum impedimento, o que faz com que seja impossível a sua integração. Por isso, na fase inicial de desenvolvimento vão ser estas duas ferramentas que irão ser testadas e analisadas para, no final, se conseguir perceber qual funciona melhor no caso de uso a ser aplicado.

## 2.7 Redes Sociais

Hoje em dia são poucas as pessoas e as empresas que não se encontram presentes numa ou em várias redes socias. Estas tornaram-se parte do quotidiano da população e são usadas cada vez para mais funcionalidades. Esta é uma das melhores fontes de informação para conhecer padrões e rotinas do sítio onde a pessoa vai no seu dia-a-dia? As redes sociais, além de se tornarem indispensáveis para os seus utilizadores, são cada vez mais acedidas através dos *smartphones*, o que em conjunto com outros dados que conseguirmos retirar podem trazer muito valor no futuro, para reconhecimento de padrões.

Se as redes sociais já estão tão envolvidas no dia-a-dia das pessoas, no futuro irão tornar-se indispensáveis, e cada vez mais, vai ser possível saber informações relativa às pessoas. Atualmente as redes sociais mais utilizadas no mundo inteiro são o *Facebook* [39] e o *Twitter* [40], sendo que nos EUA a utilização diária do *Twitter* chega a ultrapassar a do *Facebook*. O tempo que as pessoas despendem a utilizar estas redes sociais, permite-lhes seguir e analisar as suas marcas e produtos favoritos. Esta situação traz um grande valor para a marca, que assim pode realizar estudos de mercado para analisar quais dos seus produtos têm uma melhor aceitação por parte dos clientes.

Segundo o estudo feito [41], descobriu-se que 3 em cada 5 consumidores interagem com as suas marcas favoritas nas redes sociais. O canal mais popular de comunicação é o *Facebook* com 55% de interação entre os consumidores e a marca, seguido pelo *Twitter* com uma percentagem de 21% e o Pinterest [42] com 10%. Segundo o mesmo estudo, os motivos pelos quais os utilizadores interagirem com as marcas nas redes socias são a receção de cupões ou descontos, de informações sobre o lançamento de novos produtos, de apoio ao cliente e ainda para compararem com o que outros consumidores estão a comprar e o que dizem acerca do produto.

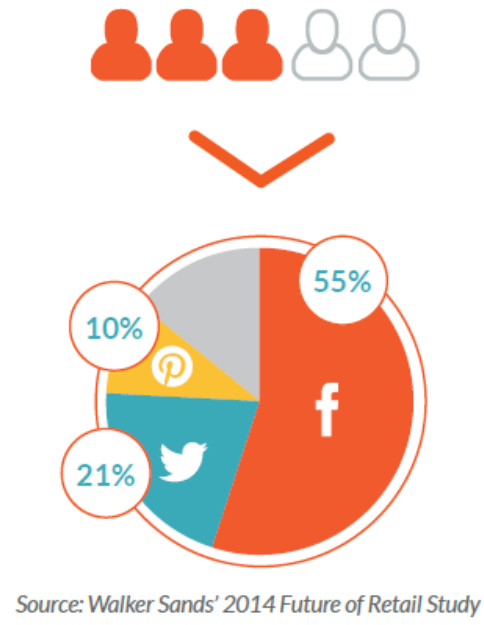



**Figura 17 - Interação entre o consumidor e as redes socias** [41]

## 2.8 Realidade Aumentada

A realidade aumentada é um tema que que tem sido falado sobretudo no mundo dos jogos. A possibilidade de interligar o mundo real com o mundo virtual é algo que capta o utilizador (principalmente por ainda criar o dito *wow factor*) e permite criar uma experiência totalmente nova relativamente à que está habituado. Com o aparecimento dos *smartphones* e o fato de todos eles virem equipados com câmara, também impulsionou esta tecnologia e permitiu que começasse a ser explorada por aplicações de jogos e outras.

A realidade aumentada é o conceito onde o mundo digital se mistura com o mundo real, a informação não segue apenas a pessoa mas também o seu olhar sobre o mundo. Ao olharmos para um objeto não conseguimos extrair toda a informação que este tem para oferecer, no entanto com a realidade aumentada pode ser possível retirar mais informação sobre objetos num mundo real. Embora seja uma tecnologia recente, tem apresentado um crescimento e avanço constante, mas ainda não se conseguiu afirmar no mercado. Contudo, o aparecimento dos *smartphones* com as suas câmaras integradas veio facilitar bastante o uso mais recorrente desta tecnologia, especialmente porque o requisito mínimo para o seu uso é uma câmara. Esta tecnologia também começou a entrar na área do retalho, onde, por exemplo, o futuro passa por o cliente poder experimentar uma peça de roupa virtual no seu corpo real para avaliar ver se gosta ou não dela.

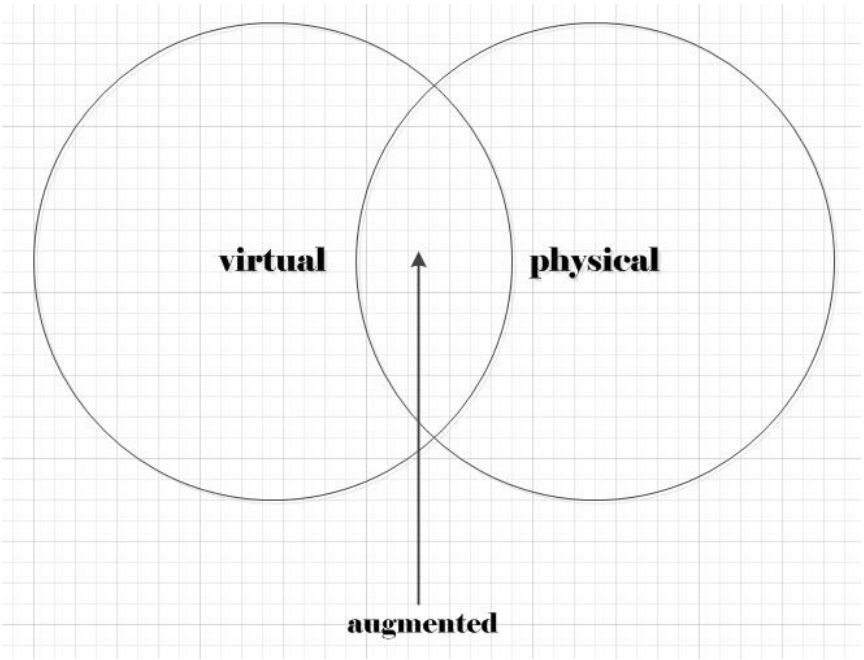



**Figura 18 - Enquadramento da realidade aumentada** [107]

A realidade aumentada pode ser categorizada em duas formas distintas: *location-based* e *vision-based*. A primeira vai apresentando conteúdo digital no mundo real à medida que o utilizador vai movendo o telemóvel ou a si próprio. O conteúdo digital que vai sendo mostrado encontra-se distribuído através das suas coordenadas geográficas. Já o *vision-based* corresponde a conteúdo digital que aparece quando o utilizador aponta a câmara do seu telemóvel para um determinado objeto ou target (*QRCode*, 2D *target*). [43]


**Ilustração 1 - Duas categorias da realidade aumentada: *location-based* e *vision-based*** [108][109]

No enquadramento desta tese, a realidade aumenta não vai traduzir resultados que futuramente possamos analisar, mas irá criar uma experiência totalmente nova para o utilizador, por exemplo, a possibilidade de ver os produtos em promoção distribuídos pela loja, conseguir ver imagens ou vídeos relacionados com o produto que pretende comprar simplesmente apontando a câmara para o próprio produto. É uma experiência que não é usual e comum de se encontrar em Portugal, mas nos EUA já começa a ser bastante usada e o seu uso deve-se principalmente à *Aisle411* [44][45].

A *Aisle411* é uma aplicação móvel que tem como objetivo criar a experiência de compra mais eficiente. A finalidade é o cliente chegar à loja com a sua lista de compras e, através da aplicação, que faz uso da realidade aumentada e da localização dentro da loja, é mostrado o caminho a seguir para o cliente conseguir comprar todos os produtos que procura. Pode-se ver o sucesso que esta aplicação está a ter nos EUA graças às parcerias que estabeleceram com a *Target* [21] e a *Walgreens* [6], duas das maiores cadeias de retalho dos EUA.

Como já foi referido, a realidade aumentada não irá auxiliar o retalhista com dados para posterior análise, mas pode influenciar o consumidor na altura de comprar um produto. Foi efetuada uma análise onde foi questionado ao consumidor em quais dos cenários a realidade aumentada poderia influenciar o seu impulso para efetuar uma compra, sendo que os resultados podem ser observados na imagem seguinte. [46]


**Figura 19 - Resultados de casos de uso da realidade aumentada** [46]

Em termos de soluções tecnológicas disponíveis para que seja possível implementar a realidade aumentada, existe alguma limitação, principalmente se estivermos à procura de uma biblioteca ou solução *Open Source*. Como foi referido, existem duas funcionalidades de maior associadas à realidade aumentada, que são o reconhecimento de objetos, imagens ou algum padrão (*vision-based*) e a geolocalização (*location-based*). Iremos tentar integrar estas duas funcionalidades, a primeira para obtermos mais informação acerca de um determinado produto e outra para conseguirmos ver em tempo real onde se encontram os produtos e a que distância.

Existem bastantes aplicações móveis no mercado que utilizam a realidade aumentada, mas possivelmente nenhuma é tão completa nem tem tantas funcionalidades como a Layar [47]. Esta permite as duas soluções, tanto *location-based* como *vision-based*, além de várias outras características. Em termos de estabilidade e precisão é das aplicações que apresenta melhores resultados. Também disponibiliza um SDK, mas como não existe nenhum plano sem custos esta solução não foi tida em conta na fase da implementação.

Uma das soluções mais conhecidas e fiáveis usadas para incorporar a realidade aumentada em dispositivos *Android* é a *Vuforia* [48]. Contudo, tem um plano com algumas limitações em termos de utilização, para quem quiser utilizar a versão de testes sem custos. Contém um SDK de desenvolvimento para *Android* e vem ainda com algumas amostras para que seja fácil a sua integração. Mas um dos maiores problemas desta solução é não ter a integração de geolocalização, ou seja, se quisermos mostrar pontos de interesse através de realidade aumentada esta solução não tem nenhuma maneira de o fazer, e essa vai ser uma funcionalidade chave na nossa aplicação.

Outra das aplicações mais utilizada nesta área é a *Wikitude* [49], que disponibiliza um SDK tanto para *Android* como para iOS e é bastante utilizada em todo o mundo (existente em mais de 10000 aplicações atualmente). Contudo, o maior problema relacionado com esta solução é que a versão sem custo vem com *watermarks* que preenchem o ecrã todo, o que fica muito pouco agradável para o consumidor final observar. Para remover essas *watermarks,* temos de adquirir uma licença, rondando a mais barata os 500€.

Por fim, uma das últimas soluções é uma *framework* chamada *Beyondar* [50], uma biblioteca de realidade aumentada, resultante do desenvolvimento de um jogo que utiliza realidade aumentada. A principal vantagem desta *framework* é ser *open source,* contudo, tem algumas desvantagens como é o facto de ser limitada em termos de funcionalidades e de não permitir fazer *image recognition (vision-based)*, uma das *features* que seria incluída no protótipo desenvolvido no âmbito desta dissertação.

## 2.9 Funf Framework

Funf [51] é uma biblioteca Open Source que nos ajuda a tirar partido dos sensores que temos disponíveis hoje em dia nos nossos *smartphones*. A quantidade, qualidade e precisão destes sensores varia entre marcas e modelos, contudo a sua recolha e análise no futuro ajudarão a identificar certos padrões.

Entre os sensores e dados que o Funf nos ajuda a recolher estão:

- GPS
- Localização
- WLAN
- Acelerómetro
- Bluetooth
- Antena ID
- Log de chamadas
- Log de SMS
- Histórico do Browser
- Contactos
- Aplicações a correr
- Aplicações instaladas
- Estado do ecrã (ligado/desligado)
- Estado da bateria

Tendo em conta estes dados, temos de analisar quais os mais relevantes e os que fazem mais sentido dentro do contexto. Contudo também temos de ter em conta que alguma desta informação pode violar a privacidade do consumidor, por isso, a privacidade também é uma variável a ter em conta quando escolhermos quais os sensores/dados que queremos recolher.


**Figura 20 - Esquema do funcionamento da *framework* Funf** [51]

# 3 Seleção de Tecnologias

Em termos de enquadramento geral da aplicação, esta vai comunicar e interagir com múltiplos dispositivos e APIs. Todos estes irão permitir criar uma experiência de compra diferente da que o utilizador está habituado (realidade aumentada, indoor maps, etc), enquanto outras vão ser utilizadas para enviar informações que no futuro serão utilizadas para reconhecer padrões de compra por parte dos consumidores. Neste capítulo irei tentar detalhar tecnicamente como funcionará ou comunicará cada um destes componentes com a aplicação *Android*.

Começando pela linguagem, a aplicação vai ser desenvolvida usando o *Android SDK* [52] disponibilizado pela *Google* [53] e usando como IDE o *Android Studio* [54] que na sua instalação já trás o suporte ao SDK. Este disponibiliza emulador para ser possível testar a aplicação em vários *smartphones* distintos, suporte de previsualização de vários componentes (*smartphones*, *tablets*, *smartwatches*) e sobretudo já vem com integração com o *Gradle* [55] o que facilita bastante na parte da compilação e integração de bibliotecas ao projeto. Seguidamente a parte de autenticação, o utilizador só poderá efetuar o seu login através duma rede social, o *Facebook* ou o *Twitter*. Para *Facebook* será utilizado o *Facebook* SDK [56] que é a biblioteca disponível para *Android*, que fornecerá toda a parte de autenticação, como também o acesso a informações específicas do utilizador. Já no *Twitter* será usado o *Twitter* Kit SDK [57], onde também teremos toda a parte de autenticação e acesso a informações dos utilizadores, embora neste caso mais limitada já que no *Twitter* não conseguimos obter tantos dados relativos ao utilizador como no *Facebook*. Será também integrada a *OpenWeatherMap* API [58] que nos irá fornecer informação das condições climatéricas em vários pontos ao longo da aplicação, para no futuro se conseguir analisar se este aspeto tem alguma influência nos consumidores se deslocarem à loja. Também teremos a nossa própria API, que irá comunicar com uma base de dados para podermos armazenar todos os dados que formos recolhendo à medida que o utilizador vai utilizando a aplicação.

Os *beacons* e o *geofencing* funcionarão para despoletar ações na aplicação, sobretudo notificações, mas enquanto que o *geofencing* é programado com a ajuda do *Google Play Services* [59] e faz uso da localização do utilizador, os *beacons* são mecanismos físicos que terão de ser calibrados e inseridos em pontos chave na loja, além de necessitarem também de ser programados. Para os beacons, dependendo dos que forem adquiridos, podem vir ou não com um SDK que facilita a parte de comunicação entre a aplicação e o dispositivo. Já o NFC também já vem integrado no próprio *smartphone*, contudo as Tags que vão ser lidas também são dispositivos físicos e também têm de ser configuradas para ser possível produzir o resultado esperado. E vão existir tags para propósitos diferentes como check-in, check-out, scan de produtos e ainda pagamentos, por isso cada uma vai ter de ter uma configuração diferente das outras.

Quanto à parte da realidade aumentada, será necessário integrar um SDK ou uma *framework* que já venha com algumas funcionalidades, tanto na parte de geolocalização, como no reconhecimento de imagens. Algumas das bibliotecas de realidade aumentada também incluem módulos feitos em C e C++, por isso também temos de analisar a necessidade e vantagens de integrar o NDK [60] na aplicação.

Relativamente à parte de recolha de informação por parte dos sensores do dispositivo, vai-se tentar integrar a biblioteca *FUNF* que já foi falada acima, contudo, se esta não traduzir os resultados que são esperados, vai ser necessário implementar uma forma de recolher os dados de múltiplos sensores em situações específicas. Isto tudo vai ter de ser feito através do *SensorManager* [61]**,** que é a classe *Android* que controla todos os sensores do *smartphone*.

Na parte relativa à localização *indoor* segue-se o mesmo caso da realidade aumentada, sendo que ainda temos várias soluções para experimentar contudo, tanto a *Meridian Apps* [38] como a Indoor Atlas [37] contêm um SDK que facilita o processo de desenvolvimento e de própria comunicação com a *Cloud* deles. Já a *RedPin* [36] só apresenta uma pequena biblioteca e alguma documentação de ajuda à integração com a aplicação. Por fim, a parte dos pagamentos através de *Bitcoins* [30] também será integrada uma biblioteca para ajudar no processo de transação do montante. De notar também que serão usadas outras bibliotecas para ajudar na parte do design e *user experience* e todas estas geridas através do *Gradle* [55]**,** o gestor de dependências do *Android*. As que não serão geridas do *Gradle* serão somente bibliotecas *open source* às quais foi necessário de efetuar modificações para que estas façam os resultados esperados.

Em termos da aplicação o funcionamento geral vai ser o que foi detalhado em cima, contudo também vai ser necessário desenvolver a API que irá armazenar todos os dados que vão ser enviados através da aplicação. Este serviço vai ser feito usando a linguagem *Python* [62] e com recurso à *framework Django* [63] e à sua extensão Django *Rest Framework* [64], o motivo é o seu bom suporte, modularidade, uma comunidade ativa, uma linguagem simples e ainda possibilidade de integração com vários tipos de base de dados como *PostGIS* [65], SQLite [66] entre outras. Através do uso desta tecnologia vai ser possível disponibilizar um *backoffice* onde é possível visualizar e gerir toda a informação e ainda foi utilizador a *framework* Swagger [67] de modo a podemos testar e visualizar mais facilmente documentação associada à API. Antes de começar a detalhar a nível de arquitetura o que foi feito, vou começar por enumerar bibliotecas e os use case que usei para chegar à arquitetura final da aplicação.

## 3.1 Backend

Um dos objetivos desta tese é desenvolver um *backend* onde fosse possível armazenar toda a informação que vai sendo recolhida e tratar esses dados, de forma, que num futuro esses dados possam ser analisados e deles conseguirmos recolher métricas de relevo para um retalhista. Foi assim construído o *backend* usando o Django [63] e o Django Rest Framework [64] para ser possível disponibilizar uma API na forma de um serviço RESTful. Foi ainda integrada a *framework* Swagger [67] para podermos ter acesso à documentação da API duma forma fácil e intuitiva. Todas as respostas aos pedidos e envio de informação para a API são feitas através de JSON. O *backend* contém ainda o *backoffice* que permite visualizar e modificar toda a informação desde que tenha credenciais de acesso ao mesmo, este *backoffice* foi construído o Django Admin. A API foi construída seguindo a metodologia CRUD (create, read, update, delete), sendo possível realizar todas essas operações em todos os modelos de dados existentes no *backend*.

O modelo de dados definido no *backend* suporta as informações nucleares da aplicação em si, recebemos informações sobre os produtos duma determinada loja e guardamos o resto da informação relativa ao registo de utilizadores, sensores, produtos vistos e notificações recebidas.

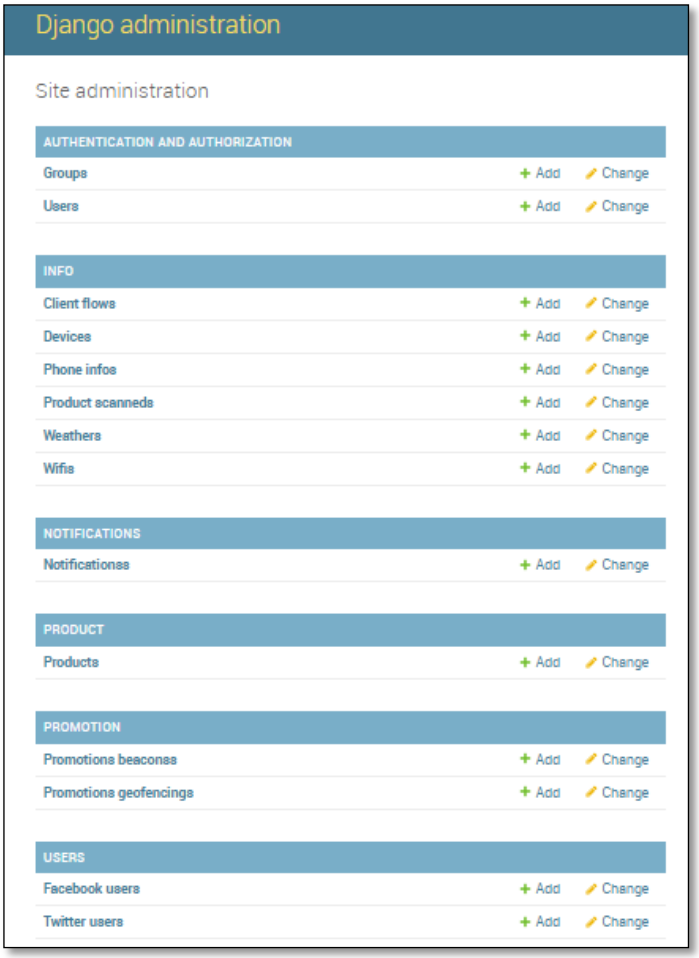



**Figura 21 - Django Admin do *backend* desenvolvido**

## 3.2 Dagger 2

Umas das bibliotecas usadas para ter uma aplicação mais estruturada foi o Dagger 2 [68], construído em cima do Dagger 1 e desenvolvida pela Java Core Team da Google.

O Dagger 2 não é nada mais que uma biblioteca de injeção de dependências. Esta metodologia não é nova, já existiram outras bibliotecas a tentar fazer o mesmo que o Dagger 2 (Spring, Google Guice), contudo muitas delas continham grandes problemas ao nível de configuração e em nível de performance, pois utilizam *reflection* que é bastante dispendioso ao nível de performance, já com o Dagger 2 conseguimos ganhar cerca de 13% de performance ao nível do processamento. Com o Dagger 2 temos as classes que precisamos, quando precisamos, pois o próprio *life-cycle* da aplicação sabe quando nos fornecer as mesmas. Podemos separar o Dagger 2 em 4 partes:

- **@Modules / @Provides** – Fornece as dependências
- **@Inject** – Injeta ou faz um pedido de injeção duma dependência
- **@Components** - é a ponte entre os @Modules e @Injects
- **@Scope** – Fornece informação de quando uma classe vai ser instanciada ao longo da aplicação (Exemplo: ***@Singleton*** – instanciada uma vez na aplicação, ***@PerActivity*** – instanciada cada vez que a Activity é iniciada.)

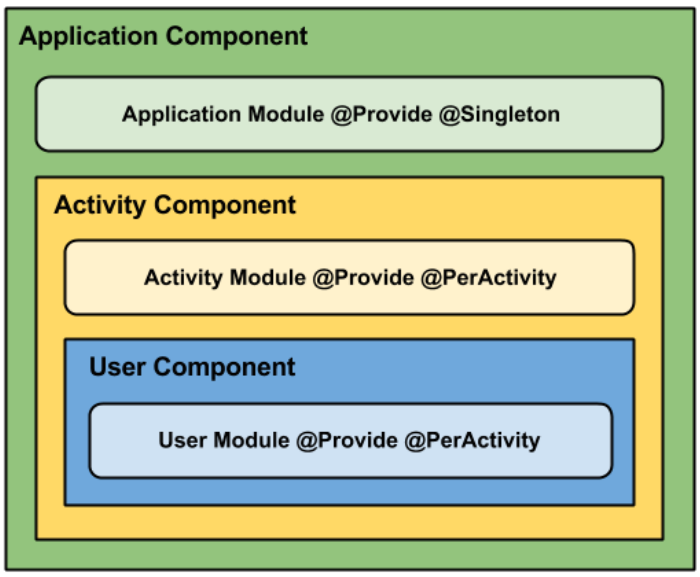



**Figura 22 - Workflow do Dagger 2** [110]

## 3.3 Data Binding

O Data Binding [69] apareceu com o lançamento do Android 6.0 (Marshmallow), e é uma biblioteca que permite construir duma forma mais eficiente os *layouts* das aplicações, e faz isso basicamente introduzindo a informação diretamente no layout. Antigamente, e ainda hoje se não usarmos uma biblioteca de injeção de views, grande parte do código que pomos numa Activity ou Fragment é basicamente para interligar os elementos das views à informação que é necessário mostrar nos mesmos, o Data Binding veio mudar isso deixando fazer associações (*bind*) de variáveis ou objetos diretamente nas views, e quando essas variáveis ou objetos sofrerem alterações, as views são atualizadas automaticamente.

Para ser possível utilizar o Data Binding é necessário de criar também um View Model que vai estar associado a uma Activity ou Fragment e é o que vai fazer a ponte para interligar as variáveis ou objetos com as views do layout.

Com a introdução da biblioteca de Data Binding por parte do Google é agora mais fácil implementar um padrão, o mesmo que foi usado na aplicação, que é o MVVM, que significa Model-View-View Model. Este padrão é composto por 3 componentes:

- **View** – define somente a estrutura layout
- **View Model** – define a ponte entre a view e o model, e lida com a lógica relacionada com a view
- **Model** – contem a informação e toda a lógica da mesma

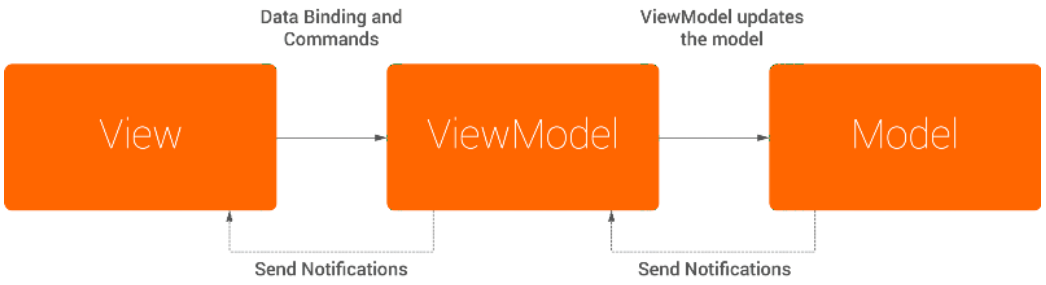



**Figura 23 - Modelo MVVM** [111]

## 3.4 Realm

O Realm [70] é uma base de dados mobile e é base de dados utilizada para fazer a persistência local de informação na aplicação. Por norma no desenvolvimento *Android* usa-se o *SQLite* [66] para fazer a persistência local, contudo é preciso bastante código para desenvolver, é preciso usar *strings hardcoded* para criação de tabelas e *queries* e não existe nenhuma forma simples de fazer uma simples *querie* e retornar o objeto que queremos.

O Realm veio resolver estes problemas, e afirma-se mesmo como o substituto do *SQLite* em Android, foi desenvolvido em C/C++, e reside aí grande parte do seu segredo ao nível da performance. Para persistir um objeto, este só tem de estender a *class RealmObject* e a partir desse momento temos uma tabela criada onde podemos fazer todo o tipo de inserções, *queries*, ordenações e eliminações, pois este já vem com grande parte destes métodos implementados.

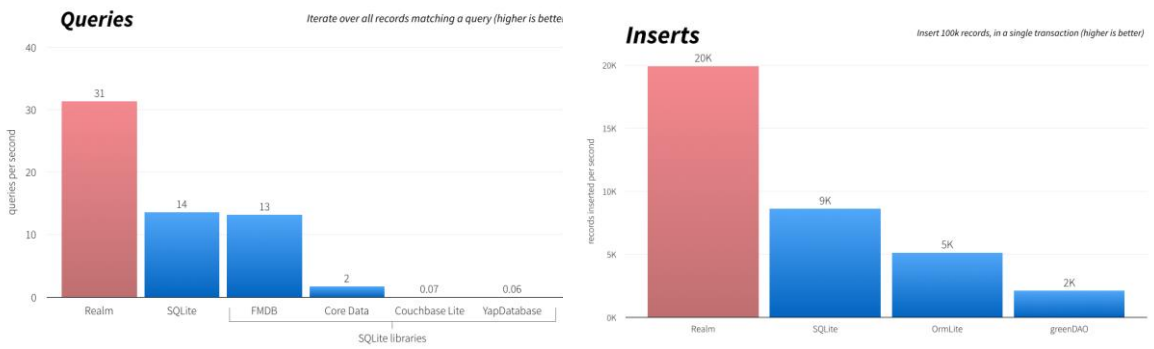



**Figura 24 - Comparação de performance entre o Realm, SQLite e outras bases de dados mobile** [70]

O Realm tem um conjunto de funcionalidade que são bastante úteis no desenvolvimento mobile, que o SQLite não permite, como o caso de podermos encriptar a nossa base de dados, para o caso desta conter informação sensível (como informações de cartões de crédito). Contudo o Realm ainda tem algumas falhas tais como, não suportar o auto incremento e só permitir os métodos *default* (*getters* e *setters*) dentro da classe do objeto a persistir.

Sendo uma solução muito mais versátil, eficiente e segura que o SQLite fez todo o sentido incluir a mesma neste projeto. Apesar de todas as limitações enumeradas acima é uma solução/tecnologia bastante recente e que está em constante melhoramento pelos seus criadores, o que pode prever que num curto espaço de tempo todas estas falhas que foram ditas sejam colmatadas.

## 3.5 Retrofit2 + OkHTTP

O Retrofit2 [71] é uma biblioteca HTTP Cliente para Android, e a sua grande vantagem em relação a outras é a sua simplicidade e performance. Com a extinção da biblioteca Apache HTTP API do SDK nativo do Android, o Retrofit2 mostra-se como a melhor solução neste campo. O Retrofit1 já era das bibliotecas mais usadas neste campo, contudo esta nova versão sofreu bastantes melhoramentos, sobretudo ao nível de como são feitos os callbacks aos pedidos feitos.

A definição dos pedidos a fazer (POST, GET, PUT, DELTE), bem como as Headers são definidas através de anotações o que simplifica bastante a compreensão e redução ao nível de código. Ao nível dos pedidos suporta tanto pedidos síncronos como pedidos assíncronos, e a sua resposta vem sempre na forma dum objeto, ou seja, ao fazermos um pedido vamos obter como resposta o objeto automaticamente criado (possível graças a um *GsonConverterFactory*).

```
@POST("/api/users/register_facebook/")
Call<FacebookU> postFacebookUser(@Body FacebookU facebookUser);
```

**Figura 25 - Exemplo de POST no Retrofit**

O OkHTTP [72] é outra biblioteca que é usada pelo Retrofit2 como HTTP interface e permite fazer os pedidos. Como podemos ver nas imagens abaixo criamos o *OkHTTPClient* com as *headers default* dos pedidos e adicionamos um *interceptor* para irmos recebendo os *log's* de todos os pedidos e repostas ao *endpoint*. Na outra imagem criação o cliente *Retrofit* onde associamos o *OkHTTPClient* criado em cima, com o URL do *endpoint* e ainda o conversor que queremos para transformar as respostas em JSON em objeto (no caso o *Gson*).

```
OkHttpClient.Builder httpClient = new OkHttpClient.Builder();
HttpLoggingInterceptor interceptor = new HttpLoggingInterceptor();
interceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
httpClient.addInterceptor(interceptor);
httpClient.addInterceptor(new Interceptor() {
    @Override
    public Response intercept(Chain chain) throws IOException {
        Request original = chain.request();
        Request request = original.newBuilder()
                .header("Accept", "application/json")
                .header("Content-Type", "application/json")
                .method(original.method(), original.body())
                .build();
        return chain.proceed(request);
    }
});
return httpClient.build();
```

**Figura 26 - Criação do OkHTTPClient**

```
new Retrofit.Builder()
 .baseUrl(this.API_URL)
 .client(client)
 .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
 .addConverterFactory(GsonConverterFactory.create(gson))
 .build();
```

**Figura 27 - Criação do cliente Retrofit**

# 4 Solução Proposta

O principal objetivo desta tese era criar uma aplicação base, que no futuro pode ser utilizada para construir uma aplicação em cima desta e já ter ao seu dispor um conjunto de funcionalidades. Neste tópico vai-se enumerar as operações que o utilizador da aplicação e o dono da loja poderão fazer e ainda como foi estruturada a arquitetura da aplicação e do *backend*, e como estes vão interagir entre si.

## 4.1 User Stories

De forma a dar uma melhor interpretação a todas as ações que o utilizador da aplicação e o dono da loja podem efetuar ao usar a aplicação e no backend, foram criadas as *user stories*.

**US1 .** Como utilizador, posso efetuar o meu login/registo através de redes sociais.
**US2 .** Como utilizador, posso ver os produtos disponíveis na loja para poder fazer uma lista de compras dos mesmos.
**US3 .** Como utilizador, posso registar a minha entrada na loja através de NFC ou QRCode.
**US4 .** Como utilizador, posso registar a minha saída na loja através de NFC ou QRCode.
**US5 .** Como utilizador, posso ver os dados relativos ao meu perfil
**US6 .** Como utilizador, posso pagar as minhas compras através de pagamentos mobile.
**US7 .** Como utilizador, posso saber a minha localização atual dentro da loja através dum mapa.
**US8 .** Como utilizador, posso saber a localização de um determinado produto através de um mapa.
**US9 .** Como utilizador, posso ver os produtos na loja através de realidade aumentada.
**US10 .** Como utilizador, posso ver características (vídeos, imagens, etc.) sobre um determinado produto através de realidade aumentada.
**US11 .** Como utilizador, posso ver detalhes sobre um determinado produto através de NFC ou QRCode.
**US12 .** Como utilizador, posso receber promoções exclusivas à medida que ando pela loja (*beacons*).
**US13 .** Como utilizador, posso receber promoções exclusivas quando me encontro perto de uma loja (*geofencing*).
**US14 .** Como utilizador, posso partilhar informações dos produtos nas redes sociais.
**US15 .** Como dono da loja, devo conseguir ver informação relevante associada aos meus consumidores.
**US16 .** Como dono da loja, devo conseguir ver todos os dados recolhidos na minha loja através duma *dashboard*.

## 4.2 Arquitetura

A definição da arquitetura é um dos pontos fulcrais de qualquer projeto desta envergadura, a forma como os serviços comunicam entre si, a troca e armazenamento de informação, a organização e acessos a classes entre muitos outros pormenores são a base que tem de ser pensada antes de qualquer outro avanço, pois, uma arquitetura mal detalhada é sinónimo de problemas futuros.

Nesta secção será apresentada a arquitetura geral da plataforma como um todo, isto é, todos os serviços, SDK's e bibliotecas que precisam de estar conectados para a aplicação funcionar. Vai ainda ser detalhado com maior pormenor como se encontra organizada estruturalmente a aplicação e como esta comunica com todos os serviços, API's e *hardware* externos.

Vai ser dado mais relevo à organização estrutural feita dentro do código da aplicação, que levou à criação dos chamados Managers que vão ser de seguida detalhados mais a pormenor. Na imagem abaixo é possível ver um *overview* geral da aplicação e todos os serviços que esta vai necessitar e como estes vão comunicar entre si.

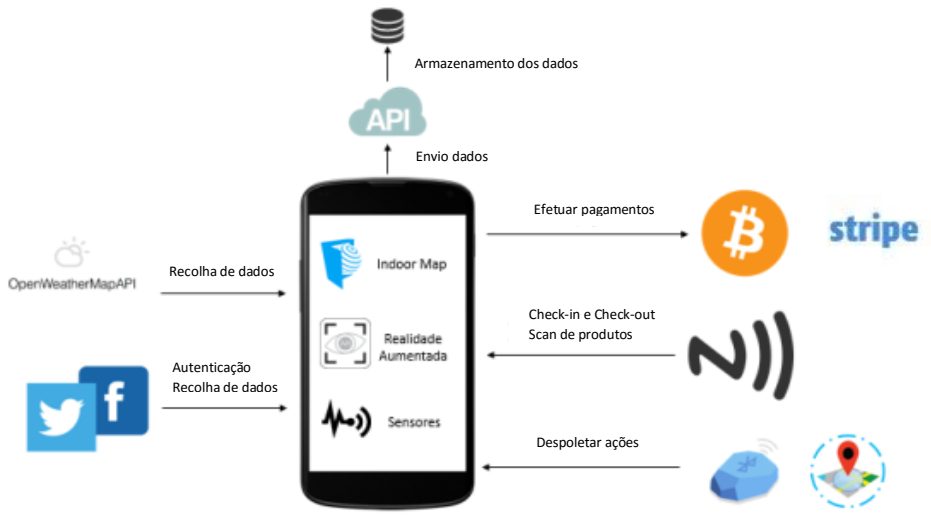



**Figura 28 - Visão geral**

Falando agora da arquitetura geral de como a aplicação está organizada, esta pode-se separar simplesmente em 3 blocos:

- Em primeiro temos a View (Fragment/Activity) que tem sempre a si associado um View Model. O View Model é a classe que vai conter as variáveis ou objetos que vão estar associadas diretamente à View, para assim quando estes valores forem alterados a View atualizar.
- Em segundo lugar temos os Managers e Providers da nossa aplicação, onde o Manager é responsável por fazer o processamento e tratamento dos dados enquanto o Provider é responsável pela receção e envio da informação, quer externamente (API) como armazenamento local (Realm).
- E por fim temos a API que é responsável por receber, processar, armazenar e enviar informação.

Com esta estrutura, é possível dizer que se segue um padrão MVVM (Model View ViewModel), este padrão é mais fácil de implementar devido ao lançamento em 2015 da biblioteca do Data Binding para Android. Através deste modelo torna-se muito mais fácil efetuar também testes unitários. Assim temos uma aplicação mais organizada e cada componente da aplicação fica somente responsável por uma parte: os Fragments e Activities ficam somente responsáveis por mostrar informação, não contêm nenhuma lógica, o View Model por ir buscar os dados e detetar alterações nos mesmos, os Managers pelo processamento da informação e os Providers pela recolha e armazenamento da informação.

A criação dos Managers e Providers foi pensada para que houvesse uma classe responsável simplesmente pela comunicação de informação com a API e outra que fizesse o processamento dessa mesma informação e a envia-se para ser mostrada ao utilizador. Assim é possível criar uma camada de abstração entre a recolha/envio de informação e o tratamento e amostragem da mesma.

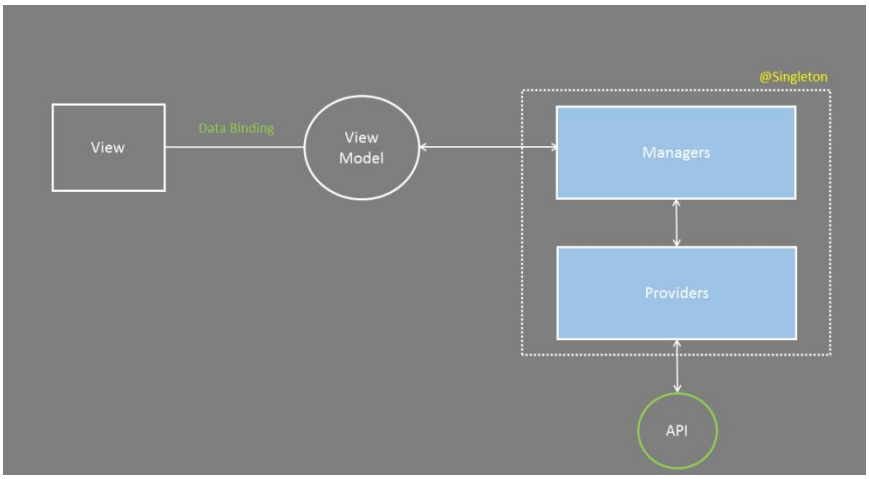



**Figura 29 - Arquitetura geral da aplicação**

Como já foi falado acima, com o Dagger2 consegue-se ter um maior controlo sobre a gestão das classes ao longo da aplicação, e assim sendo na aplicação temos somente um ApplicationComponent e um ApplicationModule. Dentro deles temos somente classes que vão ser *singletons* ao longo do *life-cycle* da aplicação, como podemos ver na imagem abaixo as classes estão separadas em dois blocos, um com as classes inerentes da aplicação e bibliotecas externas e no outro as classes criadas para o controlo da aplicação. Nas bibliotecas externas temos o OkHTTPClient, Retrofit e o Gson [73] que já foram faladas mais acima além de referências ainda ao Context [74] e Application.

Já do lado das classes próprias da aplicação temos os vários Managers criados para ajudar no controlo e processamento dos dados dentro da aplicação, cada um destes Managers recebe como argumento uma referência para o Context e para o Provider a si associado.

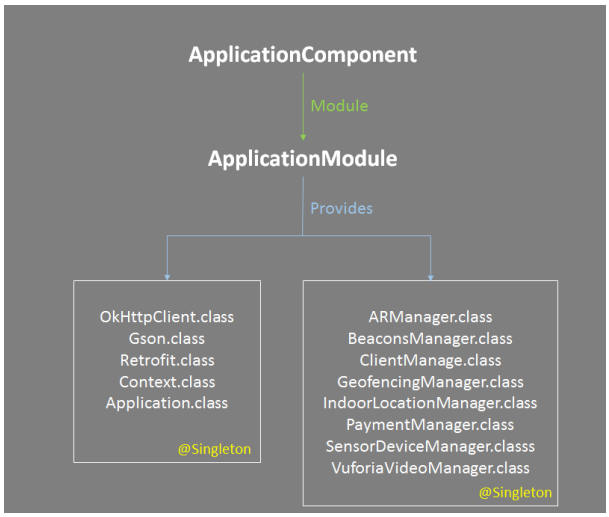



**Figura 30 - Estrutura do Dagger 2 na aplicação**

Detalhando agora mais a pormenor, na aplicação existem 8 Manager e o igual número de Providers, passando agora a aprofundar o principal papel de cada um:

- **ARManager** – é responsável por fazer o processamento e inicialização das funções necessárias ao funcionamento da biblioteca BeyondAR. Como já foi referido acima BeyondAR é uma biblioteca open source, a qual permite fazer realidade aumentada através de geolocalização. Neste manager é criada uma instância dessa biblioteca e expomos vários métodos para ser possível alterar os parâmetros em tempo real. A biblioteca foi incluída como um projeto e não através do Gradle pois ainda foi realizadas algumas alterações no core da biblioteca original que continha alguns bugs que tiveram de ser resolvidos antes de ser possível a sua utilização.

    - **ARProvider** – Neste momento o este provider não contém nada mais que o construtor pois para a realidade aumentada não está a ser recolhida nem recebida nenhuma informação específica, esta funcionalidade serve para criar uma nova experiência de compra para o cliente.

- **BeaconsManager** – é responsável por criar todos os métodos para inicialização de monitorização e variação do beacons. É a única classe que comunica com o SDK da Estimote e suporta a criação e inicialização de todos os componentes relacionados com os beacons.

    - **BeaconsProvider** – o provider é responsável por comunicar com a API da Estimote para ir buscar valores relativos a cada beacons (número de visitantes, Nº de visitantes únicos), apesar desta funcionalidade ainda não estar disponível no SDK da Estimote para android estas funcionalidades já foram incluídas, pois no futuro vão ser métricas valiosas para análise. Além do enumerado anteriormente o provider também comunica com a API da aplicação para ir buscar as promoções associadas aos beacons.

- **ClientManager** – este é o manager responsável por processar toda a informação relativa ao utilizador, seja ao nível da sua informação pessoal como ações que este faça. É o responsável por interpretar e validar a leitura duma TAG NFC, inicialização do pedido para a OpenWeather API e finalmente é o responsável por fazer a ponte entre os pedidos requeridos e a resposta dos mesmos por parte do provider.

    - **ClientProvider** – este é provavelmente o provider que tem mais processamento ao nível do envio e receção de informação. É o responsável por salvar os dados do utilizador relativos ao login no *backend*, ir buscar informação associada a um utilizador através do seu ID, salvar o tempo obtido através da chamada à OpenWeather API feita no manager, ir buscar a lista de produtos a ser usada na aplicação e salvar esses produtos localmente (através do Realm), ir armazenando e por fim salvar o ClientFlow (hora e data de entrada e saída da loja, dados relativos aos sensores, etc…) no *backend*, salvar todas as notificações recebidas relativamente a promoções quer no *backend* como localmente e alterar o seu estado quando sofre alguma alteração, salvar informação relativamente ao *smartphone* (hardware, software e rede) que o utilizador está a usar e por fim salvar todos os produtos que o utilizador fez scan.

      De notar que em todas as operações de envio e receção de informação foram criados callbacks para termos a certeza que os dados estão a ser salvos com sucesso ou se alguma operação não correu como esperado.

- **GeofencingManager** – este manager é o responsável por inicializar todas as funcionalidades necessárias para a utilização de geofencing. O geofencing é uma funcionalidade fornecida pela Google, por isso o primeiro passo de todos é conectar a aplicação com Google API Client, de seguida recebemos as geofences que queremos criar através do *backend* e registamos essas geofences aos Location Services da Google.

    - **GeofencingProvider** – responsável por recolher todas as promoções associadas ao geofencing que se encontram no *backend*.

- **IndoorLocationManager** – é o responsável por aceder ao servidor do IndoorAltas para se obter a planta correspondente ao edifício, posteriormente essa planta é armazenada localmente. Além disso inicializa-se também o LocationManager e FloorPlanManager do SDK do Indoor Atlas para se receber atualizações relativas à localização indoor.

    - **IndoorLocationProvider** – este é o provider que futuramente vai comunicar com a API da Indoor Atlas para ser possível ir buscar informação relativa aos caminhos mais comuns, *heatmaps* dos edifícios e outras informações relevante. Contudo essa API ainda não se encontra disponível para o consumidor final.

- **PaymentManager** – este manager é o responsável por tudo o que é relacionado com a parte dos pagamentos. Contém funções de conversão entre Euros e Bitcoins, recebe os dados de criação do cartão de crédito e valida as mesmas através do SDK do Stripe e por fim cria também o token do Stripe para ser possível efetuar um pagamento.

    - **PaymentProvider** – é o provider responsável por ir buscar a informação do atual preço dos bitcoins à BitcoinAverage API, armazena os dados dos cartões de crédito e tem um método para fazer o pedido de efetuar o pagamento ao *backend*.

- **VuforiaVideoManager** – este manager é o responsável por inicializar os processos do Vuforia, valida a API Key e inicia todos os componentes necessários para posteriormente ser possível mostrar um vídeo através da *image recognition*.

    - **VuforiaVideoProvider** – é o provider responsável por salvar a informar dos *assets* locais (vídeos e imagens) que serão enviados para quando for necessário fazer o *render* do vídeo.

- **SensorDeviceManager** - é responsável por ir buscar toda a informação relacionada com a parte dos sensores e software/hardware do *smartphone*. Faz recurso das classes PackageManager [75], TelephonyManager [76] e GsmCellLocation [77] para obter informação detalhadas sobre as características da rede e do telemóvel.

    - **SensorDeviceProvider** – é o responsável por fazer a inicialização da captura da informação dos sensores, através da classe SensorManager [61], traduz esses dados em informação relevante e vai armazenando essa informação para posteriormente ser guardada permanentemente e analisada. Responsável também por obter e guardar informação relativa às redes WIFI que o utilizador consegue alcançar do seu *smartphone*. Em termos de sensores, pode ser capturada e armazenada informação dos seguintes sensores:
        - Ambiente/Temperatura
        - Gravidade
        - Orientação
        - Pressão
        - Proximidade
        - Humidade
        - Rotação Vetorial
        - Campo Magnético

Quanto às API's externas que também são usadas na aplicação, neste caso a Estimote API associada a informação relativa aos beacons, BitcoinAverage API [78] para ir buscar o valor atual dos bitcoins em relação ao euro e por fim a OpenWeather API para ir buscar informação relativa ao tempo dada uma localização. Todas estas API's comunicam diretamente com um Provider, que por sua vez comunica a informação recolhida ao Manager e este distribui essa informação posteriormente quando necessário.

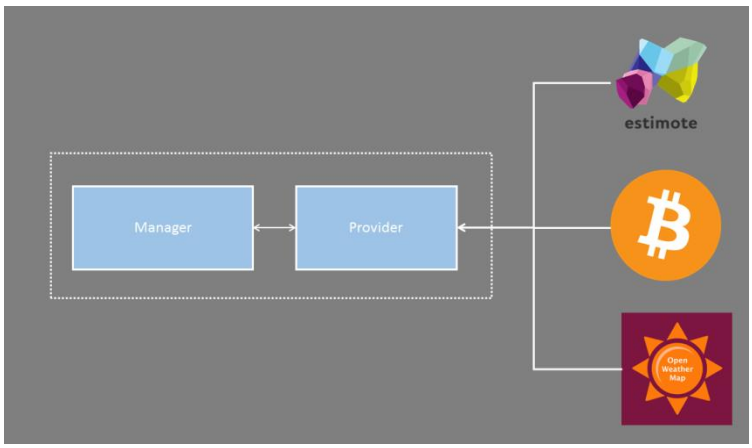



**Figura 31 - API's externas**

# 5 Desenvolvimento e integração

Nesta secção vai ser abordada todos os aspetos relativos à parte de desenvolvimento e integração de bibliotecas externas quer na parte da aplicação como do *backend*. O tópico anterior foi mais direcionado em dar um aspeto geral em como ia estar tudo organizado, estruturado e como se iam interligar o *backend* e a aplicação móvel, contudo este tópico vai ser centrar mais na parte do desenvolvimento, os passos e bibliotecas externas que foram utilizadas para chegar ao resultado final.

## 5.1 Requisitos de desenvolvimento

- **Ambiente de Desenvolvimento** – para o desenvolvimento da aplicação Android foi utilizado o Android Studio [54], utilizando o Android SDK e o Gradle [55] para fazer a gestão de versões e bibliotecas. Para o desenvolvimento do *backend* foi utilizado o PyCharm [79].

- **Gestão do Projeto** – para planear e organizar as tarefas necessárias executar relativamente à tese foi utilizado o Trello [80].

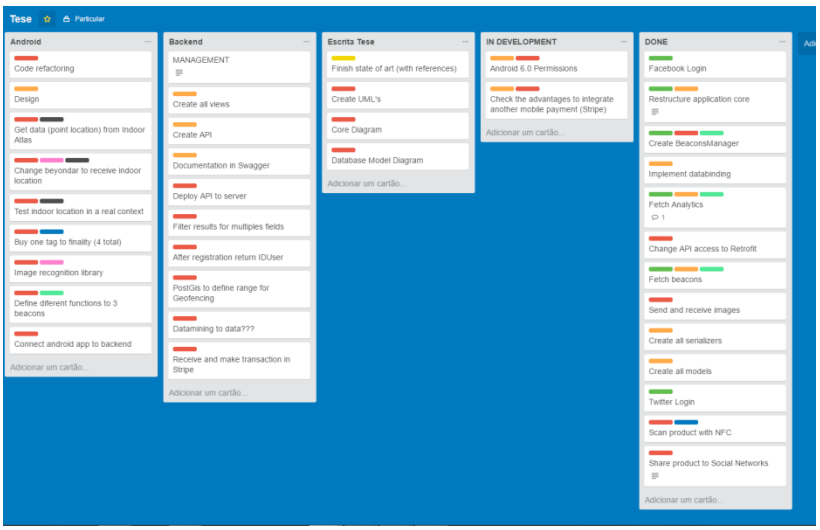



**Figura 32 - Quadro de tarefas no Trello**

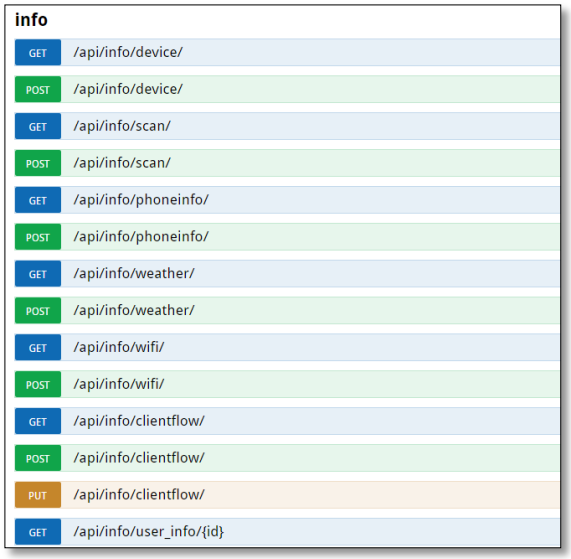
- **Armazenamento e controlo de versões** - para armazenamento de todo o código quer relativo ao *backend* quer à aplicação foram utilizados o Bitbucket [81] como repositório e o GIT [82] para controlo de versões.

## 5.2 Serviços Backend

Como foi referido a aplicação móvel faz uso de uma *backend* para o envio e armazenamento de informação, nesta secção vão ser detalhados os serviços fornecidos por esse mesmo *backend*. Todos os serviços fornecidos através do backend verão no formato JSON.

O *backend* encontra-se divido em 6 secções, cada secção contém vários serviços que vão ser necessários para o correto funcionamento da aplicação:

- **Info** – serviços responsáveis por toda a informação relacionada com os sensores, informação do *smartphone*, informação recolhida através das ações realizadas pelo utilizador.




  - **/api/info/device** – este serviço disponibiliza os métodos GET e POST. É o responsável por envio e persistência da informação relacionada com o *smartphone*.

- **/api/info/scan** – este serviço disponibiliza os métodos GET e POST. É responsável por armazenar informação relativa a quando um utilizador faz um scan num determinado produto, armazena informação sobre o produto ao qual o utilizador fez o scan, a data e a localização em que se encontrava.


```
{
  "id": 4,
  "id_product": 1,
  "id_user": 1,
  "date": "2016-04-13T12:47:48Z",
  "location": "40.6380626,-8.6353158"
}
```


- **/api/info/phoneinfo** – este serviço disponibiliza os métodos GET e POST. É responsável por armazenar a informação relativa à rede que o utilizador está a utilizar no seu *smartphone*.

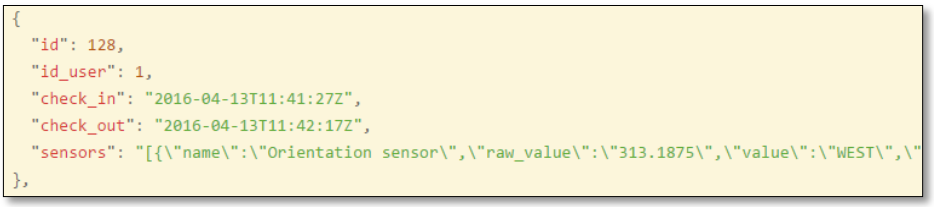

```
"id": 2,
"user_id": "1150018605026317",
"cell_id": 1618692,
"lac": 51,
"d_id": "355489065252559",
"phone_number": "",
"software_version": "",
"operator_name": "vodafone P",
"sim_country_code": "pt",
"sim_operator": "",
"sim_serial_no": "8935101811276016901",
"subscriber_id": "268010400281690",
"network_type": "",
"phone_type": "GSM",
"call_state": "IDLE",
"location_string": "[51,1618692,-1]",
"direction_string": "NONE",
"connection_state": "Disconnected",
"service_state_string": "IN SERVICE",
"phone_type_string": ""
```


- **/api/info/weather** – este serviço disponibiliza os métodos GET e POST. É responsável por armazenar a informação relativa ao tempo quando um utilizador está entrando dentro de uma loja.

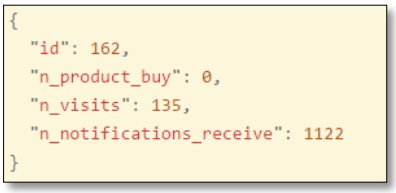

```
{
  "id": 2,
  "clouds_value": "20",
  "dt_value": 0,
  "humidity": "57",
  "pressure": "995.76",
  "sea_level": "null",
  "temperature": "12,27",
  "max_temperature": "null",
  "min_temperature": "null",
  "grnd_level": "null",
  "wind_deg": "289.01",
  "wind_speed": "1.32",
  "description": "Algumas nuvens",
  "main_description": "Clouds",
  "country": "PT",
  "city": "Aveiro",
  "icon": "02d",
  "user_id": 1
},
```

- **/api/info/wifi** – este serviço disponibiliza os métodos GET e POST. É responsável por armazenar a informação relativa às redes WIFI disponíveis quando um utilizador vai a entrar numa loja.

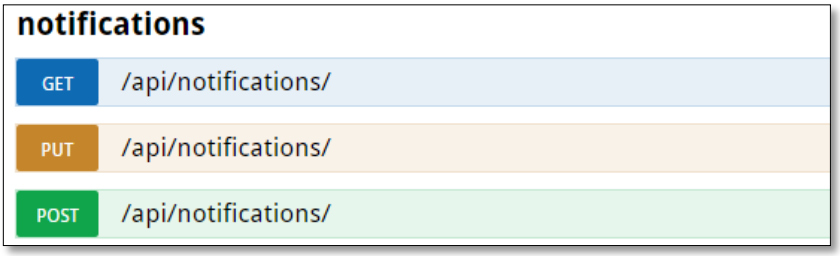

- **/api/info/clientflow** – este serviço disponibiliza os métodos GET, POST e PUT. É o serviço que armazena as horas a que um utilizador entrou e saiu da loja e todos os dados dos sensores recolhidos enquanto ele efetuou esse trajeto.




- **/api/info/user_info/{id}** – este serviço só disponibiliza o método GET. Este serviço recebe um ID, neste caso do utilizador, e responde com um conjunto de estatísticas associadas ao utilizador, entre elas o número de visitas àquela loja, o número de promoções recebidas naquela loja e o número de produtos comprados.



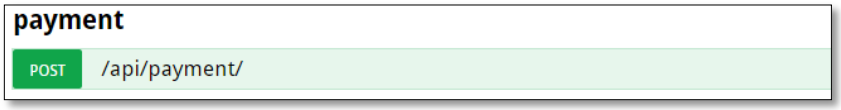
- **Notifications** – serviço responsável por receber e armazenar toda a informação relacionada com as notificações enviadas para o utilizador. Este serviço disponibiliza os métodos GET, PUT e POST, todos seguindo como resposta e receção de informação o formato JSON.



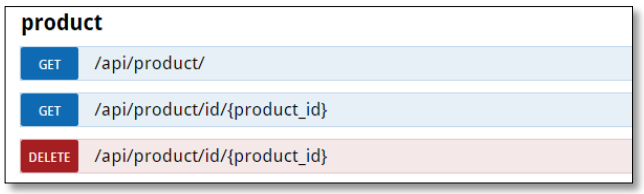

Estes serviços são disponibilizados para armazenar todas a informação relacionada com as notificações relacionadas com promoções que o utilizador recebe. Armazena informação relativa à data e hora em que a notificação foi enviada, data e hora em que o utilizador abriu a notificação, se ele relativamente a abriu e que tipo de notificação foi, ou seja, qual a ação que desencadeou o envio dessa notificação.


```
{
  "id": 2066,
  "title": "LG Nexus 5X com 50% desconto",
  "message": "Só porque decidiu aparecer hoje tome lá este desconto",
  "send_time": "2016-04-15T09:41:40Z",
  "open_time": "2016-04-14T23:00:00Z",
  "is_open": false,
  "created_on": "2016-04-15T10:41:41.857000Z",
  "updated_on": "2016-04-15T10:41:42.138000Z",
  "type": "BEACONS",
  "users_send_facebook": [
    1
  ],
  "users_send_twitter": []
}
```

- **Payment** - responsável por fazer uma transação utilizando o Stripe. Este serviço suporta somente o método POST, que vai receber um JSON com a informação do token do cartão de crédito do utilizador, a quantidade a ser cobrada (em cêntimos), a moeda e a descrição da transação. No fim de recebermos a informação, e utilizando a biblioteca do Stripe para Python, criamos uma transação e obtemos logo resposta do sucesso ou falha da mesma.



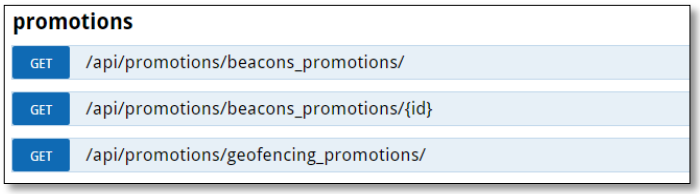

- **Product** – responsável por receber e enviar toda a informação relacionada com os produtos. Esta secção suporta somente serviços com os métodos GET e DELETE. Os dois métodos GET disponibilizam uma lista de produtos ou um produto específico se fornecido o ID do mesmo. No caso dos produtos temos também uma imagem do mesmo associada, essa imagem também é armazenada no servidor. E o método DELETE elemina um determinado produto da base de dados se for fornecido o ID do mesmo.



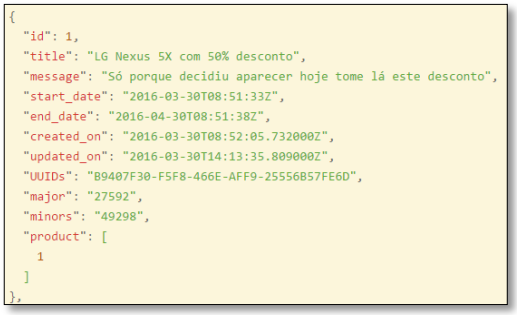


```
{
  "id": 1,
  "name": "LG Nexus 5X",
  "description": "Some think that it's time for Google to start making its own
  "price": "350.00",
  "discount": "10.00",
  "image": "/media/thumb_1165_phone_front_big.png",
  "latitude": "41.90530734",
  "longitude": "2.56580803",
  "category": "TECNOLOGIA",
  "created_on": "2016-03-16T20:59:22.912000Z",
  "updated_on": "2016-04-13T08:56:29.984000Z"
},
```

- **Promotions** – responsável por enviar todas as promoções que existem. Esta secção só contém serviços a disponibilizar o método GET.



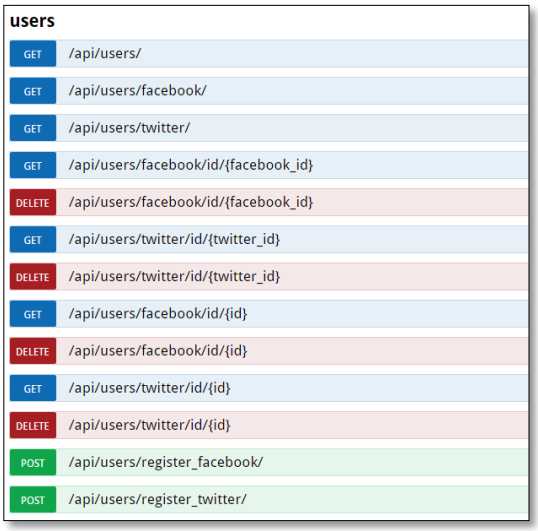

  - **/api/promotions/beacons_promotions** – este serviço retorna todas as promoções que vão ser executadas pelos beacons. Como particularidade este serviço identifica cada promoção destinada a cada beacon através do UUID, major e minor que são os identificadores específicos de cada beacon.




  - **/api/promotions/beacons_promotions/{id}** – este serviço faz exatamente o mesmo que o serviço detalhado em cima, mudando somente que este só retorna uma promoção baseada no ID da mesma.

  - **/api/promotions/geofencing_promotions** – este serviço retorna todas as promoções que vão ser executadas pelo geofencing. Como particularidade este serviço identifica cada promoção destinada cada geofence através da latitude e longitude, raio de alcance dessa mesma notificação em relação à coordenada e ainda o tipo de geofence a qual esta promoção está associada (Entrar, sair ou dentro da área do geofence).

- **Users** – responsável por armazenar e enviar toda a informação relacionada com o utilizador. Esta secção disponibiliza serviços com os métodos GET, POST e DELETE. Todos os serviços desta secção são relacionados com o registo e obtenção de informação relacionada com os utilizadores, por isso só vou detalhar os principais serviços.




  - **/api/users/facebook/id/{facebook_id}** – retorna os dados de um utilizador usando seu ID do Facebook (o Twitter segue o mesmo padrão).

  - **/api/users/facebook** – este serviço retorna toda a informação do utilizador que se encontra no *backend* após o registo do mesmo (o mesmo se aplica a um utilizador registado através do Twitter).

  - **/api/users/register_facebook** – este serviço POST recebe a informação de registo e do Facebook relativa ao utilizador (o registo através do Twitter segue o mesmo padrão).


```
"name": "João Pedro",
"profile_image": "https://graph.facebook.com/1150018605026317/picture?type=large"
"age_range_min": 21,
"device": "Android",
"email": "joaop_pedrosa@hotmail.com",
"first_name": "João",
"gender": "male",
"install_type": "UNKNOWN",
"installed": true,
"is_verified": false,
"link": "https://www.facebook.com/app_scoped_user_id/1150018605026317/",
"last_name": "Pedro",
"name_format": "{first} {last}",
"mutual_friends": 647,
"mutual_likes": 100,
"test_group": 5,
"created_on": "2016-04-07T13:08:33.170000Z",
"time_zone": "0",
"locale": "pt_PT",
"cover_image": "https://scontent.xx.fbcdn.net/hphotos-xap1/v/t1.0-9/s720x720/9981
```

## 5.3 Desenvolvimento Android

Para o desenvolvimento da aplicação em Android foi criado um projeto no Android Studio, onde se usa o Gradle para fazer o controlo de versões e gestão de dependências e o Android SDK. Na imagem abaixo é possível ver o funcionamento geral da aplicação com as principais bibliotecas e SDK's que são necessários para a aplicação. Vou de seguida detalhar o porquê e como foi utilizada cada biblioteca e SDK. Em termos de suporte e configuração o projeto no Android Studio encontra-se programado para suportar como API mínima a 16 (*Jelly Bean*) e como versão target a 23 (*Marshmallow*).

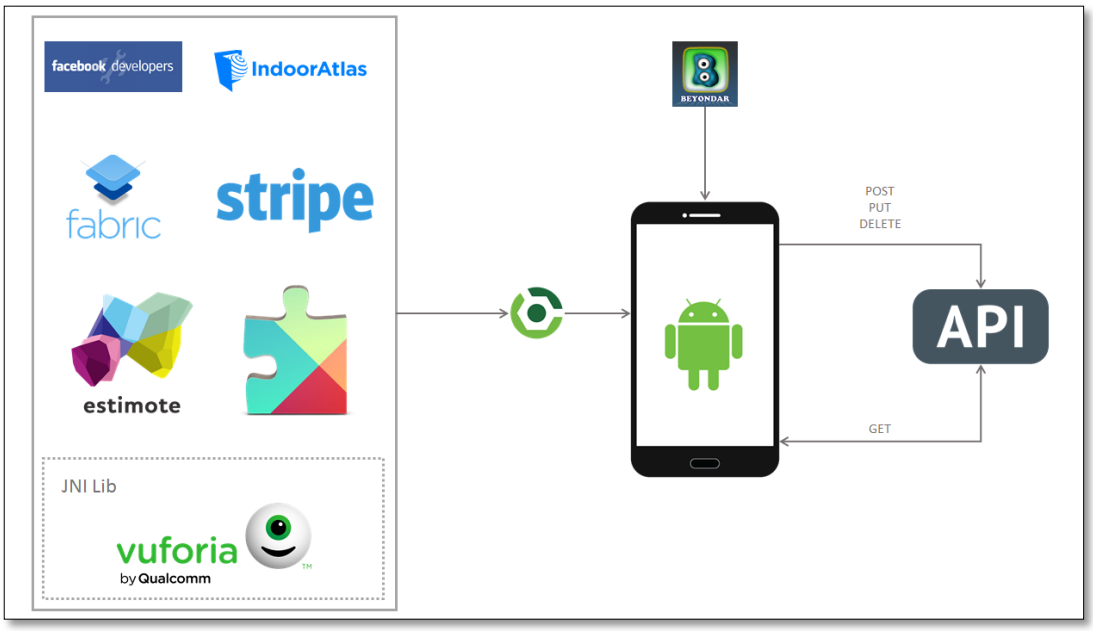



**Figura 33 - Integração dos SDK e API com a aplicação Android**

### 5.3.1 Integração Redes Sociais

O **Facebook** SDK [56] foi incluído neste projeto para ser possível ao utilizador fazer autenticação através da sua conta da rede social, e posteriormente poder partilhar informações relativas à aplicação nessa mesma rede social. A principal vantagem deste SDK é que é ele que gere toda a parte da autenticação e se chama a aplicação nativa do Facebook (se o utilizador a tiver instalada no seu *smartphone*) ou se chama a versão Graph [83] do Facebook, assim sendo o utilizador não demora tempo nenhum a efetuar o seu login (evitando os longos questionários de registo) e do cliente da loja consegue obter informação extra sobre o seu cliente que no futuro pode ser importante.

Para o SDK funcionar corretamente também é necessário criar uma conta no Facebook Developers [84] e registar a nossa aplicação para ser possível obter uma *private key* e a App ID que vai estar associada à nossa aplicação, onde temos lá todas as permissões que queremos entre outras informações relacionadas com estatística e customização de informação da aplicação. Além disso enquanto estamos na fase de desenvolvimento precisamos de obter uma Key Hash associada à nossa conta, para ser possível testar o login e todas as outras funcionalidades associadas ao Facebook.

Em termos de desenvolvimento vamos ter 3 classes principais que vão ser usadas para efetuar o login e obtenção de dados com o Facebook SDK:

- **LoginManager** – é a classe que vai controlar o inicializar o processo de login, é a classe que contêm as permissões que estamos a solicitar ao utilizador para este poder efetuar o login. É nesta classe que também vamos registar o callback para quando recebermos a informação do sucesso ou falha no login.

- **CallbackManager** - é o callback entre a nossa aplicação e o Facebook SDK, é chamado no onActivityResult da LoginActivity.

- **AccessToken** – contém o token de sessão, contém informação sobre o ID do utilizador, as permissões aceites e negadas e serve também para fazer pedidos à Graph API do Facebook.

No fim de fazer o pedido e obtermos a resposta do CallbackManager é-nos retornado um objeto JSON contendo a informação que foi pedida nas permissões inseridas no LoginManager. Nem todas as permissões são retornadas, pois em algumas é necessário a aprovação do próprio Facebook que só nos dará essa permissão depois de fazermos um requerimento descrevendo para que queremos e precisamos dessa informação.

Além destes passos também é necessário inserir algumas configurações no ficheiro AndroidManifest.xml, entre os quais uma tag *metadata* contendo o App ID da aplicação no Facebook Developers e ainda declarar a FacebookActivity, para o caso do utilizador ter a aplicação nativa do Facebook instalada essa poder ser chamada através da aplicação para o utilizador efetuar o seu login, e em passos futuros poder partilhar outras informações no Facebook.

Além do Facebook já referido foi integrado também o Fabric [57]. O **Fabric** é uma Mobile Development Platform desenvolvida pelo **Twitter**, é bastante recente e começou por conter apenas como SDK's de integração o **Crashlytics** e o Twitter, mas neste momento já conta com mais de 12 integrações possíveis. O intuito de integração desta funcionalidade na aplicação é que com pouco esforço conseguimos integrar na nossa aplicação funcionalidade cruciais à mesma, no caso da nossa aplicação foi integrado o SDK do Twitter e do Crashlytics. O **Crashlytics** [85] é útil para distribuições da versão beta da aplicação, termos informação de métricas sobre a utilização da mesma e para recebermos *bugs reports* dos vários utilizadores que estão a testar a aplicação. Isto é uma ferramenta fundamental pois devido aos elevados número de dispositivos diferentes e versões diferentes Android que existem neste momento no mercado, podem surgir problemas diferentes em cada um deles, e esta ferramenta é um auxiliar para ser possível descobrir a causa do problema

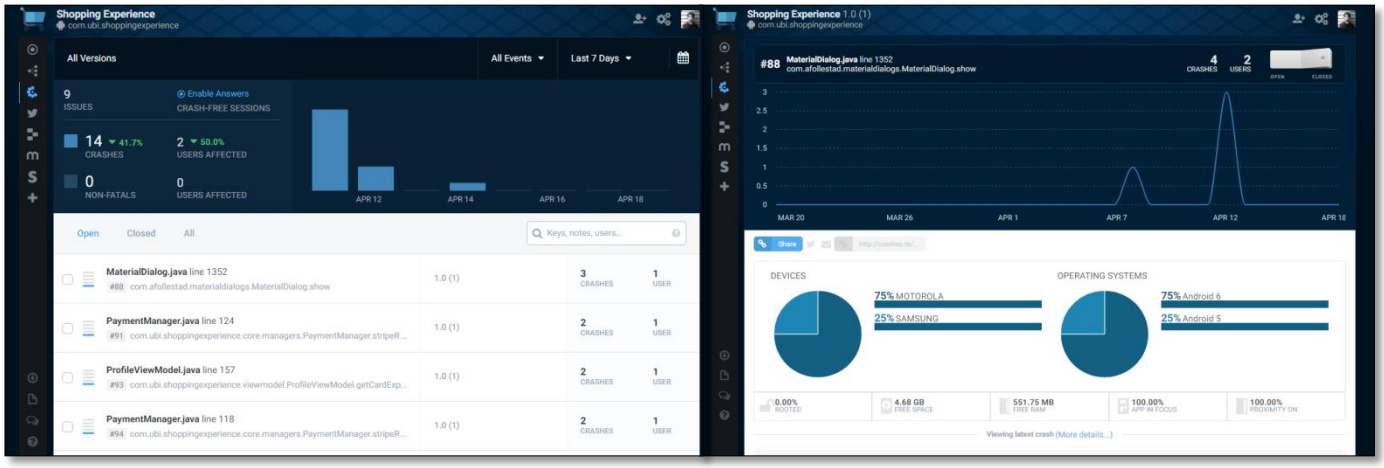



**Figura 34 - Fabric dashboard**

Já na parte do SDK do Twitter o objetivo principal passa por ser possível efetuar login na aplicação através da conta de Twitter do utilizador, e recolher alguma informação associada ao mesmo. Em termos de funcionamento assemelhasse muito ao login através do Facebook, a principal diferente é que no caso do Twitter o botão que se encontra no UI para efetuar o login é uma classe própria do Twitter e ao clicarmos nele é ele que desperta o Callback para depois recebermos a informação do utilizador. Em termos de classes importantes neste processo temos basicamente duas:

- **TwitterSession** – Classe que irá conter os dados de sessão do utilizador (contendo também o token de sessão).

- **TwitterApiClient** – Classe que acede à API do Twitter para ir recolher a informação pessoal do utilizador.

### 5.3.2 Integração indoor location

A **IndoorAltas** como já foi falado em cima é uma empresa que se especializou e contem várias patentes na área da *indoor location* obtida através do campo magnético dos edifícios. O SDK deles para Android foi integrado neste projeto para ser possível fazer uma localização indoor precisa e em que o utilizador saiba sempre onde está e onde se encontram os produtos que ele pretende comprar.

O primeiro passo para integração deste SDK é criar uma conta de *developer* no site da IndoorAltas, eles têm um plano gratuito de utilização se a aplicação não ultrapassar os 100 utilizadores únicos por mês.

Após a conta ser criada o primeiro passo a fazer é criar o que chamam na IndoorAtlas de Venue. Uma **Venue** não é nada mais que a planta ou plantas (se for um edifício de vários andares), e temos de pegar nessa imagem e inseri-la no mapa num contexto real, esta imagem pode ir sendo ajustada e vai ser a imagem que depois irá aparecer na aplicação móvel posteriormente. Este passo serve para saber qual o enquadramento, ao nível de coordenadas geográficas, do edifício que queremos fazer a localização indoor. No fim deste passo terminado fica-se com ID's de 3 aspetos importantes:

- **FloorplanID** – o ID associado à imagem daquele piso do edifício.
- **FloorID** – o ID associado àquele piso do edifício.
- **VenueID** – o ID da Venue que foi acaba de criar.


**Figura 35 - Criar uma venue no IndoorAtlas**

Depois de se criar a Venue o próximo passo é recolher informação sobre o campo magnético do edifício e fazer *upload* dessa informação para a IndoorAltas Cloud. Esta informação é recolhida através da aplicação móvel da IndoorAltas [86]. O procedimento consiste basicamente em definir todos os caminhos que serão necessários traçar no mapa e percorrê-los com o telemóvel na mão, à medida que se percorre os vários caminhos a aplicação móvel vai recolhendo informação relativa ao campo magnético dos edifícios em cada um desses trajetos, é assim que futuramente que o IndoorAtlas vai identificar a nossa localização atual. No fim de se traçar todos os caminhos da nossa planta basta finalizar a operação e todos os dados serão armazenados na Cloud do Indoor Atlas associando esses dados à Venue previamente criada.

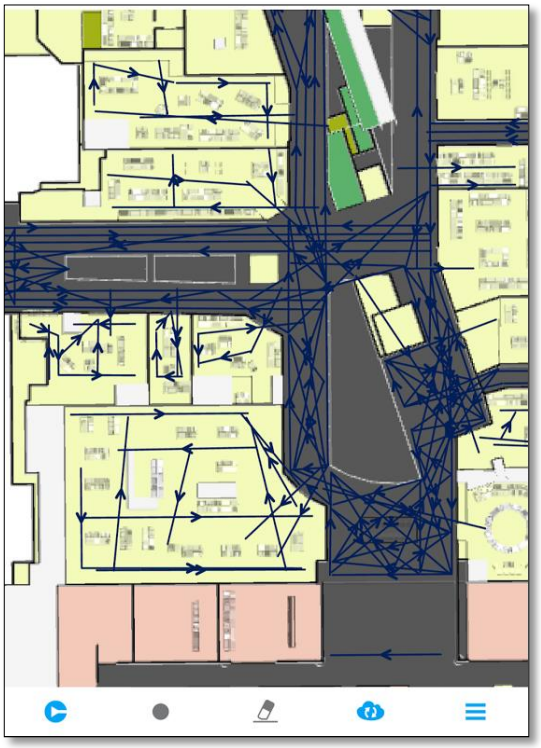


**Figura 36 - Traçar e fazer upload dos dados através da aplicação do IndoorAtlas**

O último passo em termos de configurações é criar uma aplicação dentro do site da IndoorAtlas e assim ficamos com uma **API KEY** e uma **API SECRET** que posteriormente inserida no projeto da aplicação. No fim de se concluir estes passos todos já é possível integrar o SDK do IndoorAltas no projeto.

Agora falando da parte de desenvolvimento, para ser possível uma correta integração do SDK temos de inserir no AndroidManifest.xml a API KEY e API SECRET que foram obtidas anteriormente, além disso é também necessário integrar um serviço do próprio SDK, o **IALocationService**, que quando inicializado irá mandando constantes atualizações sobre a posição atual (latitude e longitude) do utilizador.

No fim de todos os procedimentos de integração do SDK validados, é necessário fazer os acessos à Cloud do IndoorAtlas através do seu SDK, todos estes procedimentos serão feitos dentro do **IndoorLocationManager** que já foi falado acima, e todos os resultados obtidos serão mostrados no **MapFragment**. Em termos de classes fundamentais ao funcionamento da localização indoor temos:

- **IALocationManager** – responsável por inicializar e parar o serviço de localização inserido no AndroidManifest.xml, que vai mandando atualizações acerca da localização do utilizador. É inicializado no **onResume()** do MapFragment, e parado no **onPause()** e por sua vez destruído no **onDestroy()** [87]. Esta classe é inicializada recebendo como argumento uma instância da class **IARegion** onde enviamos como argumento o FloorPlanID da imagem do piso em que queremos receber atualizações sobre a posição do utilizador.

- **IAResourceManager** – responsável por fazer o pedido de todas as informações associadas à planta.

- **IAFloorPlan** – classe que contém toda a informação associada à planta, é informação recebida no fim de feito o pedido através do **IAResourceManager**.

- **IAResultCallback<IAFloorPlan>** - *callback* que recebe a imagem da planta associada ao FloorPlanID.

- **IARegion.Listener** – *listener* que fica à escuta de alterações da localização duma região (como região entende-se um piso, ou seja, uma *venue* com várias regiões é um edifício com vários pisos. O **IALocationService** vai mandado atualizações sobre a posição atual do utilizador, e este *listener* vai associar essas coordenadas com a região em que o utilizador se encontra). Contém ainda métodos que determinada quando um utilizador entra e sai duma determinada região.

Para ir atualizando o utilizador acerca da sua posição atual foi ainda incluído uma *view* chamada de **BlueDot** que não é nada mais que um círculo azul. Após de carregada a imagem da planta correspondente, queremos saber a posição atual do utilizador nessa planta visualmente, e assim sendo à medida que é recebida uma atualização na posição do utilizador, é convertida a latitude e longitude recebida para um *point,* que irá ser a posição em X e Y em que o utilizador se encontra relativamente à imagem da planta.

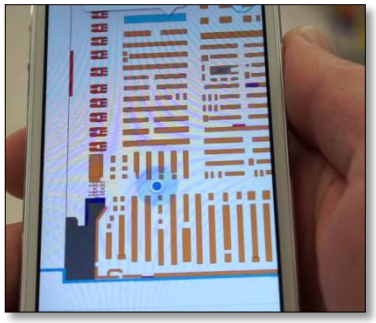


**Figura 37 - Indoor location através do IndoorAtlas**

### *5.3.3* Integração *mobile payments*

A integração com o SDK da **Stripe** [88] realizou-se para ser possível efetuar pagamentos através do cartão de crédito na aplicação. Contudo do lado da aplicação vai estar somente a parte de criação do cartão de crédito, validação do mesmo e criação do token para esse cartão, a transação em si é feita do lado do *backend*. Para a integração do SDK tanto do lado da aplicação como do *backend* precisamos de criar uma conta no site da Stripe para termos acesso a uma API KEY. Em termos de desenvolvimento o Stripe fornece uma API KEY de testes para ser possível testar a solução desenvolvida sem serem feitas transações na realidade, e através da *dashboard* que fornecem também no site deles conseguimos ter acesso às transações que foram feitas, a que horas e por quem.

Para a criação do *token* associado ao cartão de crédito basta instanciar a classe **Stripe** e chamar o método **createToken()** que recebe como argumentos um objeto **Card** (com os dados do cartão de crédito), a API KEY e ainda um *callback* que nos informa se o *token* foi criado com sucesso. Posteriormente a este passo para efetuar uma transação basta fazer um POST ao *backend* com esse *token*, o montante a cobrar, a moeda e uma descrição da transação.

O Stripe também já suporta integração com o **Android Pay** [33], contudo o uso de Digital Wallets ainda não é muito comum tanto em Portugal como no resto da Europa, dai a não integração.

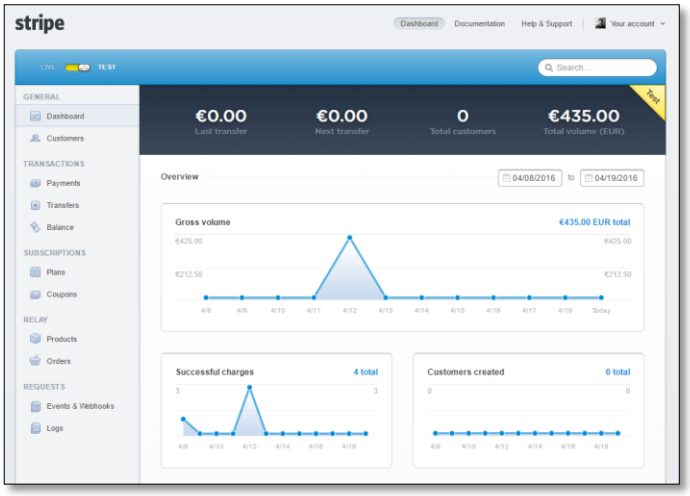



**Figura 38 - Stripe dashboard**

### 5.3.4 Integração beacons

O SDK da **Estimote** foi integrado para ser possível comunicar com os beacons através da aplicação Android. O primeiro passo neste processo é adquirir os beacons, que podem ser encomendados através da própria página da Estimote. No fim desse processo feito cria-se uma conta com o mesmo email com o qual a encomenda e ao efetuar o login chega-se a uma *dashboard* onde se vê todas as informações associadas aos beacons que foram adquiridos (é o sitio onde se pode alterar alguma dessa informação). No fim do processo concluído é necessário criar uma aplicação nova no mesmo site que será a aplicação associada ao projeto, no fim desta estar criada temos acesso a um APP ID e um APP TOKEN que serão posteriormente inseridos no projeto da aplicação e servirão para identificar os beacons físicos que estarão associados àquela aplicação.

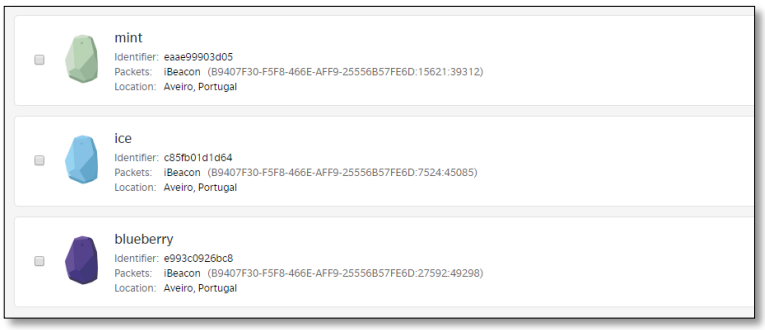



**Figura 39 - Estimote dashboard**

Em termos da integração do SDK na aplicação o primeiro passo será inserir no *AndroidManifest.xml* [89] as permissões de uso do Bluetooth pois será através desta tecnologia que vai ser possível detetar os beacons. Posteriormente na *Application* da aplicação móvel é necessário fazer a inicialização do SDK com as KEYS que foram obtidas acima. Em termos de classes fundamentais neste SDK existe basicamente uma, que é o **BeaconManager** da própria Estimote. Esta classe permite controlar todos os aspetos relacionados com os beacons, e em termos dos beacons temos duas principais funcionalidades:

- **Monitoring** – cria uma região virtual com um raio aproximado de 70 metros (valor por defeito, para alterar este valor temos de baixar o valor de *transmit power* dos beacons) e deteta movimentos dentro dessa região (entrada e saída da região, se se está a mover dentro da região). Segundo a Estimote é possível monitorizar ate 20 regiões ao mesmo tempo.

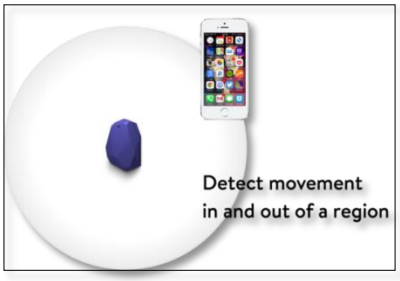



**Figura 40 - Monitoring através dos beacons da Estimote**

- **Ranging** – deteta os beacons duma determinada região que se encontram nas proximidades e consegue informar a que distância está dos mesmos (com valores atualizados a cada segundo). Esta funcionalidade só funciona se a aplicação estiver a correr, ao contrário do Monitoring onde continuamos a receber os eventos mesmo que a aplicação esteja em *background*.

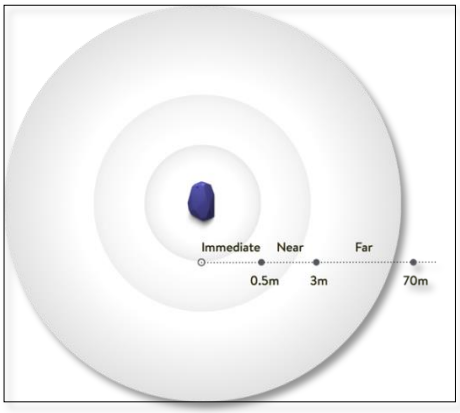



**Figura 41 - Ranging através dos beacons da Estimote**

Estas duas funcionalidades fornecem *callback* bastante importante, em *Monitoring* temos como *callback* as funções **OnEnteredRegion()** e **OnExitedRegion()** onde se consegue receber em cada uma delas a região (e o conjunto de beacons a ela associada) da qual o utilizador está a entrar e a sair respetivamente. Já na parte do *Ranging* temos **OnBeaconsDiscovered()** em que recebemos a informação da região e dos beacons que entraram no alcance do utilizador.

Estas duas funcionalidades estão a ser expostas no projeto (de notar que as duas não são compatíveis com a mesma instância do **BeaconManager**, e assim sendo para conseguir ter as duas a funcionar ao mesmo tempo, cada uma terá de ter uma instância diferente da classe **BeaconManager**). De notar em cima que foi falado em região, mas não foi explicado o seu significado e é um conceito importante na área dos beacons, uma região é uma barreira virtual criada pela agregação de um, dois ou todos os beacons, ou seja, com 3 beacons conseguimos criar várias regiões distintas e cada uma com um propósito diferente.

Além destas funcionalidades explicadas acima é ainda possível saber a temperatura do ambiente em que se encontra cada beacon, saber se um utilizador se encontra em movimento ou se está parado. O conjunto de aplicações para estas funcionalidades é infinito, por isso, é que foi criado no contexto da aplicação a classe própria **BeaconsManager,** para expor todas as funcionalidades e ser possível inicializar e começar a trabalhar com estas funcionalidades duma forma mais simples e rápida.

### 5.3.5 Integração geofencing

Para a integração do geofencing foi necessário integrar o **Google Play Services**, que é a biblioteca que comunica com os serviços e classes que a Google disponibiliza para o desenvolvimento Android. O Google Play Services fornece um conjunto enorme de funcionalidades, que antigamente se encontravam todas juntas na mesma biblioteca, contudo devido à limitação do desenvolvimento em Android que não suporta mais do que 64 mil métodos (por causa de limitações nos DEX files) [90], em versões mais recentes a Google dividiu os seus serviços em categorias e no caso do projeto vamos simplesmente importar a biblioteca associada ao Google Play Services Location [91], para ser possível efetuar o geofencing. Esta biblioteca é muito usada quando queremos ter uma aplicação *location aware*, envolve todo um conjunto de funcionalidades e ferramentas associadas à localização do utilizador. Como principais funcionalidades fornece a localização atual do utilizador, atualizações constantes relativas à localização do utilizador e ainda tradução de coordenadas em endereços.

Em termos de programação o primeiro passo é ligar a aplicação ao **Google API Client** através da função **buildGoogleApiCliente()**, e informar que queremos aceder à API de **LocationService**. Após a ligação ser estabelecida com sucesso podemos chamar a classe **Geofence.Builder()** e criar as *geofences* através do método **generateGeofencing()**, e por fim inserirmos essas mesmas *geofences* criadas na API dos LocationServices da Google. Para a criação duma *geofence* precisamos dum ID da mesma, a latitude, longitude, raio de alcance o tipo e o tempo de expiração.

```
protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this.mContext)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();
    mGoogleApiClient.connect();
}
```

```
@Override
public Geofence generateGeofencing(String title, String message, double latitude,
    return new Geofence.Builder()
            .setRequestId(title+ ":" +message)
            .setCircularRegion(latitude, longitude,radius)
            .setExpirationDuration(GEOFENCE_EXPIRATION_IN_MILLISECONDS)
            .setTransitionTypes(type)
            .build();
}
```

```
public void addGeofencingToGoogleLocationServices() {
    if (!mGoogleApiClient.isConnected()) {
        Log.e(TAG,"Google API Client not connected!");
        return;
    }

    try {
        LocationServices.GeofencingApi.addGeofences(
                mGoogleApiClient,
                getGeofencingRequest(),
                getGeofencePendingIntent()
        ).setResultCallback(this);
    } catch (SecurityException securityException) {
        logSecurityException(securityException);
    }
}
```

**Figura 42 - Código de comunicação com a Google Play Services e criação das geofences**

Assim como nos beacons, também para a parte do geofencing foi criada uma classe própria, GeofencingManager, para ser possível criar, eliminar e gerir geofences duma forma rápida e eficiente. É um módulo independente que futuramente pode ser integrado em qualquer outro projeto que queira utilizar esta funcionalidade.

### 5.3.6 Integração realidade aumentada (image recognition)

A integração do SDK da **Vuforia** tem como funcionalidade a possibilidade de integração da realidade aumentada através de *image recognition*. **Image recognition** é uma das funcionalidades do SDK da Vuforia e permite mostrar um vídeo ou imagem quando apontando para uma superfície ou imagem na vida real, e assim criando a realidade aumentada.

O primeiro passo neste processo além do download do SDK (de notar que o SDK contém duas vertentes, a parte em java e a parte em C/C++, que será colocada numa pasta à parte e será também compilada pelo Gradle), é criar uma conta de developer na Vuforia, posteriormente é necessário criar uma licença, que neste caso será uma licença de Starter que tem algumas limitações, contudo é gratuita. No fim da conta criada temos acesso à License KEY será posteriormente inserida no projeto da aplicação para validar a nossa sessão. No fim de todas as configurações em termos de validação de utilização do SDK o próximo passo será ir à secção dos Target Managers e criar uma nova base de dados do tipo Device, vai ser nesta base de dados que vamos inserir as imagens que vão funcionar como target a mostrar os elementos de realidade aumentada. Ao inserirmos uma imagem conseguimos ver as várias *features* (pontos amarelos) que esta imagem contém e isso dará um ranking de quanto *augmentable* será a imagem (neste caso 3 estrelas). Para uma imagem poder exercer a função terá de ter pelo menos 3 estrelas, senão não será distinta o suficiente para puder ser usada como target. Quanto mais características (relevos, cores distintas, múltiplas formas), mais *augmentable* a imagem será e de mais fácil identificação será.

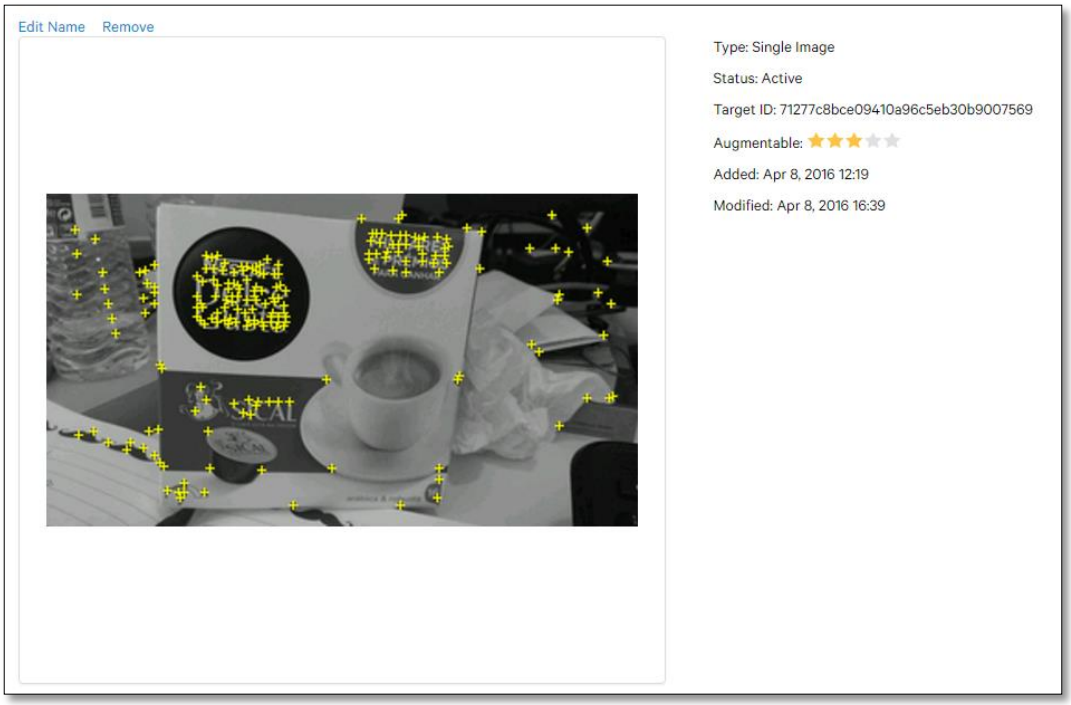



**Figura 43 - Inserção duma imagem para ser uma possível *image recognition* através do Target Manager da Vuforia**

No fim de serem inseridas todas as imagens (no máximo 80 na versão *Starter*) que vão funcionar como target, já se pode fazer download dessa base de dados que não será nada mais que um ficheiro **.xml** e um ficheiro **.dat** que contém os dados que o SDK da Vuforia irá reconhecer para identificar a imagem como target. Esses ficheiros serão inseridos na pasta **assets** do projeto da aplicação, juntamente com as imagens ou vídeos que queremos mostrar quando apontarmos para uma *image recognition*.

Em termos programáticos o primeiro passo será criar uma instância da classe Vuforia, e inicializar a mesma classe utilizando a License Key obtida anteriormente. Após isso é inicializada a câmara e os *trackers* que vão ser os responsáveis por ficar "à escuta" para verificar se alguns dos padrões para que a câmara está a apontar, corresponde aos padrões encontrados nas imagens que foram inseridas anteriormente na base de dados, se for o caso é criada uma **GLSurfaceView** [92] que será sobreposta por cima do padrão reconhecido e dentro dessa GLSurfaceView será mostrada a imagem ou vídeo que é suposto mostrar para aquela *image recognition*. Em termos de classes a ter em conta neste SDK existem:

- **DataSet** – classe que irá conter a informação do ficheiro .dat (o ficheiro .xml contém as configurações do vários *trackables* contidos no ficheiro .dat).
- **Tracker** – contém o algoritmo que deteta e acompanha o objeto na vida real.
- **TrackerManager** – aceder aos *trackers* criados e disponíveis.

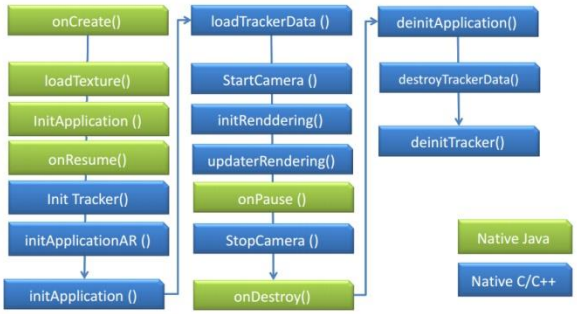



**Figura 44 – Life-cycle de inicialização do SDK da Vuforia**

Do ponto de vista do utilizador este só verá um *Fragment* que irá ter aberta a câmara e basta percorrer o espaço em que se encontra com o telemóvel e encontrar alguma *image recognition* para conseguir ver o resultado da realidade aumentada

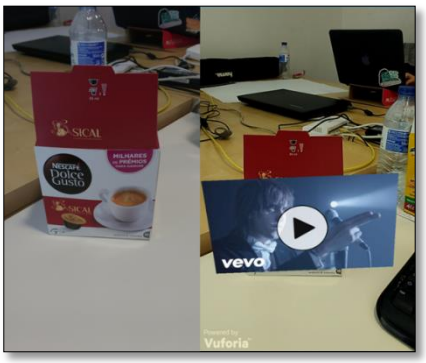



**Figura 45 - Resultado do *image recognition* da Vuforia**

### 5.3.7 Integração realidade aumentada (geolocalização)

**BeyondAR** é uma biblioteca que permite desenvolver realidade aumentada através da geolocalização. É uma biblioteca que serviu inicialmente para um desenvolvimento de um jogo mobile, contudo posteriormente o código foi colocado *open source*. Foram incluídas duas bibliotecas de realidade aumentada (BeyondAR e Vuforia) pois cada uma tem uma funcionalidade diferente.

Em termos programáticos, esta biblioteca foi usada mas ainda foi modificada pois continha alguns erros que tiveram de ser corrigidos antes de ser possível a sua utilização. Em termos de classes a ter em conta nesta biblioteca existem:

- **BeyondarObject** – objeto usado para na realidade aumentada, é o objeto que contém as coordenadas e todas as informações associadas ao objeto a ser mostrado.
- **World** – é o objeto que vai receber todos os *BeyondarObjects* que vão ser mostrados na realidade aumentada.
- **BeyondarFragment** – *Fragment* que vai mostrar e controlar a *CameraView* e *BeyondarGLSurfaceView* (classe que vai mostrar o render do *World* e mostrar esse resultado por cima da *CameraView*).
- **LowPassFilter** – classe que irá controlar a sensibilidade dos sensores (sensores de movimento e giroscópio).
- **ARRenderer** – responsável por fazer o render do World para OpenGL.

Para o lado do utilizador os resultados no "mundo" vão sendo mostrados à medida que este vai rodando o seu telemóvel baseando os resultados na sua localização atual. Para evitar muito ruido visual, foi definido como máxima distância para mostrar objetos de 1KM, ou seja, se um objeto estiver a mais de 1KM da posição atual do utilizador este já não vai aparecer. Já como em cima referido, também foi criado um ARManager, para no futuro e independentemente do resto esta tecnologia puder ser utilizada duma forma mais rápida e eficiente.

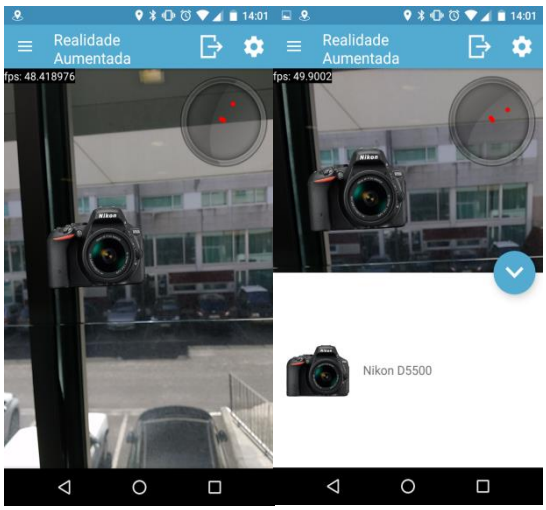



**Figura 46 - Resultado do uso do BeyondAR para realizar realidade aumentada através da geolocalização**

## 5.4 SESDK – *Shopping Experience* SDK

O principal objetivo desta dissertação é a criação duma aplicação "base" que pudesse no futuro ser reutilizada e aplicada a vários casos de uso distintos. Contudo ao longo do desenvolvimento, e à medida que iam sendo integradas cada vez mais tecnologias (por vezes o pouco complexas ao nível de integração), foi pensado que se poderia também desenvolver uma biblioteca que pudesse ajudar neste processo de integração destas tecnologias distintas com o mínimo esforço possível, e foi assim que foi criado o SESDK.

Ao longo do desenvolvimento e integração das tecnologias conseguiu se perceber que tanto a parte de configuração como desenvolvimento para integração da própria tecnologia era muito custosa tanto ao nível de tempo despendido como da perceção dos passos necessários para ela funcionar, e assim sendo decidiu-se criar este SDK. O SDK foi disponibilizado no GitHub [93] com toda a documentação necessária à sua integração.

Em termos de integração do SDK é bastante simples, basta importa a pasta correspondente ao SDK para o diretório do projeto e compila-lo em conjunto com o projeto, de seguida basta instanciá-lo no módulo *Application* do projeto e a partir dai basta inicializar os Managers que se pretende usar.

```
public class SampleApplication extends Application{
    @Override
    public void onCreate() {
        super.onCreate();
        SE.init(this);
    }
}
```

**Figura 47 - Instanciação do SESDK num projeto**

O SDK contém 6 Managers distintos, cada um responsável por simplesmente facilitar o processo de inicializar as ferramentas como também de chamar as funcionalidades principais de cada um.

- **BeaconsManager** – Inicialização dos SDK da Estimote e callbacks das funcionalidades principais do mesmo.
- **GeofencingManager** – Registo de geofences e *Service* que está à "escuta" de quando entramos numa delas.
- **PaymentManager** – Inicialização do SDK da Stripe e validação de cartões de crédito e criação do Stripe Token.
- **VuforiaVideoManager** – Inicialização do SDK da Vuforia e possibilidade de construção de realidade aumentada mostrando um vídeo através de image *recognition*.
- **MapManager** – Inicialização do SDK do IndoorAtlas e callback com imagem do mapa.
- **ARManager** – criação de realidade aumentada através de geolocalização usando a biblioteca Beyondar.

## 5.5 Aplicação Android

Nesta seção vai ser detalhado o *workflow* da aplicação que foi desenvolvida para demonstrar e todas as funcionalidades que já acima foram referidas. Esta secção também ajuda a perceber todo o trabalho que foi desenvolvido ao nível de programação sem contar com a integração das bibliotecas e SDK já falados em cima.

### 5.5.1 Login e Registo

A imagem abaixo demonstra o processo de login/registo do utilizador na aplicação. No ecrã inicial é possível ver os botões para efetuar o login/registo do utilizador, este ecrã a partir do momento que o utilizador tenha conta só aparecerá a primeira vez, pois se o utilizador já estiver credenciado a aplicação fará o *auto login* e enviará logo para o último ecrã desta secção. O utilizador só terá de passar pelo menu de aprovação tanto do Facebook como do Twitter a primeira vez. Se o utilizador tiver as aplicações nativas tanto do Facebook como do Twitter serão essas as *Activities* que serão chamadas para efetuar o login, se por sua vez não possuir nenhuma delas será chamada uma *WebView* para proceder à autenticação.

Se todo o processo de login/registo correr conforme o esperado o utilizador será direcionado para o último ecrã que se encontra na imagem abaixo, se surgir algum problema aparecerá uma mensagem com o erro e o utilizador retornará ao primeiro ecrã.

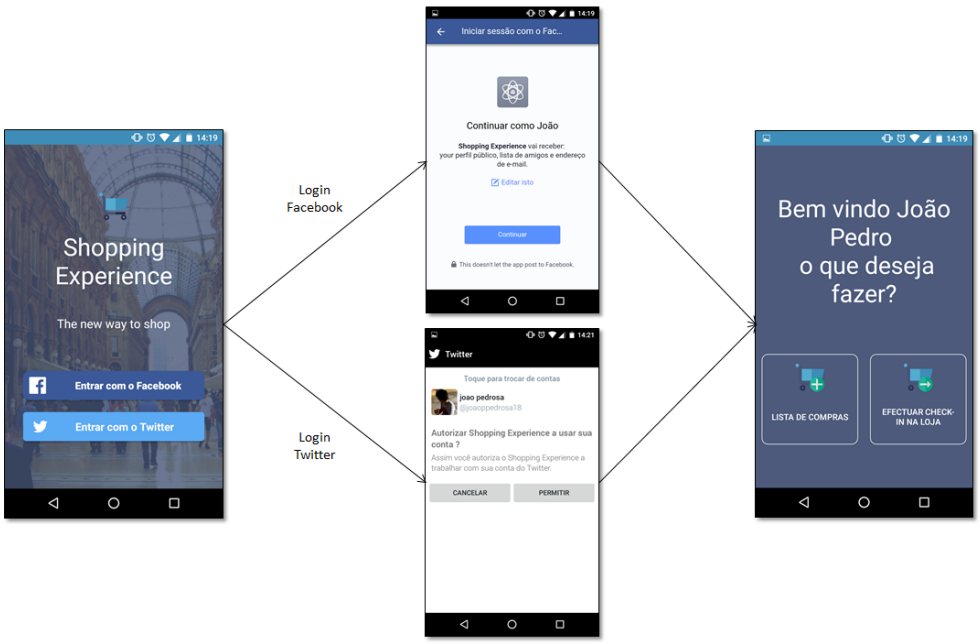



**Figura 48 - Ecrãs associados ao processo de login/registo**

### 5.5.2 *Shopping List* e Check-in

Na imagem abaixo pode-se ver os ecrãs que ocorrem no fim do utilizador efetuar o login e previamente a efetuar o seu check-in na loja. No primeiro ecrã aparece uma mensagem de boas vindas, seguida de dois botões com duas opções distintas: efetuar uma lista de compra ou efetuar check-in na loja. Se o utilizador escolher efetuar uma lista de compras (ecrãs de baixo) surge-lhe uma lista de produtos disponíveis onde ele pode ver detalhes sobre os mesmos, clicar sobre eles para receber mais informações, adicionar os produtos a um carrinho, pesquisar por um produto específico, e por fim pode ver uma lista final dos produtos que selecionou e salvá-la.

Se por sua vez o utilizador pretender efetuar o seu check-in na loja ai surge novamente duas opções, se *smartphone* do utilizador tiver a tecnologia NFC ser-lhe-á pedido para que este faça check-in encostando o seu telemóvel a uma TAG NFC que se deverá encontrar na entrada da loja se por sua vez o *smartphone* não tiver NFC, aparecerá automaticamente ao utilizador a opção para efetuar check-in através de QRCode, e assim sendo aparecerá um QRCode Scanner onde o utilizador só terá de digitalizar um QRCode que por sua vez também se encontrará à entrada da loja.

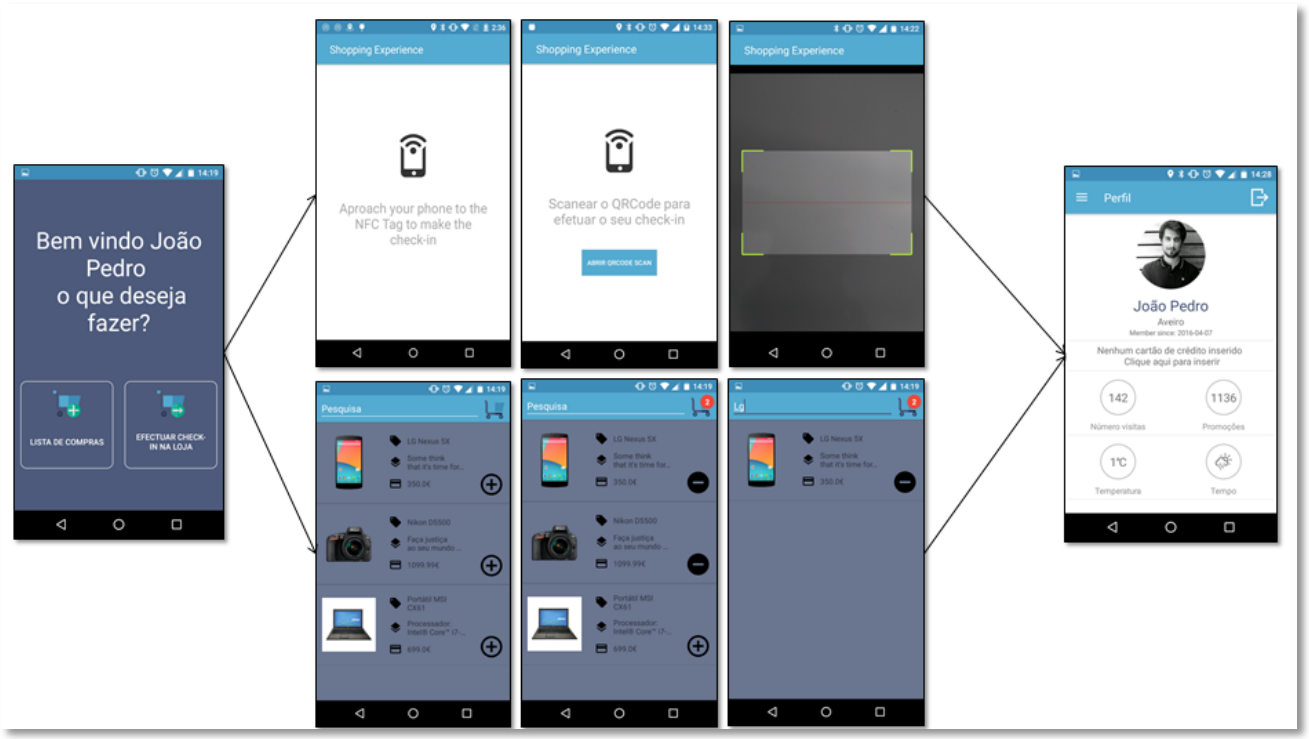



**Figura 49 - Ecrãs associados à *shopping list* e ao check-in**

### 5.5.3 Perfil e Check-out

No fim de efetuarmos o check-in na loja com sucesso, aparecerá um menu com várias funcionalidades que podemos usar dentro da loja, neste tópico vou falar da secção do perfil. Nesta secção conseguimos ter informação acerca do registo do utilizador (nome, localização, data de registo), sobre a sua interação com aquela loja (número de visitas, número de promoções recebidas). Contém também informação sobre o tempo que se encontra na sua localização (esta informação é mais relevante para o futuro processo de *datamining*) e pode ainda inserir e visualizar informação de dados do cartão de crédito que no futuro será usado para efetuar pagamentos.

Na parte superior de todos os ecrãs encontra-se um botão para fazer check-out, ao fazer check-out pode-se efetuar o pagamento através de Bitcoins ou Stripe. Se for feito através de bitcoins, será apresentada uma mensagem com o total que temos a pagar juntamente com a conversão desse valor para bitcoins, posteriormente basta clicar no botão de pagamento e transferir essa informação para uma wallet de bitcoins (que neste caso será a wallet da conta da loja). Se o pagamento for feito através do Stripe pode-se ver o mesmo ecrã com o valor que é necessário pagar, e ao clicar no botão de pagamento essa informação é enviada para o *backend* onde será então feita a transação.

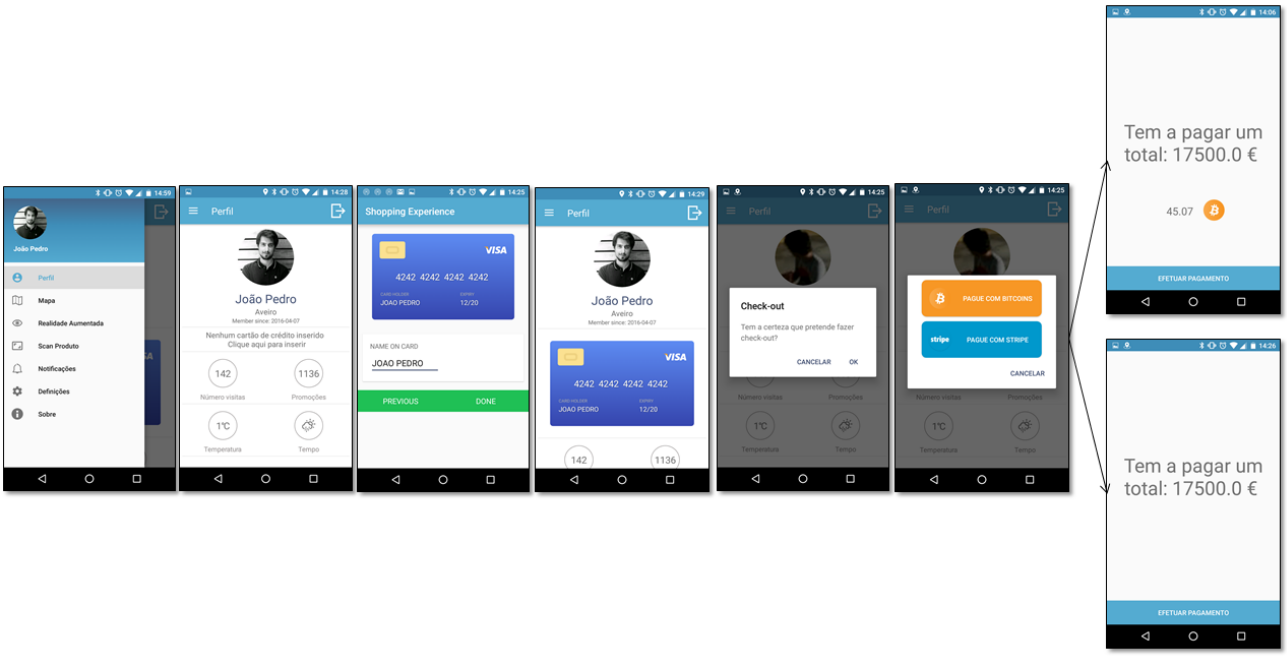



**Figura 50 - Ecrãs associados à secção do perfil e check-out**

### 5.5.4 Mapa

Nesta secção é possível visualizar o mapa com a posição atual do cliente indicada através dum ponto azul, e no canto inferior direito é possível efetuar uma pesquisa por produtos. Ao abrir esse produto é possível ver a sua localização na planta da loja. À medida que se vai andando pela loja o ponto azul vai acompanhando o nosso processo e vai atualizando e pondo a par constantemente da localização atual do utilizador, pode-se fazer *zoom in* e *zoom out* na imagem para ver mais em pormenor a planta da loja (este é um dos motivos pelos quais a imagem demora algum tempo a carregar).

Além de ser possível visualizar a localização atual na loja, neste ecrã podemos ainda efetuar pesquisa de produtos. Clicando no botão que se encontra no canto inferior direito é aberto um *dialog* com uma lista de produtos, onde se pode filtrar ou selecionar um produto. Ao selecionar um, o comportamento é o mesmo já detalhado para este ecrã anteriormente, pode-se ver detalhes do produto, partilhar o produto nas redes sociais e ver este produto através de realidade aumentada ou a sua localização na loja atual.

Ao escolher a opção de visualizar o produto no mapa aparece um marcador a mostrar a localização do produto na loja (como é possível ver no último ecrã da imagem abaixo). Graças a esta funcionalidade pode-se ver qual o melhor caminho a seguir quando queremos ir comprar aquele produto.

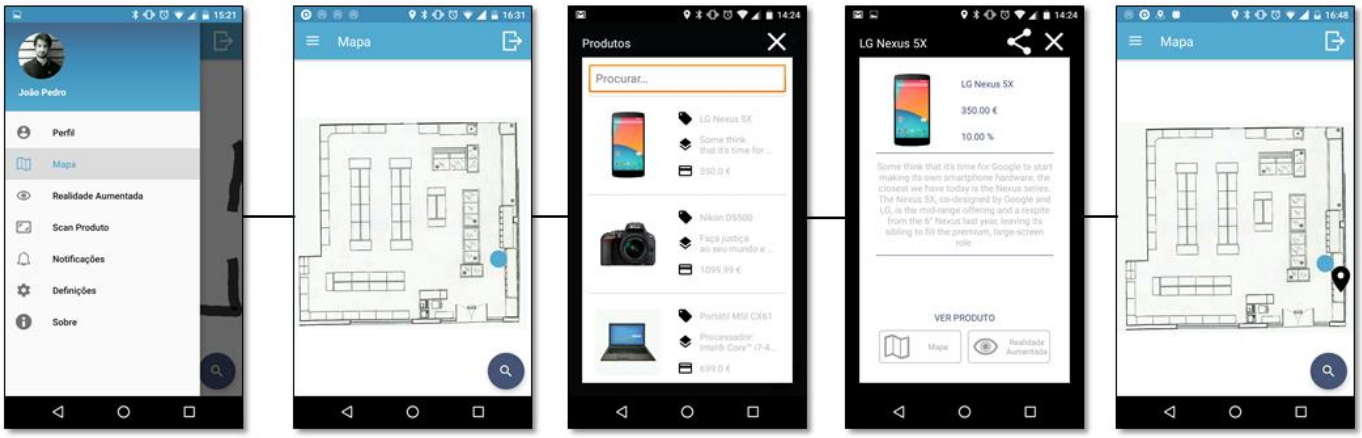



**Figura 51 - Ecrãs associados à secção do mapa**

### 5.5.5 Realidade Aumentada (Geolocalização)

Nesta secção é possível ver os produtos que existem disponíveis na loja através de realidade aumenta. Os produtos aparecem distribuídos consoante a sua localização no mundo real, e a sua posição encontra-se visível num radar que se encontra no canto superior direito, para assim o utilizador saber por onde deve apontar a câmara para visualizar o produto. Ao carregar em cima do produto aparece um painel com informação detalha acerca desse produto, com esta funcionalidade o utilizador ganha uma nova maneira de puder ver e encontrar os produtos na loja.

Além desta funcionalidade, também no canto superior direito existe uma opção com definições. Este painel foi criado visto que cada telemóvel contem sensores diferentes, uns são mais sensíveis ao movimento que outros, e com esta funcionalidade o utilizador consegue regular a experiência consoante o seu telemóvel. As opções disponíveis neste painel são:

- **Push Away** – controla a distância dos objetos, consegue aproximá-los ou afastá-los consoante o gosto do utilizador.

- **Pull Closer** – permite controlar a distância dos objetos no radar

- **Low Pass Filter** – controla a sensibilidade com que o utilizador "gira" o telemóvel, e essa mesma sensibilidade influencia a forma como os objetos vão aparecendo.

- **Max distance** – controla a máxima distância que queremos que os objetos aparecem.

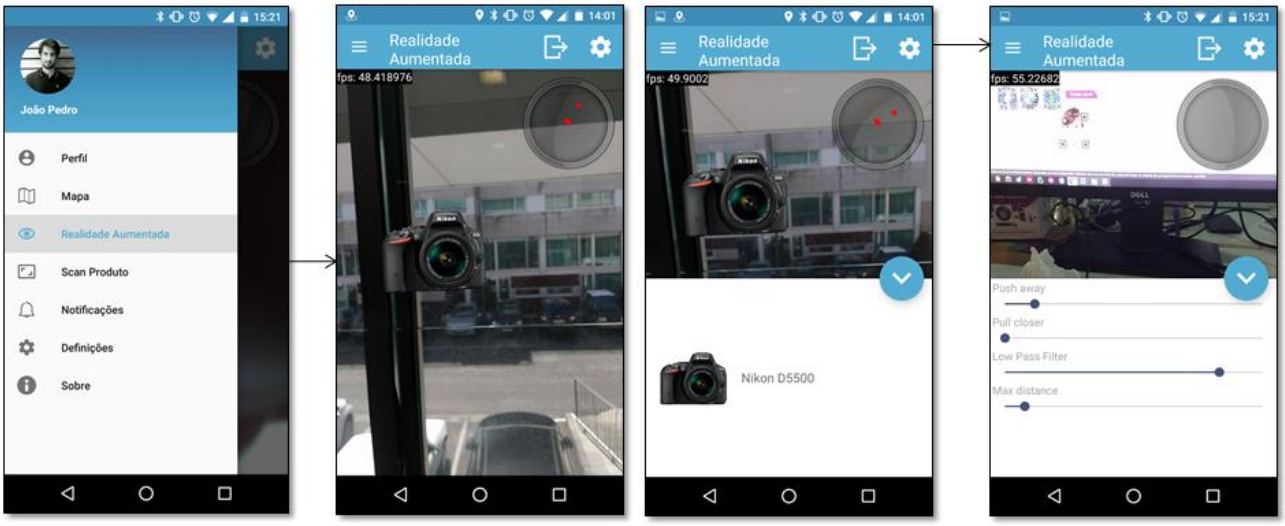



**Figura 52 - Ecrãs associados à realidade aumentada através de geolocalização**

### 5.5.6 Realidade Aumentada (Image Recognition) e Scan

Nesta secção é possível fazer o scan de produtos através de NFC e QRCODE mas também através da *image recognition*. O utilizador nesta secção para visualizar informação detalhadas acerca de um produto basta passar o telemóvel por cima da TAG NFC associada ao mesmo ou ao seu QRCode, e posteriormente irá aparecer um *dialog* com informação acerca desse produto. Nesse *dialog* pode-se efetuar várias opções como ver a posição desse objeto através do mapa (mostra a localização em relação à posição do produto) ou ver esse produto através da realidade aumentada detalhada anteriormente. Pode-se também partilhar a informação acerca daquele produto nas redes socias.

Além disso na parte de *image recognition* que também já foi falada anteriormente é possível ver outras características associadas ao produto (imagens, vídeos) simplesmente apontando a câmara para a embalagem do produto. Ao passar a imagem em cima da embalagem irá aparecer uma imagem ou vídeo em que o utilizador pode reproduzir e assim ver como por exemplo o processo de fabrico desse produto, o modo de utilização entre muitas outras vertentes.

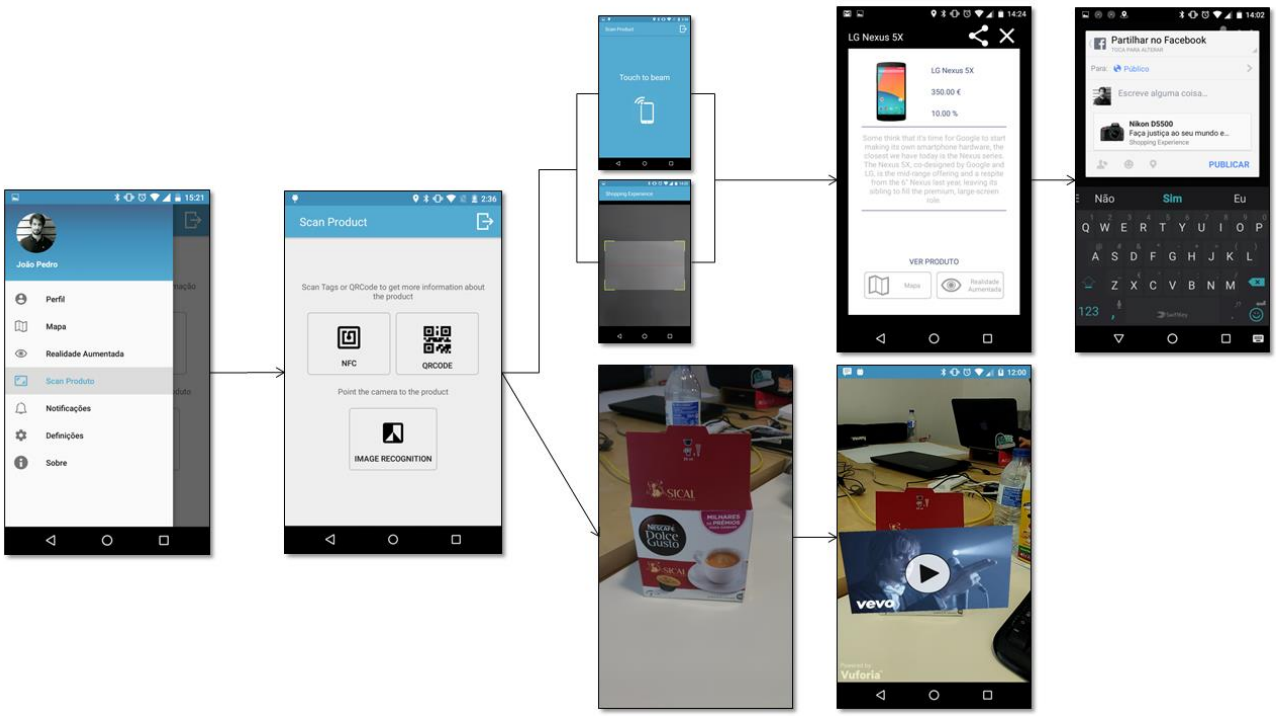



**Figura 53 - Ecrãs associados ao scan de produtos através de NFC, QRCode e image *recognition***

### 5.5.7 Notificações

Neste tópico detalha-se a parte das notificações, a aplicação recebe notificações de dois tipos: *beacons* e *geofencing*. O que difere estas notificações de outras que se recebe por norma, é o contexto em que as notificações são enviadas, ou seja, estas notificações são enviadas quando um utilizador chega perto de uma loja através do *geofencing* e quando o utilizador percorre os corredores duma loja e vai recebendo notificações de promoções naquele corredor graças aos *beacons*.

Todas as notificações recebidas são armazenadas no *backend* e na própria aplicação pois a sua abertura ou não, se compra o produto do qual recebeu a notificação entre muitas outras informações são bastante valiosas para conhecer um pouco mais o padrão de compra dos consumidores. A cada notificação vai também um produto associado, e assim sendo, duma forma rápida e eficaz ver a que produto está associado a promoção.

Através destas tecnologias consegue-se enviar ao consumidor notificações com contexto, ou seja, notificações que o cliente quer receber naquele momento. E no futuro poder também enviar promoções direcionadas simplesmente a um utilizador, pois já se vai conhecer melhor o seu perfil de compra, e saber que se se enviar um certo tipo de promoção a probabilidade de ele comprar o produto a ela associada é bastante grande.

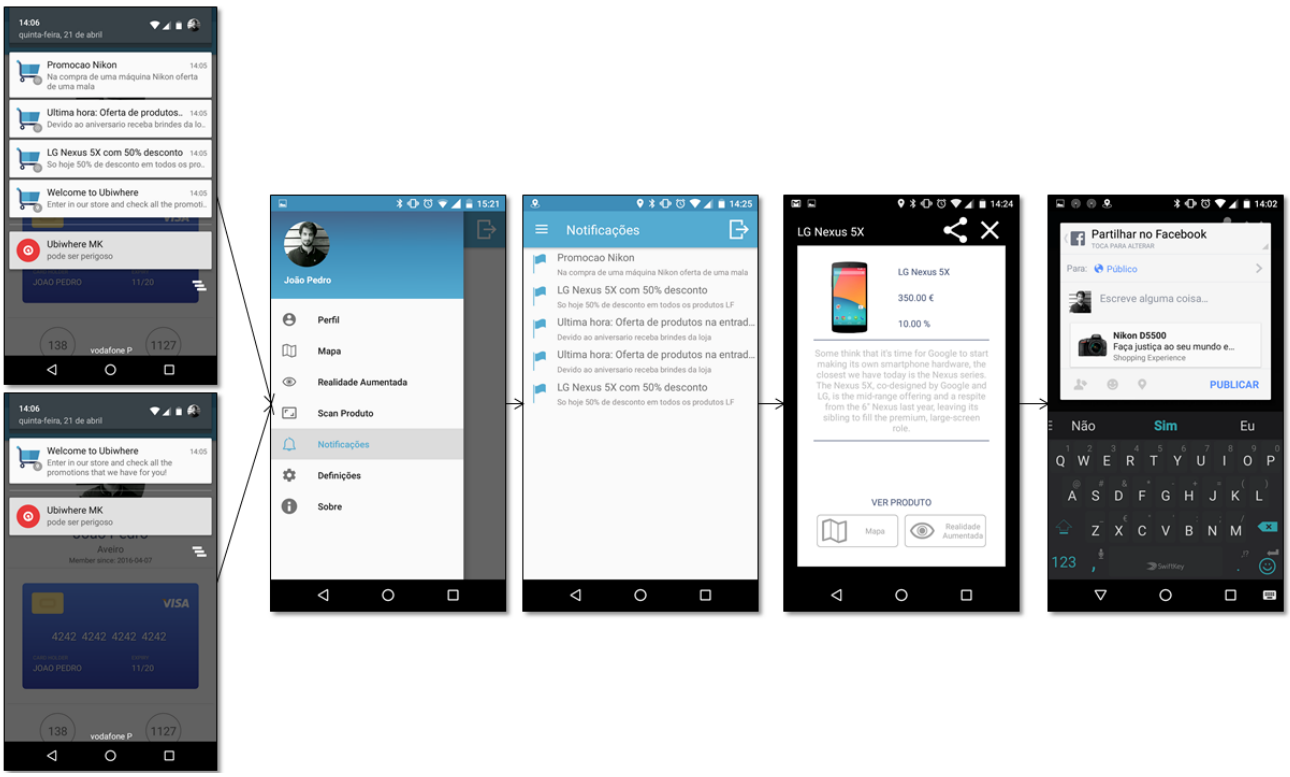



**Figura 54 - Ecrãs associados às notificações**

# 6 Conclusão

Os *smartphones* vão sem duvida continuar a revolucionar a forma como agimos e vivemos os nosso dia-a-dia, visto serem uma ferramenta, neste momento, com as mesmas capacidades que um computador e que andas sempre connosco, o que faz as sua funcionalidades ínfimas. A área do mobile teve um crescimento bastante acentuado nos últimos anos, e prevê-se que cada vez mais este mercado continue a crescer e a revolucionar o mundo como hoje o conhecemos. Com a constante evolução da tecnologia os *smartphones* terão mais e distintas funcionalidades, se hoje em dia já conseguimos graças à *localion aware*, Bluetooth, NFC, wireless entre muitas outras tecnologias criar aplicações revolucionárias que mudam o nosso dia-a-dia, no futuro ainda mais, por isso é que esta é uma área de bastante interesse da minha parte, e tento estar sempre a par de novas tecnologias e bibliotecas nesta área.

Na área do desenvolvimento, especialmente direcionado para Android, já existem muitas e boas ferramentas para auxiliar à criação e integração destas novas "tecnologias do futuro", muitas delas estão neste momento a ter a sua expansão ao nível do conhecimento e utilização (como é o caso da realidade aumentada), contudo há algumas que ainda são bastante recentes e ainda pouco exploradas e conhecidas pela parte dos clientes e que podem ser uma mais-valia para o mesmo (como é o caso do geofencing e dos beacons).

O principal objetivo a alcançar era a criação de uma aplicação que reunisse o melhor dos dois mundos para o sector do retalho (nova experiência de compra para o cliente e melhor conhecimento do seu cliente para o retalhista), fazendo isto sem muitos custos ao nível de hardware e integrando tecnologias inovadoras, e isto tudo utilizando somente o *smartphone* do cliente. Como foi explicado ao longo deste documento, a criação duma aplicação com tantas tecnologias distintas nem sempre é fácil e cria muitos problemas, pois é necessário ter em conta a usabilidade por parte do utilizador, a fluidez da aplicação e o design que deve ser apelativo e minimalista. Em termos de usabilidade de design (mesmo não sendo o foco) foi tida sempre uma sensibilidade a este nível, pois hoje em dia uma aplicação (mesmo não sendo uma aplicação "final") que não tenha um design e uma usabilidade boa para o utilizador, mesmo que as ferramentas sejam boas e funcionais o cliente vai deixar de utilizar a mesma.

Concluindo, acho que o facto de esta tese ter sido desenvolvida num âmbito empresarial foi um mais-valia, pois quando surgia alguma dúvida ao nível do desenvolvimento existia sempre alguém com conhecimentos suficientes para auxiliar, e também porque desenvolvendo uma aplicação destas num âmbito empresarial temos sempre a sensibilidade de adaptar a mesma tanto para o consumidor final como para um cliente intermédio, e assim estruturar a aplicação para estar ser adaptável e escalável para vários clientes distintos. A nível pessoal evolui bastante ao nível de experiencia tanto de programar e arquitetura de uma aplicação Android, como desenvolver e integrar um *backend* para a mesma, explorei algumas áreas que ainda tinha bastante lacunas ao nível do desenvolvimento Android, e a experimentação de várias tecnologias e ferramentas distintas, foram fundamentais pois ganhei bastante sensibilidade nas vantagens e desvantagens no uso de cada uma das tecnologias.

## 6.1 Análise de Dados

Um dos objetivos futuros deste trabalho é ser possível analisar as métricas que vão sendo recolhidas e a partir delas conseguir reconhecer alguns padrões relacionados com os clientes duma loja, sobretudo o padrão de compra. Apesar de não ser um dos objetivos nesta tese, é preciso ter ideia de que dados recolher e que informações esses dados podem fornecer, por isso foi necessário pensar nesse cenário no início de desenvolvimento tanto do *backend* como da aplicação. Maior parte dos artigos que foram lidos na fase prévia à execução referiam um número de métricas como as base a analisar, isto é, métricas que todas as empresas que já forneciam este tipo de serviços recolhiam, entre elas estão os dados relativos ao tempo, recolha das redes wireless ao alcance do utilizador, entre outras. Nesta secção vai ser detalhado como e para que podem ser usados os dados que estão a ser recolhidos e além disso mostrar alguns dos dados recolhidos através do **Power BI** [94] da Azure [94].

- **Sensores** – como já foi referido em cima graças ao *SensorDeviceManager* pode-se recolher informação relativa a vários sensores, e cada um nos podem dar métricas importantes. Um exemplo disto é por exemplo o sensor de orientação que envia informação quase a cada segundo, e vai armazenando a orientação em que o telemóvel se encontra (norte, sul, este ou oeste), graças a esta informação como sabemos a hora de check-in e check-out e se o utilizador se encontra em movimento ou não somos capazes de saber os caminhos mais percorridos pelos utilizadores (esta informação também nos pode ser fornecida através da *dashboard* do IndoorAtlas).
- **Redes Wireless** – estão a ser armazenadas entre as redes ao alcance dos utilizadores a força do seu sinal, através dum padrão, e se a loja tiver mais que uma entrada, podemos ver qual a entrada mais utilizada pelos clientes (esta será a entrada onde devem ser expostos produtos em promoção, pois a probabilidade de compra é maior devido ao maior tráfego de pessoas).
- **Informação Social Networks** – os dados recolhidos através das redes sociais servem para conhecer os clientes mais ao nível dos gostos, clientes com os mesmos gostos podem ter tendência a adquirir os mesmos produtos.
- **Dados do tempo** – com os dados do tempo o dono da loja pode verificar se os dados climatéricos ou temperatura têm influência na ida dos clientes à loja.
- **Notificação –** armazenamento da hora em que a notificação foi enviada, para quem, se abriu ou não abriu a notificação, e se abriu a que hora o fez. Com esta informação descrita podemos comparar a taxa de conversão na compra de produtos através do envio de notificações de promoções.
- **Scan de produtos** – está a ser armazenada o produto digitalizado, o utilizador e a localização no mesmo. Graças a esta informação no futuro podemos ver se a taxa de conversão dum produto por exemplo for inferior a 50% no fim do produto digitalizado quer dizer que o utilizador viu alguma informação que não gostou nesse produto.

- **Check-in e check-out** – útil para verificar a hora de maior afluência de clientes à loja.

Como já foi então referido acima, juntaram-se os dados que foram sendo recolhidos ao longo da fase de desenvolvimento da aplicação e do caso de estudo e inseriu-se esse conjunto de dados no Power BI do Azure. O Power BI é uma ferramenta da Azure que transforma a informação em dados visuais que se possa analisar, e além disso como tem também o suporte de relações entre tabelas podemos observar relações entre os dados e tirar algum conhecimento e valor disso. Foram criadas duas páginas com 9 gráficos distintos onde cada um está responsável por apresentar informação de uma determinada área.

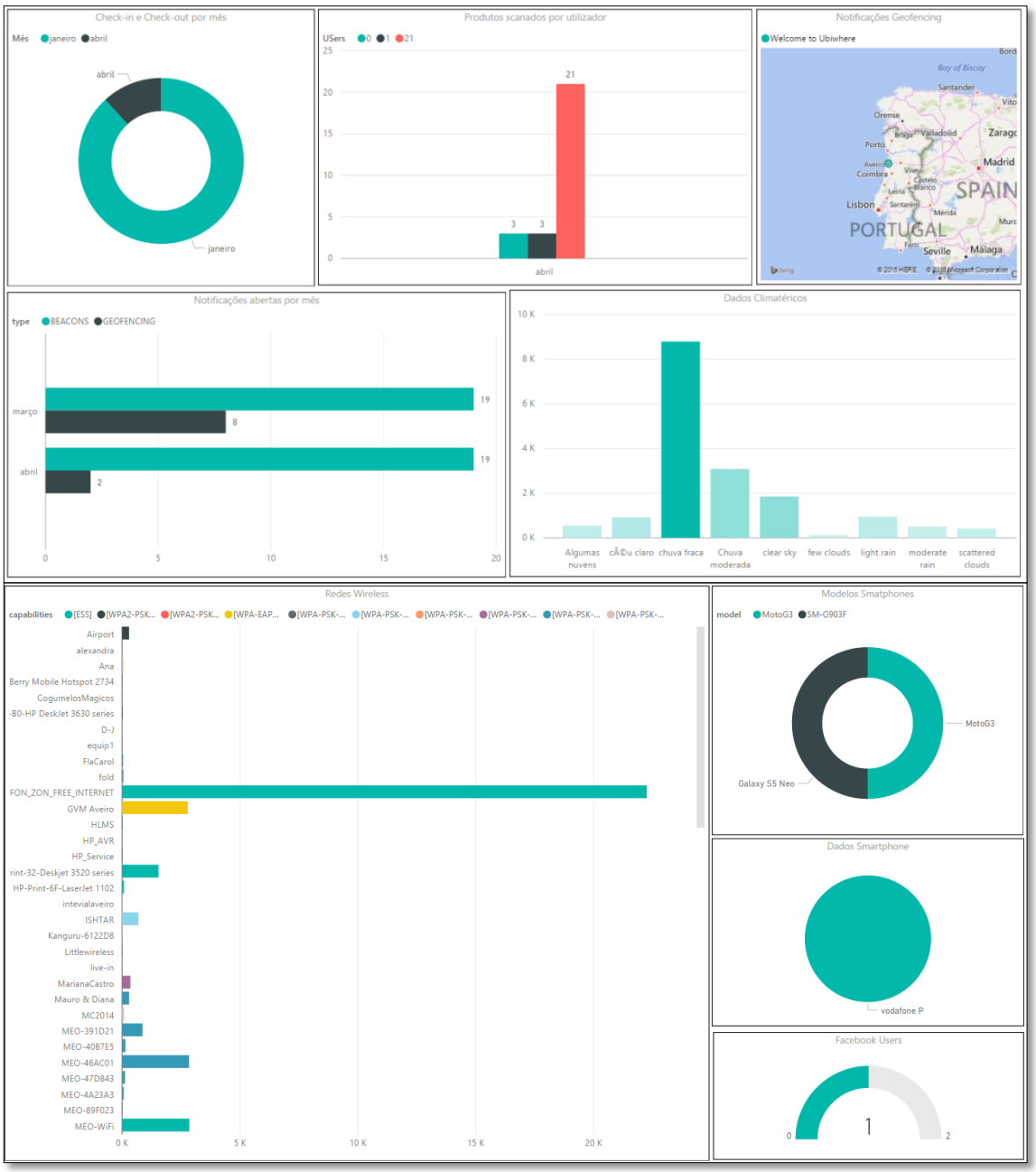



**Figura 55 - Análise dos dados no Power BI**

Na imagem acima pode-se ver a *dashboard* criada para análise dos dados relacionados com a aplicação, neste conjunto de gráficos temos informação relativa:

- **Redes Wireless** – onde é relacionado informação com o SSID, a força e ainda as *capabilities* da rede.
- **Informação do *smartphone*** – comparando o número de *smartphones* distintos e o modelo dos mesmos.
- **Rede** – dados de rede do *smartphone*, apresentando os vários operadores de rede.
- **Facebook** – o nº de utilizadores registados através desta rede social.
- **Client Flow** – onde comparamos o nº de check-in e check-out efetuados na loja por mês.
- **Produtos *digitalizados*** – onde vemos o nº de vezes que um produto foi digitalizado e por quantos utilizadores.
- **Geofencing** – onde podemos ver as promoções *geofencing* ativas num mapa.
- **Notificações** – o nº de notificações enviadas por mês e que tipo de notificação foi (*geofencing* ou *beacons*).
- **Dados climatéricos** – Informação relativa ao estado do tempo e temperatura.

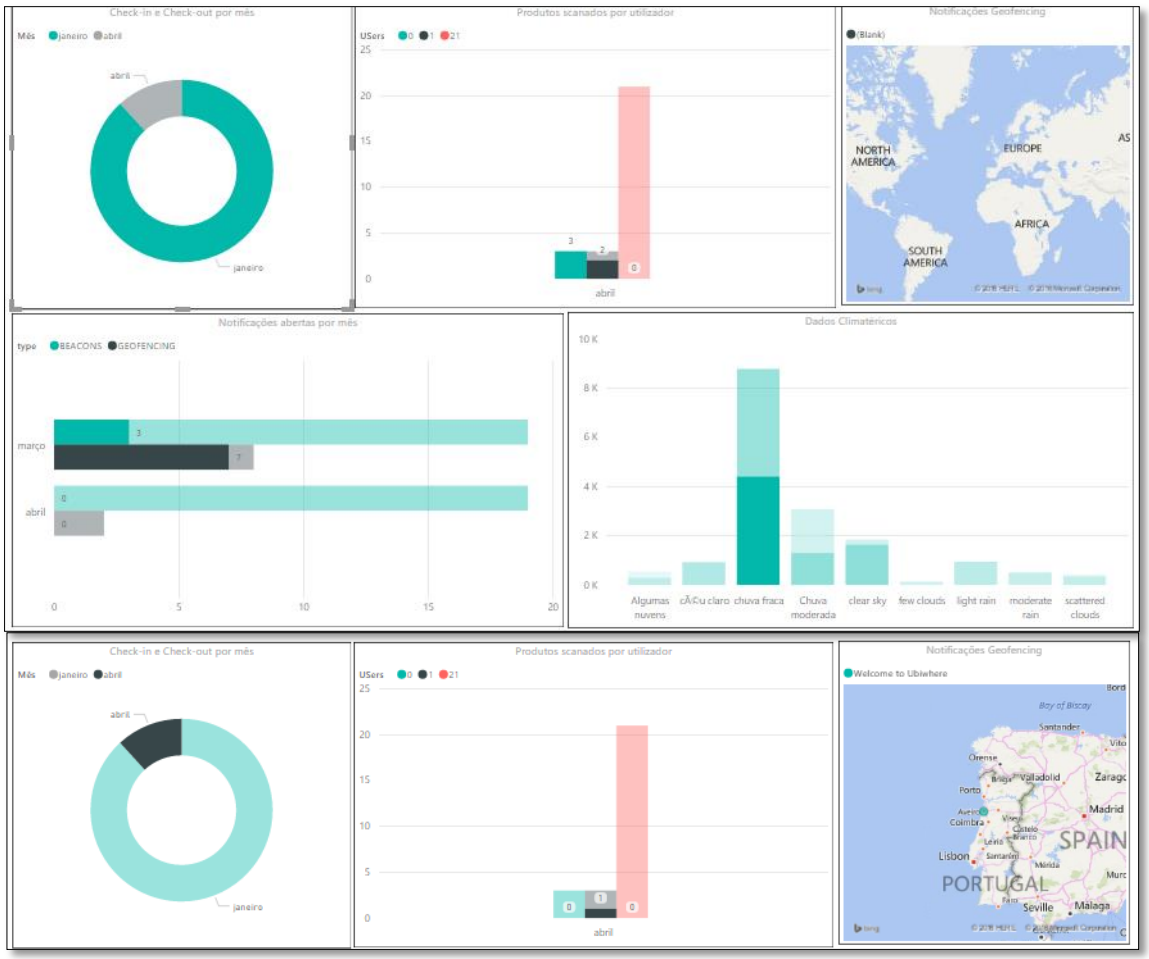



**Figura 56 - Relação entre os dados**

Na imagem acima é possível observar a relação entre os dados, pois na metade superior selecionamos os dados aos check-in e check-out do mês de janeiro e conseguimos ver o nº de produtos digitalizados, notificações enviadas e as alterações climatéricas que se verificaram nesse mês, mas também podemos ver que não tínhamos nenhuma promoção de *geofencing* associada. Já na parte inferior da imagem, e ao selecionar o mês de abril conseguimos ver que já temos um ponto no mapa associado a uma promoção *geofencing* podemos também ver a variação dos valores dos produtos digitalizados.

Além de ser possível ver a relação entre os gráficos, é também possível analisar cada gráfico individualmente e analisar os dados contidos em certos pontos dos gráficos ou tabelas.

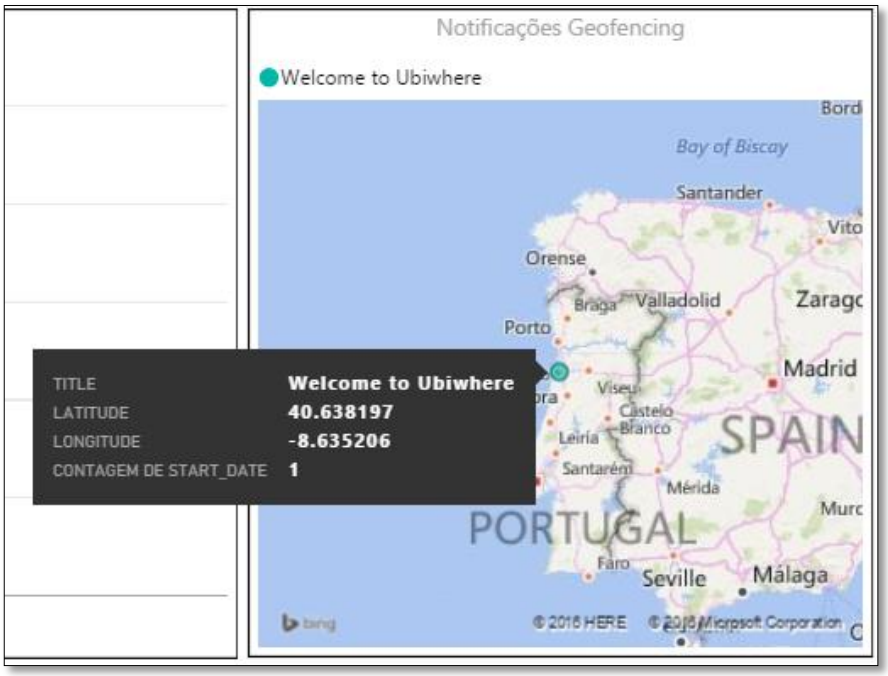



**Figura 57 - Detalhe do ponto duma promoção *geofencing* no mapa da *dashboard***

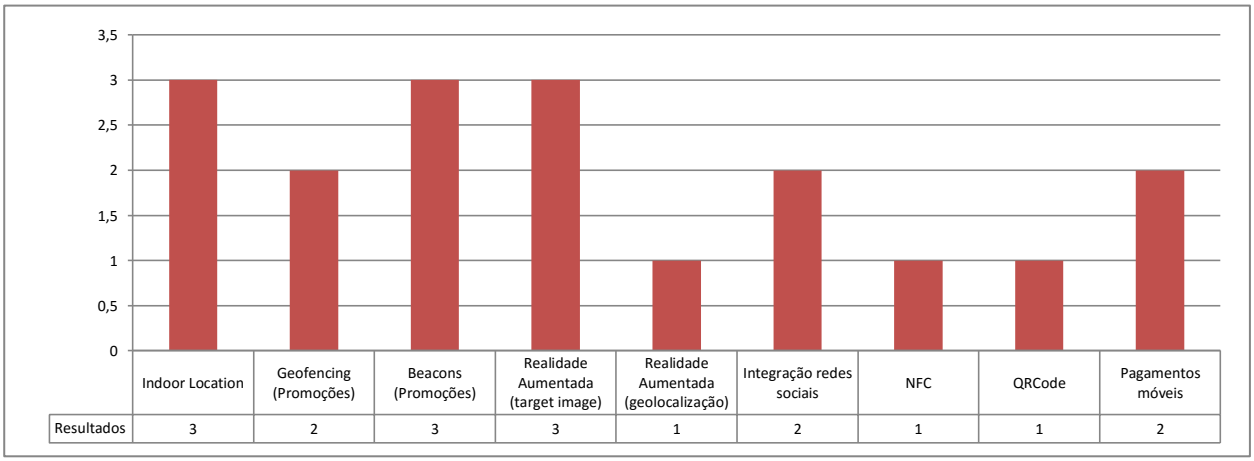
Como foi possível verificar com a integração com o Power BI é possível recolher alguma informação e mais-valia dos dados que foram e vão sendo recolhidos através da aplicação, contudo não é um sistema de *dataminig* preparado e especializado para a análise de dados deste tipo. Para ser possível recolher mesmo com relevância informação concreta dos dados é necessário no futuro desenvolver um processo de *datamining* especializado em torno dos dados recolhidos, que nos retorne as métricas e resultados que os donos da loja têm e querem saber.

## 6.2 Caso de Estudo

Para ser possível validar algumas das funcionalidades que foram desenvolvidos no projeto, no fim de desenvolvida a aplicação foi aplicada a um caso de estudo num contexto real. Este caso de estudo consiste em validar algumas das funcionalidades que foram criadas na aplicação e o conceito da nova experiência de compra criada para os clientes. Assim sendo, foi usado como caso de estudo um pequeno minimercado na zona de Pombal – Leiria. Este minimercado contém pelo menos 5 corredores dentro da loja, e o objetivo passa por um cliente conseguir saber onde se encontra na loja, ir recebendo notificações à medida que chega perto da loja e à medida que vai percorrendo a loja, e ainda conseguir fazer check-in e check-out duma forma rápida, eficiente e totalmente independente. O principal objetivo deste caso de uso seria validar algumas das funcionalidades que só conseguimos tirar o verdadeiro valor delas quando aplicadas num caso real, como é o caso do *geofencing*, *beacons* e *indoor location*, e assim sendo, são essas as principais funcionalidades que foram tentar ser validadas.

Para o caso de estudo ser mais realista e ser possível perceber melhor o que os utilizadores necessitam foram usadas 3 pessoas, com idades compreendidas entre os 19 e os 24 anos, nenhum deles da área das tecnologias (logo sem qualquer conhecimento prévio de maior parte das tecnologias) e por fim sem qualquer conhecimento prévio do funcionamento da aplicação.

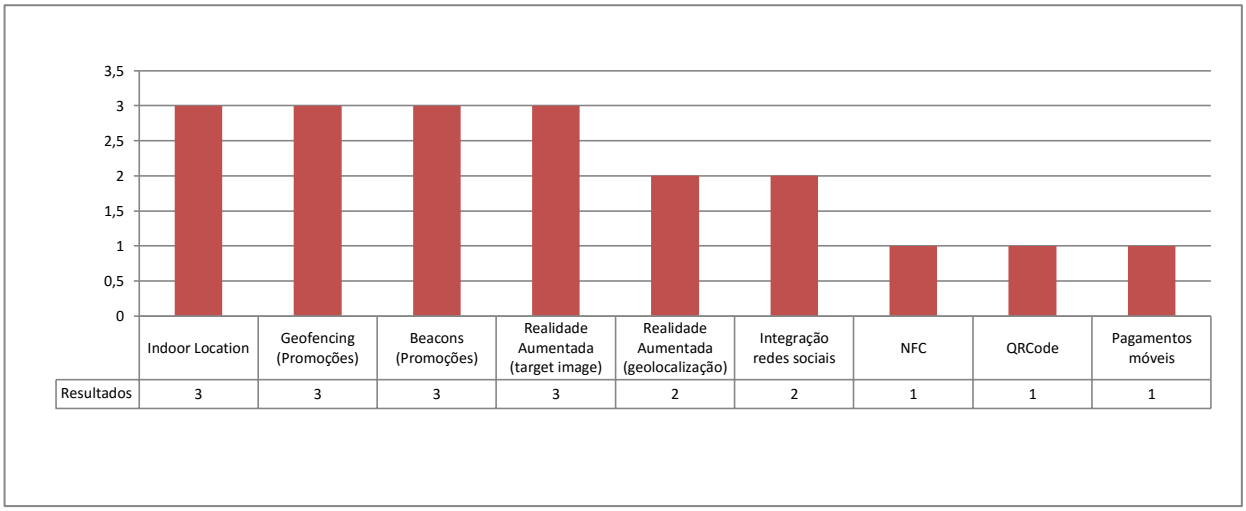
Neste caso de estudo específico foi pedido numa fase inicial para explorarem um pouco a aplicação, e posteriormente, efetuarem um conjunto de operações dentro da loja através da aplicação e de seguida responderem qual das tecnologias facilitava de alguma forma o processo de compra e por fim se as funcionalidades os poderiam atrair a ir uma loja física invés das compras *online*. No gráfico abaixo conseguimos ver as respostas à primeira pergunta colocada, a facilitação do processo de compra.




| | Indoor Location | Geofencing (Promoções) | Beacons (Promoções) | Realidade Aumentada (target image) | Realidade Aumentada (geolocalização) | Integração redes sociais | NFC | QRCode | Pagamentos móveis |
|---|---|---|---|---|---|---|---|---|---|
| Resultados | 3 | 2 | 3 | 3 | 1 | 2 | 1 | 1 | 2 |

**Figura 58 - Tabela relativa à facilitação do processo de compra das tecnologias**

O gráfico abaixo mostra os resultados obtidos quanto à tecnologia(s) que podem influenciar o utilizador a deslocar-se à loja física para efetuar as suas compras.




| | Indoor Location | Geofencing (Promoções) | Beacons (Promoções) | Realidade Aumentada (target image) | Realidade Aumentada (geolocalização) | Integração redes sociais | NFC | QRCode | Pagamentos móveis |
|---|---|---|---|---|---|---|---|---|---|
| Resultados | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 1 | 1 |

**Figura 59 - Tabela relativa à influência das tecnologias na ida à loja**

Pelos resultados obtidos é possível chegar à conclusão que as tecnologias "menos conhecidas" pelo público geral são as que têm uma melhor aceitação, e que promoções enviadas através de *location aware* causam um maior impacto do ponto de vista do cliente. É possível verificar também que Portugal os pagamentos móveis ainda têm um longo caminho a percorrer para convencer os clientes a usá-los diariamente. Mesmo sendo cada vez mais inovador e facilitando o processo de compra tanto ao nível da rapidez de processos como de controlo e gestão de gastos, o povo português continua a ser um povo desconfiado neste campo, e tudo o que não seja dinheiro, cartão de crédito, cheque ou pagamentos através de entidade referência, terá um grande trabalho no processo de aceitação em Portugal.

Graças a este estudo foi possível também verificar que existem tecnologias que decerto têm de se encontrar presentes na "loja do futuro" como é o caso da *indoor location* e das promoções através de *location aware*. Muitas das compras feitas por parte do cliente são feitas por impulso, e é neste campo que as promoções dadas com contexto têm a sua mais-valia, pois se o cliente tiver um incentivo para efetuar uma compra, a maior parte das vezes vai fazê-la seja por impulso ou necessidade.

Através do estudo que foi realizado foi possível recolher algumas métricas interessantes, contudo, o grupo de estudo foi muito reduzido, para ser possível ter mais relevância e veracidade nos dados obtidos é necessário fazer um caso de estudo numa superfície comercial maior e com um grupo de estudo maior.

## 6.3 Dificuldades

Ao longo do desenvolvimento da plataforma foram sentidas várias dificuldades que tiveram de ser ultrapassadas para ser possível chegar aos objetivos esperados no âmbito desta dissertação.

Uma das primeiras dificuldades sentidas foi na criação do *backend* e da API para o mesmo, visto que era a primeira vez que estava a mexer com aquelas tecnologias, tendo sido necessário passar pela curva de aprendizagem das tecnologias e *frameworks* necessárias.

Outra das dificuldades sentidas relaciona-se com a integração do SDK da Estimote, já que este estava com um problema que não conseguia acionar eventos relativos a todos os beacons, quando na documentação esse processo era possível. Assim, foi necessário entrar em contacto com a equipa da Estimote onde graças à troca de alguns emails, envio de código e de *samples* foi possível chegar à raiz do problema e solucioná-lo em tempo útil.

Um dos maiores problemas sentidos durante o desenvolvimento desta dissertação foi na parte de mostrar a imagem da planta para ser possível fazer a localização indoor. O problema é que a imagem precisava de vir dos servidores da Indoor Atlas, passar por uma biblioteca para ser possível rodar e fazer zoom na imagem e só por fim mostrá-la, e este processo levava a imagem levasse cerca de 20 segundos até ser mostrada, o que no campo do mobile é bastante tempo para um utilizador estar à espera. No início foi possível reduzir algum deste tempo armazenando a imagem localmente mas, mesmo assim ainda era bastante tempo que se demorava a carregar a mesma imagem. A solução encontrada foi reduzir o tamanho da imagem sem reduzir muito a sua qualidade e alojar novamente essa imagem nos servidores da Indoor Atlas, sendo agora possível mostrar a imagem com um ponto azul a mostrar a localização do utilizador em cerca de 5 segundos.

Por fim, outra das dificuldades sentidas foi na utilização do Data Binding, visto que é uma abordagem totalmente diferente daquelas que estava habituado a usar no desenvolvimento em Android. Um dos maiores problemas foi o fato de ser uma biblioteca lançada recentemente, e à medida que ia desenvolvendo deparava-me com algumas implementações que precisava de fazer e não existia apoio *online*, nem documentação específica ou comunidade para ajudar a resolver o mesmo, o que por vezes levou a que tivesse de implementar coisas por tentativa e erro. Contudo esse processo ajudou-me a compreender melhor o uso dessa biblioteca e como tirar o melhor proveito da mesma.

# 7 Trabalho Futuro

Um dos objetivos iniciais principais desta tese era desenvolver uma aplicação que recolhesse informação que no futuro pudesse ser usada para reconhecer padrões de compra dos utilizadores, por isso um dos trabalhos futuros a fazer é pegar em todos os dados recolhidos, processá-los e aplicar *datamining* sobre os mesmos de modo a que consigamos recolher algumas métricas de valor. Possivelmente a melhor solução será implementar um *dashboard* onde um dono duma loja possa fazer cruzamento de dados e descobrir informações novas e úteis sobre os hábitos e compras e ida à loja dos seus clientes. Contribuindo assim para também criar uma melhor experiência aos visitantes da loja.

Outros das tarefas a fazer no futuro é ligar o *backend* criado com uma base de dados de produtos duma loja, pois neste momento os produtos que se encontram no *backend* são "fictícios" e só para demonstrar o conceito. Se for possível ligar a uma base de dados de uma loja real todos o processo de pagamento poderá ser efetuado na realidade, pois temos acesso às características que cada produto e acima de tudo ao seu preço, imposto e desconto associado. Outro dos possíveis trabalhos futuros seria integrar o Firebase [95] em vez de criar um backend de raiz. O Firebase disponibiliza serviços como *realtime-database* que notifica em tempo real quando à alteração nos dados além dum poderoso conjunto de *analytics*, contudo estas funcionalidades todas só foram apresentadas no Google I/O 2016 o que inviabilizou a sua integração.

Além das propostas feitas acima, também uma das funcionalidades a usar será integrar os beacons para ajudar na localização indoor. Como foi já foi referido na secção dos beacons, se tivermos pelo menos 4 beacons conseguimos saber a nossa posição em relação aos mesmos, e assim integrando os beacons com um serviço de indoor location conseguimos melhorar bastante ao nível de precisão.

Um dos trabalhos futuros a desenvolver (que já foi também referido anteriormente) é a necessidade de fazer um novo caso de estudo numa superfície comercial maior e com um grupo de estudo mais numeroso, pois só assim é possível tirar mais valor dos dados obtidos nesse estudo, e perceber de melhor forma o que o consumidor final duma superfície comercial dessas precisa para facilitar o seu processo de compra e para ser atraído para a loja.

Por fim, como objetivo a implementar no futuro seria tentar integrar todas estas funcionalidades no resto dos *smartphones* no mercado, ou seja, iOS e Windows Phone. No iOS a integração não seria muito complicada pois tirando o geofencing e o NFC, praticamente todas as outras tecnologias usadas dispõem de SDK ou bibliotecas similares para integração com estes sistemas operativo. Contudo para o Windows Phone o caso já não é o mesmo, pois ao nível de SDK e bibliotecas não existem praticamente nenhumas, umas das soluções possíveis seria contruir um aplicação hibrida utilizando ferramentas como o Ionic [96] ou o React [97], contudo também íamos ter sempre dificuldades ao nível de integração com algumas funcionalidades.

# 8 Referências


[1] "Ubiwhere | Research and Innovation | Idea to Product | User-centered Solutions." [Online]. Available: http://www.ubiwhere.com/en/. [Accessed: 29-Jan-2016].

[2] Platt Retail Institute Platt, "The Future of Retail : A Perspective on Emerging Technology and Store Formats," no. February, 2014.

[3] E. COMMISSION, "Final Report from the Expert Group on Retail Sector Innovation," Brussels, 2013.

[4] "Aisle411 | Indoor Mapping Software | Indoor Maps App." [Online]. Available: http://aisle411.com/. [Accessed: 29-Jan-2016].

[5] "Toysrus.pt Home - Página Oficial da Toys'R'Us - Brinquedos, videojogos e muito mais." [Online]. Available: http://www.toysrus.pt/home/index.jsp?categoryId=4599081. [Accessed: 29-Jan-2016].

[6] "Welcome to Walgreens - Your Home for Prescriptions, Photos and Health Information." [Online]. Available: http://www.walgreens.com/. [Accessed: 29-Jan-2016].

[7] "Movvo, Inc." [Online]. Available: https://movvo.com/. [Accessed: 03-Dec-2015].

[8] SwiftIQ, "SwiftIQ." [Online]. Available: http://www.swiftiq.com/. [Accessed: 11-Dec-2015].

[9] "RetailNext | Comprehensive In-Store Analytics." [Online]. Available: http://retailnext.net/. [Accessed: 29-Jan-2016].

[10] "Xhockware." [Online]. Available: http://www.xhockware.com/. [Accessed: 29-Jan-2016].

[11] M. R. Experience, R. Can, A. Being, and S. Away, "Riding the Tsunami :," 2012.

[12] "Deloitte Portugal | Audit, Consulting, Financial Advisory, Risk Management & Tax services and reports | Global." [Online]. Available: http://www2.deloitte.com/pt/pt.html. [Accessed: 29-Jan-2016].

[13] Mike Brinker, K. Lobaugh, and A. Paul, "The dawn of mobile influence," p. 14, 2012.

[14] "Android." [Online]. Available: https://www.android.com/. [Accessed: 29-Jan-2016].

[15] "iOS 9 - Apple (PT)." [Online]. Available: http://www.apple.com/pt/ios/. [Accessed: 29-Jan-2016].

[16] "Windows Phones - Microsoft." [Online]. Available: https://www.microsoft.com/pt-pt/windows/phones. [Accessed: 29-Jan-2016].

[17] L. Hyvonen, A. Pinto, and J. Troelsen, "Near field communication," *US Pat. 8,212,735*, 2012.

[18] T. Wiechert, A. Schaller, and F. Thiesse, "Near Field Communication Use in Retail Stores : Effects on the Customer Shopping Process 2 Retail Applications for NFC," *Mob. und Ubiquitäre Informationssysteme - Entwicklung, Implementierung und Anwendung*, pp. 137–141, 2009.

[19] "TPG Rewards Inc." [Online]. Available: http://tpgrewards.com/. [Accessed: 29-Jan-2016].

[20] "CloudTags." [Online]. Available: http://www.cloudtags.com/. [Accessed: 29-Jan-2016].

[21] "Target : Expect More. Pay Less." [Online]. Available: http://intl.target.com/. [Accessed:

29-Jan-2016].

[22] “Swirl | Beacon-Powered Mobile Marketing at Scale.” [Online]. Available: http://www.swirl.com/. [Accessed: 29-Jan-2016].

[23] “THE STOREFRONT OF THE FUTURE : 5 RETAIL TECHNOLOGY,” pp. 1–3, 2015.

[24] “Estimote Beacons — real world context for your apps.” [Online]. Available: http://estimote.com/. [Accessed: 29-Jan-2016].

[25] Atos, “The Future of In-Store Shopping,” 2013.

[26] “Geofencing Notifications - Plot Projects.” [Online]. Available: http://www.plotprojects.com/. [Accessed: 29-Jan-2016].

[27] “L’Oréal Paris - Maquilhagem, Coloração, Produtos de beleza, Cosméticos, Cuidados com o corpo.” [Online]. Available: http://www.lorealparis.pt/_pt/_pt/home/index.aspx. [Accessed: 29-Jan-2016].

[28] “Starbucks Coffee Company.” [Online]. Available: http://www.starbucks.pt/. [Accessed: 29-Jan-2016].

[29] “Starbucks breaks first location-based mobile campaign with major carrier - Mobile Marketer - Database/CRM.” [Online]. Available: http://www.mobilemarketer.com/cms/news/database-crm/7754.html. [Accessed: 29-Jan-2016].

[30] Bitcoin Project, “Bitcoin,” 2009. [Online]. Available: https://bitcoin.org/en/. [Accessed: 03-Dec-2015].

[31] M. L. T. Cossio, L. F. Giesen, G. Araya, M. L. S. Pérez-Cotapos, R. L. VERGARA, M. Manca, R. A. Tohme, S. D. Holmberg, T. Bressmann, D. R. Lirio, J. S. Román, R. G. Solís, S. Thakur, S. N. Rao, E. L. Modelado, A. D. E. La, C. Durante, U. N. A. Tradición, M. En, E. L. Espejo, D. E. L. A. S. Fuentes, U. A. De Yucatán, C. M. Lenin, L. F. Cian, M. J. Douglas, L. Plata, and F. Héritier, “No Title No Title,” *Uma ética para quantos?*, vol. XXXIII, no. 2, pp. 81–87, 2012.

[32] G. PwC, “Total Retail 2015:,” no. February, 2015.

[33] “Android – Android Pay.” [Online]. Available: https://www.android.com/pay/. [Accessed: 16-May-2016].

[34] “Stripe.” [Online]. Available: https://stripe.com/. [Accessed: 14-Apr-2016].

[35] “Braintree Payments: Accept Online and Mobile Payments.” [Online]. Available: https://www.braintreepayments.com/. [Accessed: 14-Apr-2016].

[36] “Redpin - Indoor Positioning for the Rest of Us.” [Online]. Available: http://redpin.org/. [Accessed: 29-Jan-2016].

[37] “IndoorAtlas.” [Online]. Available: https://www.indooratlas.com/. [Accessed: 29-Jan-2016].

[38] “Meridian | Indoor GPS, iBeacon.” [Online]. Available: http://meridianapps.com/. [Accessed: 29-Jan-2016].

[39] Facebook, “Facebook,” 2016. [Online]. Available: https://www.facebook.com/. [Accessed: 03-Dec-2015].

[40] I. Twitter, “Twitter.” [Online]. Available: https://twitter.com/. [Accessed: 03-Dec-2015].

[41] W. Sands and R. Study, “Reinventing Retail : What Businesses Need to Know for 2014 Walker Sands ’ 2014 Future of Retail Study,” pp. 1–9, 2014.

[42] Pinterest, “Pinterest,” 2016. [Online]. Available: https://pt.pinterest.com/. [Accessed: 03-Dec-2015].

[43] P. A. KENGNE, “Mobile Augmented Reality,” Lahti University of Applied Sciences, 2014.

[44] “Aisle411 | Store Map Availability | 12,000+ Retail Stores.” [Online]. Available: http://aisle411.com/availability/. [Accessed: 29-Jan-2016].

[45] “Walgreens Tests Google’s Augmented Reality for Loyalty App | Data-Driven Marketing - AdAge.” [Online]. Available: http://adage.com/article/datadriven-marketing/walgreens-tests-google-s-augmented-reality-loyalty-app/293961/. [Accessed: 29-Jan-2016].

[46] O. Retail, “The Store of the Future Happening Now.”

[47] “Augmented Reality | Interactive Print | Layar.” [Online]. Available: https://www.layar.com/. [Accessed: 29-Jan-2016].

[48] “Vuforia Developer Portal.” [Online]. Available: https://developer.vuforia.com/. [Accessed: 29-Jan-2016].

[49] “Wikitude - The World’s leading Augmented Reality SDK.” [Online]. Available: http://www.wikitude.com/. [Accessed: 29-Jan-2016].

[50] “Home | BeyondAR.” [Online]. Available: http://beyondar.com/. [Accessed: 29-Jan-2016].

[51] “funf | Open Sensing Framework.” [Online]. Available: http://funf.org/. [Accessed: 29-Jan-2016].

[52] “Installing the Android SDK | Android Developers.” [Online]. Available: http://developer.android.com/sdk/installing/index.html. [Accessed: 29-Jan-2016].

[53] “Empresa – Google.” [Online]. Available: https://www.google.com/about/company/. [Accessed: 29-Jan-2016].

[54] “Download Android Studio and SDK Tools | Android Developers.” [Online]. Available: http://developer.android.com/sdk/index.html. [Accessed: 29-Jan-2016].

[55] “Gradle l Modern Open-Source Enterprise Build Automation - Gradle.” [Online]. Available: http://gradle.org/. [Accessed: 29-Jan-2016].

[56] Facebook, “Facebook SDK for Android,” 2016. [Online]. Available: https://developers.facebook.com/docs/android. [Accessed: 14-Feb-2016].

[57] “Fabric - Twitter’s Mobile Development Platform.” [Online]. Available: https://get.fabric.io/kits?locale=pt. [Accessed: 29-Jan-2016].

[58] “OpenWeatherMap current weather and forecast.” [Online]. Available: http://openweathermap.org/. [Accessed: 29-Jan-2016].

[59] “Serviços do Google Play – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.gms&hl=pt_PT. [Accessed: 29-Jan-2016].

[60] “Android NDK | Android Developers.” [Online]. Available: http://developer.android.com/tools/sdk/ndk/index.html. [Accessed: 29-Jan-2016].

[61] “SensorManager | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/hardware/SensorManager.html. [Accessed: 29-Jan-2016].

[62] “Download Python | Python.org.” [Online]. Available: https://www.python.org/downloads/. [Accessed: 29-Jan-2016].

[63] “The Web framework for perfectionists with deadlines | Django.” [Online]. Available: https://www.djangoproject.com/. [Accessed: 29-Jan-2016].

[64] “Django REST framework.” [Online]. Available: http://www.django-rest-framework.org/. [Accessed: 29-Jan-2016].

[65] “PostGIS — Spatial and Geographic Objects for PostgreSQL.” [Online]. Available: http://postgis.net/. [Accessed: 29-Jan-2016].

[66] “SQLite Home Page.” [Online]. Available: https://www.sqlite.org/. [Accessed: 29-Jan-2016].

[67] “Swagger | The World’s Most Popular Framework for APIs.” [Online]. Available: http://swagger.io/. [Accessed: 29-Jan-2016].

[68] “Dagger ‡ A fast dependency injector for Android and Java.” [Online]. Available: http://google.github.io/dagger/. [Accessed: 13-Apr-2016].

[69] “Data Binding Guide | Android Developers.” [Online]. Available: http://developer.android.com/tools/data-binding/guide.html. [Accessed: 13-Apr-2016].

[70] “Realm is a mobile database: a replacement for SQLite & Core Data.” [Online]. Available: https://realm.io/. [Accessed: 13-Apr-2016].

[71] “Retrofit.” [Online]. Available: http://square.github.io/retrofit/. [Accessed: 26-Apr-2016].

[72] “OkHttp.” [Online]. Available: http://square.github.io/okhttp/. [Accessed: 26-Apr-2016].

[73] “Gson (Gson 2.3.1 API).” [Online]. Available: https://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/com/google/gson/Gson.html. [Accessed: 26-Apr-2016].

[74] “Context | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/content/Context.html. [Accessed: 26-Apr-2016].

[75] “PackageManager | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/content/pm/PackageManager.html. [Accessed: 26-Apr-2016].

[76] “TelephonyManager | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/telephony/TelephonyManager.html. [Accessed: 26-Apr-2016].

[77] “GsmCellLocation | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/telephony/gsm/GsmCellLocation.html. [Accessed: 26-Apr-2016].

[78] “466.53 USD | BitcoinAverage Price Index.” [Online]. Available: https://bitcoinaverage.com/#USD. [Accessed: 26-Apr-2016].

[79] “PyCharm.” [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed: 26-Apr-

2016].

[80] “Trello.” [Online]. Available: https://trello.com/. [Accessed: 26-Apr-2016].

[81] “Bitbucket.” [Online]. Available: https://bitbucket.org/. [Accessed: 26-Apr-2016].

[82] “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 26-Apr-2016].

[83] Facebook, “Facebook Graph API,” 2016. [Online]. Available: https://developers.facebook.com/docs/graph-api. [Accessed: 15-Feb-2016].

[84] Facebook, “Facebook Developers,” 2016. [Online]. Available: https://developers.facebook.com/. [Accessed: 16-Dec-2015].

[85] “The most powerful, yet lightest weight crash reporting solution for iOS and Android developers. | Crashlytics.” [Online]. Available: https://try.crashlytics.com/. [Accessed: 26-Apr-2016].

[86] “IndoorAtlas MapCreator – Aplicações Android no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.indooratlas.mapcreator.main&hl=pt-PT. [Accessed: 19-Apr-2016].

[87] “Part 1 - Creating A Fragment - Xamarin.” [Online]. Available: https://developer.xamarin.com/guides/android/platform_features/fragments/part_1_-_creating_a_fragment/. [Accessed: 19-Apr-2016].

[88] “Android Integration.” [Online]. Available: https://stripe.com/docs/mobile/android. [Accessed: 26-Apr-2016].

[89] “App Manifest | Android Developers.” [Online]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html. [Accessed: 26-Apr-2016].

[90] “Building Apps with Over 64K Methods | Android Developers.” [Online]. Available: http://developer.android.com/tools/building/multidex.html. [Accessed: 21-Apr-2016].

[91] “Making Your App Location-Aware | Android Developers.” [Online]. Available: http://developer.android.com/training/location/index.html. [Accessed: 26-Apr-2016].

[92] “GLSurfaceView | Android Developers.” [Online]. Available: http://developer.android.com/reference/android/opengl/GLSurfaceView.html. [Accessed: 21-Apr-2016].

[93] GitHub, “GitHub.” [Online]. Available: https://github.com/. [Accessed: 10-May-2016].

[94] “Azure and Power BI | Microsoft Power BI.” [Online]. Available: https://powerbi.microsoft.com/en-us/documentation/powerbi-azure-and-power-bi/. [Accessed: 26-Apr-2016].

[95] “Firebase - Login to legacy console.” [Online]. Available: https://www.firebase.com/. [Accessed: 31-May-2016].

[96] “Ionic: Advanced HTML5 Hybrid Mobile App Framework.” [Online]. Available: http://ionicframework.com/. [Accessed: 26-Apr-2016].

[97] “A JavaScript library for building user interfaces | React.” [Online]. Available: https://facebook.github.io/react/. [Accessed: 26-Apr-2016].

[98] “Gartner Predicts a Promising Future for Windows Phone 7 Developers.” [Online].

Available: http://blog.scottlogic.com/2011/04/08/gartner-predicts-a-promising-future-for-windows-phone-7-developers.html. [Accessed: 29-Jan-2016].

[99] "New NXP tags enable simple Wi-Fi and Bluetooth pairing via NFC | IT Eco Map & News Navigator." [Online]. Available: http://itersnews.com/?p=44302. [Accessed: 29-Jan-2016].

[100] "iBeacons and the future of retail shopping: Consumers are ready, but are retailers? | VentureBeat | Mobile | by Meghan Kelly." [Online]. Available: http://venturebeat.com/2014/01/15/ibeacon-retailers/. [Accessed: 29-Jan-2016].

[101] "Reality matters — Introducing the Estimote Indoor Location SDK, the..." [Online]. Available: http://blog.estimote.com/post/98316374485/introducing-the-estimote-indoor-location-sdk-the. [Accessed: 01-Dec-2015].

[102] "Creating and Monitoring Geofences | Android Developers." [Online]. Available: http://developer.android.com/training/location/geofencing.html. [Accessed: 29-Jan-2016].

[103] "Stripe Would Be Perfect If... - Liam Kaufman." [Online]. Available: http://liamkaufman.com/blog/2012/11/09/stripe-would-be-perfect-if/. [Accessed: 15-Apr-2016].

[104] I. . Sterling, Greg (Opus Research, "Magnetic Positioning," pp. 1 – 8, 2014.

[105] I. Ltd, "Ambient magnetic field-based indoor location technology," no. July, 2012.

[106] D. Sheet, K. E. Y. Features, O. F. The, M. Platform, and T. H. E. M. Editor, "MERIDIAN MOBILE APP."

[107] "It is Alive in the Lab: 24 posts from February 2013." [Online]. Available: http://labs.blogs.com/its_alive_in_the_lab/2013/02/. [Accessed: 29-Jan-2016].

[108] "Augmented reality: the darling of the ad industry | 12Ahead." [Online]. Available: http://www.12ahead.com/augmented-reality-darling-ad-industry. [Accessed: 29-Jan-2016].

[109] "QR Codes in Layar? Yes! | Layar Blog." [Online]. Available: https://www.layar.com/news/blog/2013/03/05/qr-codes-in-layar-yes/. [Accessed: 29-Jan-2016].

[110] "Tasting Dagger 2 on Android – Fernando Cejas." [Online]. Available: http://fernandocejas.com/2015/04/11/tasting-dagger-2-on-android/. [Accessed: 13-Apr-2016].

[111] "Approaching Android with MVVM — ribot labs." [Online]. Available: https://labs.ribot.co.uk/approaching-android-with-mvvm-8ceec02d5442#.3vaoec6yq. [Accessed: 14-Apr-2016].