Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

16811 – Pedro André Casimiro Nunes

MOVTOUR

Trabalho Final de Mestrado

Orientado por:

Professor: **Luís Miguel Lopes de Oliveira**

Professor: **Renato Eduardo Silva Panda**

Trabalho final de mestrado apresentado ao Instituto Politécnico de Tomar
para cumprimento dos requisitos necessários
à obtenção do grau de Mestre em Engenharia Informática – Internet das Coisas

Resumo

A utilização de aplicações móveis para promoção do turismo é uma das tendências atuais. Para além da informação útil disponibilizada ao visitante, este tipo de aplicações pode também ser utilizada para recolher informação sobre os mesmos. Das várias aplicações móveis existentes para a vertente do turismo verificou-se que, ou são restritas a uma região ou monumentos específicos, ou fornecem uma espécie de páginas amarelas para várias cidades, estando por norma apenas disponíveis para cidades de grandes dimensões. De forma a oferecer ao utilizador uma experiência mais personalizada, algumas destas utilizam o sistema de GPS do dispositivo. No entanto, esta solução tem alguns inconvenientes, tais como o consumo excessivo de bateria, a falta de precisão ou até de cobertura entre e dentro de edifícios. Por outro lado, do que se pode verificar, não existe nenhuma aplicação que alerte e informe o visitante quando este se aproxima de um ponto de interesse. Este projeto tem como objetivo atacar as limitações identificadas, para tal propondo um sistema que permita ao turista receber automaticamente informação personalizada sobre os pontos de interesse que visita e ao mesmo tempo que forneça informação sobre as tendências dos visitantes.

Este projeto consiste no desenvolvimento de um sistema para a gestão de pontos de interesse e recolha de informação sobre os visitantes, o MovTour. Para tal foi criado um *backoffice*, com a função de gerir e registar a informação turística, e uma aplicação móvel a ser utilizada pelo turista, que interage com *beacons* (pequenos sensores) colocados nos pontos de interesse, através de Bluetooth. Após encontrar um *beacon*, a aplicação móvel notifica o *backoffice* sobre a visita e apresenta informação relevante ao visitante. O *backoffice* encontra-se em funcionamento, estando alojado num servidor do Instituto Politécnico de Tomar. A aplicação móvel foi desenvolvida tanto para dispositivos Android como para Apple iOS, estando a versão Android já disponível no Google Play. Foram também iniciados testes de captura de sinais dos *beacons* no Convento de Cristo, em Tomar.

Palavras-chave – Turismo, Estatísticas, Recolha de dados, React-Native, Ruby on Rails, Bluetooth Low Energy, Beacon

O código-fonte do projeto está disponível em:

https://github.com/NearDeath/Movtour (*backoffice*)

https://github.com/Pacn91/Movtour (aplicação móvel)

O *backoffice* está disponível em:

http://movtour.ipt.pt

Utilizador: teste@teste.pt Palavra-chave: 123456

A aplicação móvel está disponível na Google Play em:

https://play.google.com/store/apps/details?id=com.movtour&hl=pt_PT

Abstract

Mobile applications are nowadays one of the trends in the tourism promotion field. In addition to the information provided to the tourists, these applications can also be used to harvest information about them. However, most of the existing tourism-specific mobile applications are either restricted to a specific region or monument, or only provide a broad catalog of existing attractions or businesses ratings in several cities, but normally restricted to major cities. In order to enhance the tourist experience in a new city, some applications use GPS to help visitors locate attractions. However, this solution has some drawbacks, such as the excessive battery consumption, lack of accuracy when locating smaller points or even lack of coverage between and within buildings. Moreover, to the best of our knowledge, there are no applications able to automatically notify and provide information to the user as he or she approaches a point of interest. This project aims to address the abovementioned limitations, by proposing a system that, on one hand, automatically provides personalized information to the visitor passing by a specific point of interest, but that is also able to collect and provide insights about the visitors over time.

This project consists in the development of an information system to manage tourism information regarding monuments, points of interest and related visitors insights and statistics, the MovTour. To this end, a backoffice was developed and is used to manage all the touristic information in the system. To complement this, a mobile application to be used by the tourist was also developed, which interacts with Bluetooth beacons located at each point of interest. After locating a beacon, the application notifies the backoffice about the visit and shows relevant information to the visitor. The backoffice is in production, hosted by the Polytechnic Institute of Tomar. The mobile application was developed for Android and Apple iOS, with the first already available on Google Play. Finally, there are also ongoing beacon signal capture tests at Covent of Christ, in Tomar.

Keywords – Tourism, Statistics, Data collection, React-Native, Ruby on Rails, Bluetooth Low Energy, Beacon

The source code of the project is available at:

https://github.com/NearDeath/Movtour (backoffice)

https://github.com/Pacn91/Movtour (mobile application)

The backoffice is available at:

http://movtour.ipt.pt

User: teste@teste.pt Password: 123456

The mobile application is available on Google Play at:

https://play.google.com/store/apps/details?id=com.movtour&hl=pt_PT

Agradecimentos

Este trabalho está integrado e foi suportado financeiramente pelo projeto IC&DT Movtour nº SAICT-POL/24068/2016 através dos programas operacionais Centro2020 e FCT.

A sua realização só foi possível graças ao apoio e incentivo de diversas pessoas e entidades, para com os quais estarei eternamente grato.

Ao professor Renato Eduardo Silva Panda, pela sua orientação, suporte, disponibilidade total e verdade seja dita, engraçadas críticas construtivas que me permitiram não só solucionar os problemas que ocorreram durante o desenvolvimento, mas estimularam também a curiosidade por novas tecnologias, levando assim ao bom funcionamento do projeto e ao desenvolvimento de novas competências.

Ao professor Luís Miguel Lopes de Oliveira, pela sua orientação, pelo saber que me transmitiu, pelas opiniões e críticas, e apoio em decisões que envolviam o curso do projeto, optando sempre por um caminho mais realista e que permitiu que o projeto se desenvolvesse tão bem.

Ao João Frederico Pinto Coelho e professor Luís Manuel Mota dos Santos Figueira, do laboratório de Turismo do Instituto Politécnico de Tomar, pela disponibilização dos conteúdos presentes neste projeto, como também de todas condições proporcionadas para desenvolvimento deste trabalho.

Aos meus amigos e colegas, Miguel Coelho, Pedro Batista, Pedro Dias, Renato Antunes, David Carrilho entre outros que mesmo não sendo mencionados sabem quem são. Amigos que estiveram ao meu lado durante todos estes anos e que me ofereceram o seu companheirismo, força e apoio em todos os momentos.

Por último, às pessoas mais importantes da minha vida e sem as quais nada disto seria possível. Dirijo um agradecimento especial aos meus pais e irmãos que me apoiam incondicionalmente desde o primeiro dia, contando sempre com o seu incentivo, paciência e amizade constante. Um muito obrigado!

Índice

Índice de Figuras

Índice de Tabelas

Notação e Glossário

API	***A**pplication **P**rogramming **I**nterface (Interface de programação de aplicações)*
AR	***A**ugmented **R**eality (Realidade aumentada)*
BLE	***B**luetooth **L**ow **E**nergy (Bluetooth de baixo consumo energético)*
CDN	***C**ontent **D**elivery **N**etworks (Rede de fornecimento de conteúdo)*
DRY	***D**on't **R**epeat **Y**ourself (Não repita a si mesmo)*
GPS	***G**lobal **P**osition **S**ystem (Sistema de posicionamento global)*
JSON	***J**ava**S**cript **O**bject **N**otation (Notação de objeto Javascript)*
ORM	***O**bject-**R**elational **M**apping (Mapeamento objeto-relacional)*
POI	***P**oint **O**f **I**nterest (Ponto de Interesse)*
RoR	***R**uby **O**n **R**ails*
UX	***U**ser **E**xperience (Experiência de utilizador)*
VM	***V**irtual **M**achine (Máquina virtual)*
WYSIWYG	***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et (O que se vê é o que se obtém)*

1 Introdução

1.1 Enquadramento tecnológico

Hoje em dia vivemos na era da informação, onde a tecnologia se tornou ubíqua, fazendo parte do nosso quotidiano. A título de exemplo, em 2017 existiam cerca de 5 mil milhões de dispositivos, em que 3.5 mil milhões tinham acesso à internet [1]. Este número de dispositivos encontra-se na sua grande maioria nos países desenvolvidos, sendo difícil encontrar alguém num destes países que não possua pelo menos um dispositivo móvel. Além disso, esses dispositivos disponibilizam cada vez mais funcionalidades, tais como sistema de posicionamento por satélite (GPS), Bluetooth, internet móvel, vários tipos de sensores, entre outros.

Os dispositivos móveis oferecem hoje em dia um sem número de aplicações para os mais diversos fins, incluindo o turismo. A utilização de aplicações móveis para promoção do turismo é por isso uma das tendências atuais. Para além da informação útil disponibilizada ao visitante, estes tipos de aplicações podem também ser utilizadas para recolher informação relevante sobre os mesmos. Verificando-se em Portugal um crescimento acentuado do número de turistas nos últimos anos, tendo chegado aos 20,6 milhões de hóspedes em 2017 [2], seria de esperar que as cidades aproveitassem também as aplicações móveis para otimizar a sua oferta turística. No entanto constatamos que são poucas as que o fazem. Continua a existir falta de informação tanto para o turista que visita as cidades como para os responsáveis pelo turismo que pouco sabem daqueles que os visitam. Sem dados e indicadores é difícil otimizar corretamente a oferta turística de uma dada região.

1.2 Motivação e objetivos

Estando Portugal com um crescimento elevado de turistas, é importante começar a recolher e analisar dados sobre as pessoas que nos visitam. A análise desta informação permite, entre outras coisas, conhecer os hábitos dos visitantes e com isto implementar melhorias no serviço turístico, aumentando ainda mais o número de visitantes e o retorno obtido para Portugal. Com este trabalho é apresentado uma solução para este problema. Dotando a entidade responsável pelo turismo de um sistema que permite, por um lado apresentar informação aos turistas através destes novos

métodos, e por outro obter informação em tempo real sobre os visitantes e locais por onde se deslocam.

1.3 Descrição da solução proposta

A solução que aqui é apresentada consiste em duas partes distintas: uma aplicação *web* e uma aplicação móvel. A aplicação web (*backoffice*) foi desenvolvida em Ruby on Rails (RoR) [3], Bootstrap [4] e jQuery [5], onde os responsáveis pelo turismo podem gerir toda a informação sobre os locais de interesse e associar cada um destes pontos a dispositivos Bluetooth (*beacon*). Esta aplicação permite ainda analisar os dados recolhidos sobre os turistas e fornece também uma interface de programação de aplicações (API *REST)* que permite aceder a esta informação de forma programática.

Para ligar a esta, existe uma aplicação móvel desenvolvida em React Native e disponível para Android e iOS, sendo que a versão Android já se encontra na loja Google Play. Esta aplicação permite aos turistas obter informação sobre os locais a visitar, tendo a particularidade de interagir com o utilizador, avisando-o sempre que este se aproxima de um ponto de interesse (através dos *beacons*). Além disso a aplicação regista esses acessos, enviando-os para o servidor, o que permite ao gestor ver tendências tais como as horas de mais afluência de visitantes nos monumentos, os monumentos mais visitados, entre outros dados estatísticos que depois de analisados permitem implementar melhorias ao serviço turístico.

1.4 Organização do relatório

Este relatório está organizado da seguinte forma: no capítulo 2 é abordado o estado de arte na área, referindo diferentes soluções de turismo no mercado atual, mostrando depois aplicações móveis já existentes para o turismo. São também abordadas várias tecnologias existentes que poderiam ser utilizadas no desenvolvimento da solução que apresentamos, mostrando comparações entre estas que pesaram na escolha final. No capítulo 3 é abordada de forma resumida a arquitetura do sistema MovTour. O capítulo 4 descreve de forma pormenorizada os resultados obtidos neste trabalho, detalhando as características principais do *backoffice* e da aplicação móvel. Por fim, as conclusões e trabalho futuro são apresentadas no capítulo 5.

2 Estado de arte

Neste capítulo é apresentado o estado de arte relevante para o projeto. São analisadas as diferentes soluções e tendências para o turismo que se encontram atualmente no mercado e como se comparam.

2.1 Projetos em Turismo

Numa altura de elevado crescimento turístico e tecnológico, os países e municípios têm-se tentado manter atualizados, criando novos sistemas e aplicações para oferecerem um melhor serviço turístico.

De seguida são apresentados alguns projetos inerentes ao Turismo.

2.1.1 Exploresia

A Indonésia é um dos países onde o número de Turistas tem vindo a aumentar, no entanto, a falta de informação é bastante significativa. Um exemplo disso, é haver turistas que pensam que Bali é um país e não uma província da Indonésia. Para colmatar essa falta de conhecimento, começaram-se a utilizar aplicações móveis. Foi criada uma aplicação que faz uso de realidade aumentada. Uma aplicação que demonstra todos os pontos de interesse da Indonésia e que demonstra os locais com imagens 360º e manipulação de objetos 3D (ver Figura 1). Deste modo o turista pode pré-visualizar o ponto de interesse, e despertar interesse em visitar o local [6].

Figura 1 - Cenário de turismo virtual (retirado de [6])

2.1.2 e-Tourism

Este projeto dá ênfase a um fator muito importante, o *feedback* do próprio turista. A utilização de classificações e opiniões de outros turistas é uma mais-valia para a criação de um sistema mais robusto e que permite ajudar novos turistas que tenham interesses similares. Estas informações vão permitir que por um lado os promotores de turismo melhorem os serviços prestados e por outro, que novos turistas recebam recomendações turísticas seja com base na sua localização, por preferências definidas na aplicação e/ou por *feedback* de outros turistas [7].

2.1.3 Sistema de informação ao turismo baseado em *beacons* e realidade aumentada

Este projeto [8] pretende criar um novo sistema de informação de turismo no Japão, para abranger tantos os turistas que viajam em grupo (entenda-se grupo como pessoas que viajam com pacotes já pré-definidos), como também para turistas que viajam de forma independente (pessoas que gerem o seu próprio itinerário e tempo). Para esse efeito, é utilizada realidade aumentada que por sua vez utiliza informação proveniente de *beacons* presentes nos pontos de interesse. Sempre que um turista se aproximar de um ponto de interesse, imagens de realidade aumentada (AR) e vídeos são sobrepostos à imagem da câmara do dispositivo móvel, como se pode verificar na Figura 2.

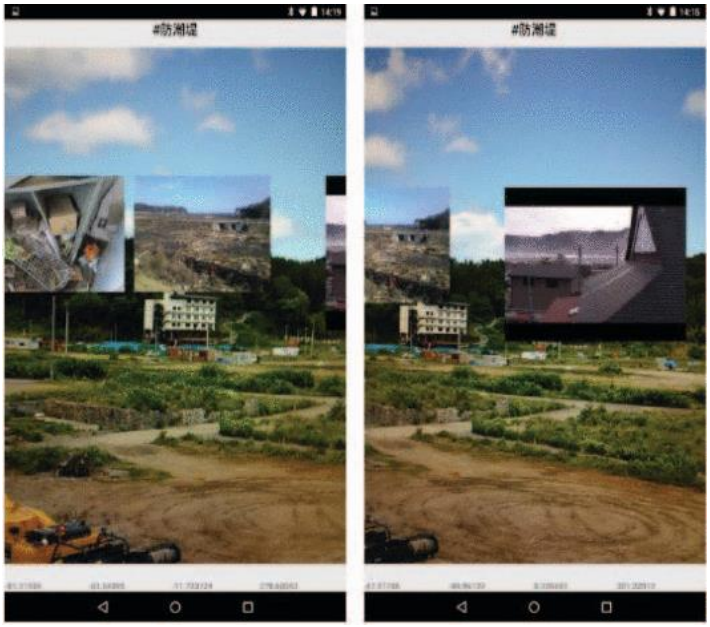

Figura 2 - Imagens AR e vídeos através da câmera (retirado de [8])

2.1.4 Sistema recomendado para turismo móvel

Este projeto recomenda um sistema que faz uso das informações provenientes dos turistas, como por exemplo pontuações, avaliações, ideias e comportamentos, de maneira a melhorar o serviço para turistas com interesses similares. Com esses dados os sistemas que fazem recomendações aos turistas, pode ser melhorado, indo mais ao encontro do que cada turista realmente procura.

O projeto também recomenda que se utilize redes de sensores nos pontos de interesse, para permitir aos turistas enviar informações de maneira conveniente e sem custos associados sobre os pontos de interesse. Desta maneira é possível dar pesos diferentes às avaliações, consoante se são feitas no local em questão ou se são feitas noutro local através da internet [9].

2.1.5 Conclusão

Com estes projetos e artigos, verifica-se que a utilização de realidade aumentada é a nova tendência nas aplicações móveis. Começa também a haver uma maior preocupação em recolher opinião do turista, pois isso oferece aos promotores de turismo informação que lhes permite melhorar o serviço turístico.

No entanto, do que se pôde verificar, não se vê projetos que englobem todas essas características. Além disso, as aplicações verificadas necessitam sempre que o utilizador interaja com ela sempre que chegam a um ponto de interesse, faltando-lhes ser mais automatizadas.

2.2 Aplicações móveis aplicadas ao turismo

A utilização de aplicações móveis para o turismo não é algo inexplorado, sendo que os *smartphones* são hoje em dia já bastante aproveitados nesta área. Dentro deste leque, existem um sem número de variantes disponíveis: desde aplicações sem custos para o utilizador, fornecidas pelas entidades públicas e circunscritas a locais específicos; aplicações pagas que fornecem informações mais detalhadas, servindo por exemplo de guia áudio para determinado monumento (*audio guides*) ou até agregadores de locais de interesse a nível global, como o TripAdvisor [10].

De seguida são apresentadas algumas destas aplicações.

2.2.1 oPORTO Insight

oPORTO Insight é uma aplicação que permite ajudar turistas e potenciais turistas a encontrar os melhores locais de eventos na cidade do Porto, além de descobrir as histórias da cidade, os segredos que ninguém conhece, os locais escondidos e os roteiros cuidadosamente escolhidos com revisão de locais para uma experiência personalizada. Tem a vantagem de funcionar sem conexão à internet. Na Figura 3 pode-se visualizar alguns *screenshots* da aplicação.

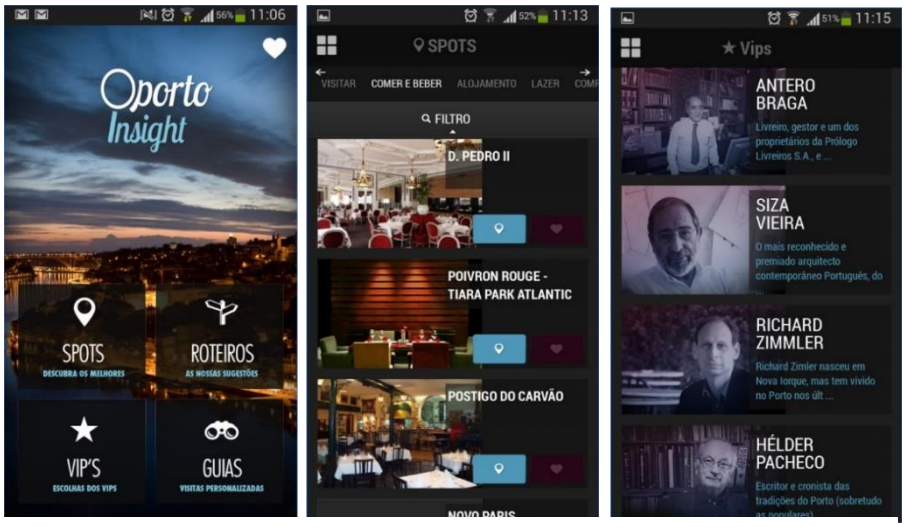

Figura 3 - Imagens da aplicação oPORTO Insight (retirado de [56])

2.2.2 Descubra Tomar

A aplicação Descubra Tomar [11], do grupo Descubra, permite aceder a uma grande panóplia de monumentos, restauração, alojamentos, entre outros locais da cidade de Tomar. Pode ainda ver a distribuição de locais, com a ajuda de um mapa interativo. Cada um desses locais contém uma descrição e a sua localização exata através do uso de GPS. Por fim, pode ainda visualizar os eventos que vão haver na cidade (ou arredores) através da Agenda incluída na aplicação.

No entanto, a aplicação peca por ter demasiada informação (tornando-se confuso procurar informação em específico), não ser intuitiva e por conter diversos *bugs* (por exemplo, na versão testada abrir o mapa interativo resultará no fecho da aplicação). Falta-lhe também ter suporte para versões mais atuais do Android. Na Figura 4 pode-se visualizar alguns *screenshots* da aplicação.

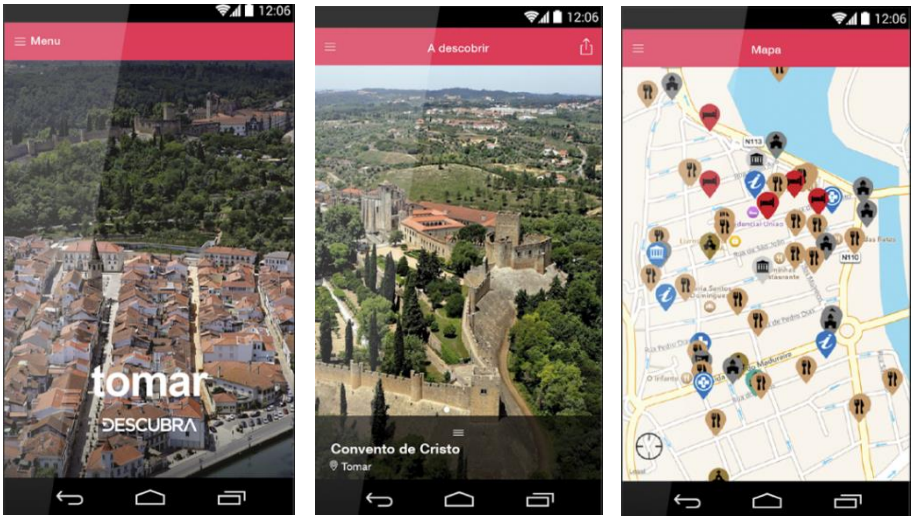

Figura 4 - Aplicação Descubra Tomar (retirado de [11])

2.2.3 TripAdvisor

O TripAdvisor é uma aplicação grátis e com mais de um milhão de transferências. Disponibiliza mais de 500 milhões de avaliações e opiniões dos viajantes, além de fotos e mapas, como se pode verificar na Figura 5. O TripAdvisor facilita a busca de tarifas aéreas mais baixas, os melhores hotéis, restaurantes e atrações. É ainda possível reservar hotéis, restaurantes e voos pela aplicação. Não necessita de dados móveis, permitindo descarregar mapas e avaliações antes de ir viajar.

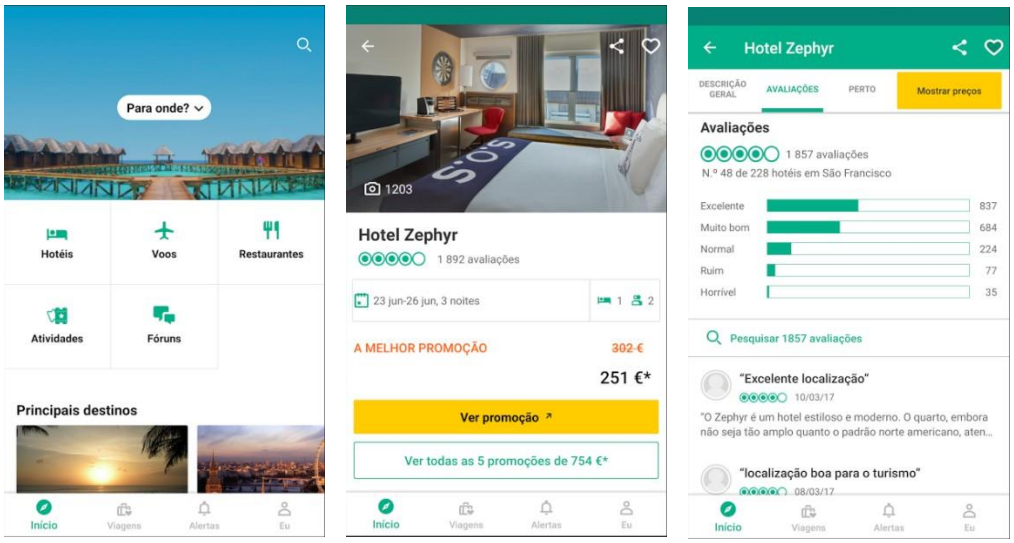

Figura 5 - Imagens da aplicação TripAdvisor (retirado de [10])

2.2.4 Guides by Lonely Planet

Segundo os criadores, Guides by Lonely Planet [12] é a aplicação certa para chegar ao coração de um destino. Esta aplicação, capaz de ser utilizada sem ter ligação à internet, permitindo descarregar mapas e áudio antes de ir em viagem, tem a capacidade de se tornar um conversor de moeda e até tem conselhos dos especialistas do destino que visita. A Figura 6 mostra o *layout* da aplicação.

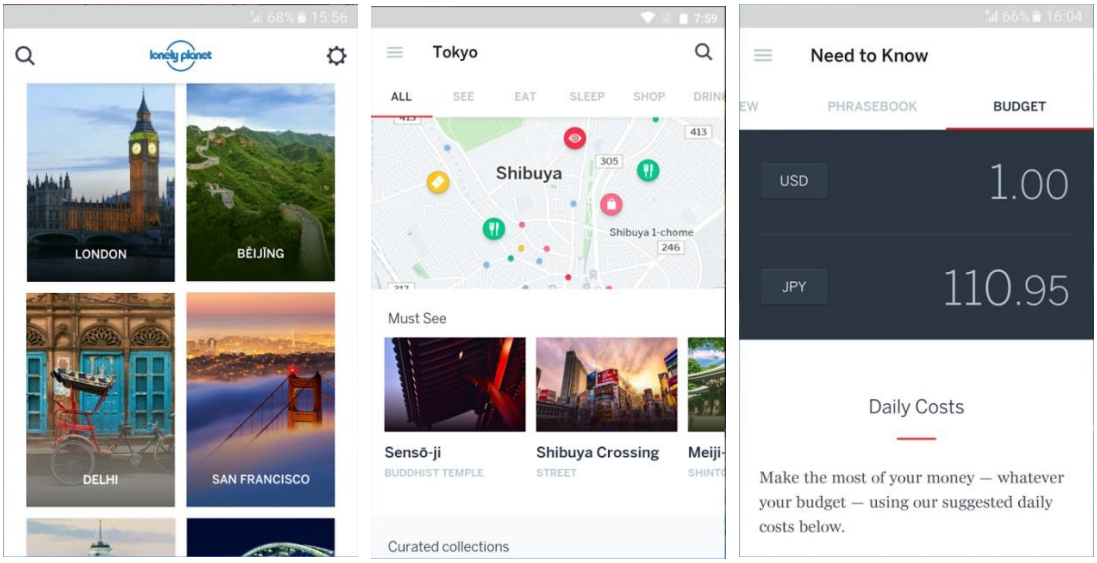

Figura 6 - Imagens da aplicação Guides by Lonely Planet (retirado de [53])

2.2.5 Conclusão

Das várias aplicações móveis para a vertente do turismo verificou-se que ou são restritas a uma região ou monumentos específicos, ou fornecem uma espécie de páginas amarelas para várias cidades embora esteja por norma disponível apenas para cidades grandes. Algumas delas funcionam com GPS, permitindo ao utilizador de certa forma saber onde está em relação a certo ponto de interesse, no entanto este pode ser impreciso ou simplesmente não funcionar dentro de edifícios, além de consumir bastante bateria. Por outro lado, tanto quanto foi possível verificar, não existe nenhuma solução que utilize *beacons* para que permita ao responsável pelo turismo (e não a uma entidade que pretende lucrar com publicidade) acesso a dados sobre visitantes. Assim esta ideia de aplicação destaca-se das outras por ter uma vertente autónoma, que interage com o utilizador sempre que esse passar perto de um Ponto de Interesse (PoI), por meio de notificações que ao serem pressionadas mostram informação sobre esse PoI. Além de mais, como a ligação entre o dispositivo e o

beacon é feita por Bluetooth Low Energy (BLE), funciona mesmo dentro de edifícios e reduz o consumo de bateria em comparação com o GPS.

3 *Background* em tecnologias de Turismo

Neste capítulo são apresentadas as tecnologias disponíveis para o desenvolvimento de aplicações *web* e móveis. É feita também uma breve análise das principais metodologias de desenvolvimento de *software* consideradas para a realização deste projeto.

3.1 Desenvolvimento de Aplicações Web

Existem cada vez mais opções possíveis para o desenvolvimento de aplicações *web*. Diferentes padrões de desenvolvimento, diferentes *frameworks* e tecnologias. De seguida são apresentadas algumas das tecnologias ponderadas para o desenvolvimento deste projeto.

De forma simplificada, uma aplicação *web* pode ser dividida em 3 partes distintas (ver Figura 7): um servidor de base de dados onde é feita a persistência de dados, um servidor *web* onde é executada a lógica da aplicação e uma camada de visualização apresentada ao utilizador. As duas primeiras estão normalmente presentes do lado do servidor (*server side*), sendo a última é enviada ao utilizador e executada do lado do cliente (*client side*), no seu navegador.

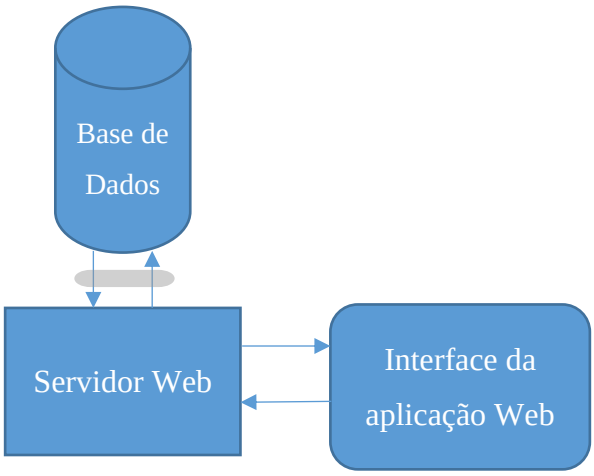

Figura 7 - Iteração típica de aplicações web

3.1.1 Padrão de desenvolvimento MVC

A aplicação a desenvolver segue o modelo MVC [13]. Este modelo é um padrão de arquitetura de software que separa o código desta em três blocos distintos (ver Figura 8). É uma forma de estruturar a aplicação de forma que a interface de visualização (*view*) esteja separada do controlo da informação (*models*). Esta separação é gerida por uma camada controladora, onde está a lógica da aplicação (*controllers*).

O *model* é a camada que representa os dados, fornecendo meios de acesso (leitura e escrita).

O *controller* contém métodos públicos/privados ou funções onde é implementada toda a lógica da aplicação, normalmente chamados de ações. Cada Ação é responsável por uma "página" do seu sistema. Este utiliza um dos modelos para aceder e guardar a informação, chamando/passada a informação depois à vista para que esta seja mostrada ao utilizador.

A *view* é responsável pela interação do sistema com o utilizador. Tudo o que o utilizador vê *(*seja por exemplo em HTML, XML, JSON ou até um ficheiro do tipo PDF) é gerado e exibido pela *view*. Esta tem como responsabilidade manipular os dados para a sua exibição.

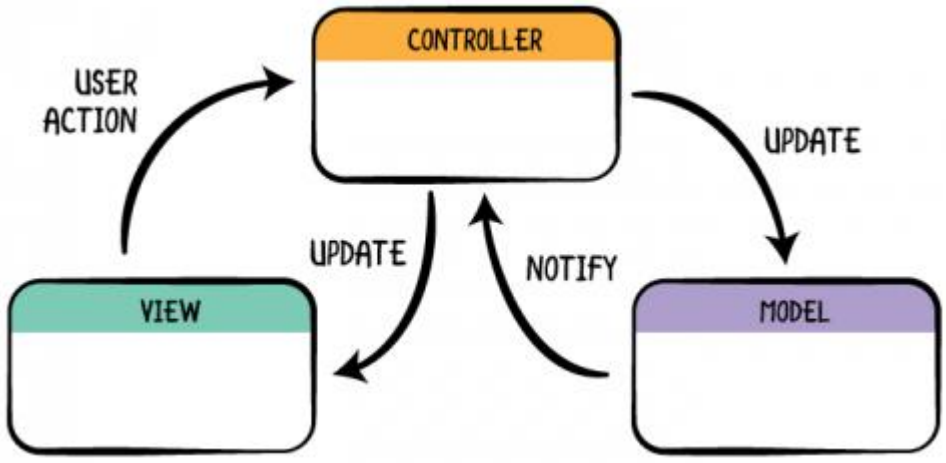

Figura 8 - Diagrama do modelo MVC (retirado de [54])

3.1.2 *Frameworks Web* (Servidor)

Hoje em dia existem imensas *frameworks* para o desenvolvimento de aplicações *web* para as mais diversas linguagens. Estas acabam por forçar o uso correto de certos padrões

de desenvolvimento tipo o MVC, o que resulta numa boa organização e reutilização de código, que sem uma *framework* dava muito mais trabalho. As principais vantagens de usar este tipo de *frameworks* são:

- Eficiência: Utilização de classes/funções presentes nos *frameworks* permitem poupar tempo de desenvolvimento;
- Reutilização: Utilização do mesmo código, apenas uma vez declarado, para várias funções;
- Segurança: Nível de segurança elevado, pois existem diversas pessoas a desenvolver e a reportar erros, de forma colaborativa, contribuindo para um aumento de segurança e de estabilidade;

Para a realização do projeto foram estudadas e comparadas algumas das *frameworks* mais conhecidas para a criação de aplicações *web*:

- Ruby on Rails (Ruby);
- Django (Python);
- Laravel (PHP);
- Asp.Net.

3.1.2.1 Ruby on Rails

Ruby on Rails (RoR) é uma *framework* livre que pretende aumentar a velocidade e a facilidade no desenvolvimento de *sites* orientados a base de dados. RoR é um projeto de código aberto escrito na linguagem de programação Ruby. As aplicações criadas, utilizando a *framework* RoR, são desenvolvidas com base no padrão de arquitetura MVC. Esta *framework* oferece sistema de migrações, ORM (Mapeamento objeto-relacional), uma comunidade forte e ativa, favorita das *Startups*, multiplataforma e segurança. Esta *framework* também incentiva ao uso do princípio DRY (*Don't repeat yourself*), fazendo com que o programador aproveite ao máximo o código já feito, evitando a repetição.

3.1.2.2 Django

Django [14] é uma *framework* de desenvolvimento rápido para aplicações web. Este é escrito na linguagem de programação Python. Django utiliza o modelo MTV (em comparação com o modelo MVC, a única diferença é a nomenclatura das camadas, visto que

o controller é renomeado para *view* e a *view* toma o nome de *template*). À semelhança do RoR, Django também utiliza o princípio DRY.

3.1.2.3 Laravel

Laravel [15] é uma *framework* para a linguagem de programação PHP. Esta *framework* é livre e *open-source* para o desenvolvimento de sistemas web que utilizem o padrão MVC. Laravel contém algumas características salientes como a sua sintaxe simples e concisa, um sistema modular com gerador de dependências dedicado e várias formas de acesso a base de dados relacionais.

3.1.2.4 Asp.net

Asp.net [16] é a plataforma da Microsoft para o desenvolvimento de aplicações web. O Asp.net é baseado na *framework* .Net herdando todas as suas características como qualquer aplicação .Net. As aplicações para esta *framework* podem ser escritas nas linguagens de programação C#, F# e Visual Basic. Esta *framework* segue o modelo MVC.

A Tabela 1 apresenta uma comparação das principais funcionalidades entre as várias *frameworks*.

Tabela 1 - Comparação de Frameworks em determinadas linguagens

	Linguagem	ORM	Frameworks				
			MVC	Testes	Migração de Base de Dados	Templates	Validação de Formulários
Ruby on Rails	Ruby	Sim (ActiveRecord)	Sim (ActiveRecord, Action Pack)	Sim (Testes unitários, testes funcionais e testes de integração)	Sim	Sim	Sim
Django	Python	Sim	Sim	Sim	Sim	Sim (built-in, Jinja2, Mako, Cheetah)	Sim
Laravel	PHP >= 5.5.9	Sim	Sim	PHPUnit	Sim	Sim	Sim
Asp.net	Muitas	Sim	Sim	Sim	Sim	Sim	Sim

Tal como a comparação demonstra, as principais *frameworks* disponibilizam as principais funcionalidades necessárias. A *framework* escolhida para o desenvolvimento da aplicação *web* foi a Ruby on Rails. Esta escolha deveu-se ao facto de ser uma *framework* madura, com uma comunidade forte, bastantes bibliotecas disponíveis e que é fortemente conhecida pela sua simplicidade, reutilização, capacidade de expansão, testabilidade, produtividade e manutenção [17]. Para além disso, é uma linguagem com a qual já se tinha algum contacto, ao contrário de Python e PHP. A *framework* Asp.NET foi excluída por, entre outras questões, obrigar ao uso de ambientes Windows, o que implica licenças comerciais, e ser por norma mais pesada.

3.1.3 Frameworks de Front-End

As *frameworks web* existentes permitem só por si criar uma aplicação funcional. No entanto, hoje em dia é necessário ir além disso e criar uma aplicação que proporcione uma boa experiência ao utilizador, facilitando-lhe o trabalho sempre que possível. Para facilitar isso, existem *frameworks* de *front-end*, que utilizam HTML+CSS+JS e podem ser facilmente usadas em conjunto com a maioria das *frameworks web*, ajudando no desenvolvimento da camada de visualização. As duas mais conhecidas são Bootstrap e Foundation.

3.1.3.1 Bootstrap

O Bootstrap é uma *framework*, *open-source*, bastante popular para desenvolvimento de *front-end*. Esta facilita o desenvolvimento de páginas responsivas – que se adaptem a diferentes tipos de equipamentos, desde a criação de protótipos simples para provas de conceito a plataformas completas. É baseada em *templates*, formulários, botões das linguagens HTML e CSS, podendo também conter extensões JavaScript.

3.1.3.2 Foundation

O Foundation [18] é uma *framework* para qualquer dispositivo, meio e acessibilidade. Esta é uma família de *frameworks* de *front-end* responsivos que facilitam o *design* de *sites* e aplicativos. O Foundation é semântico, legível, flexível e completamente personalizável. Esta *framework* utiliza também as linguagens HTML, CSS e JavaScript.

3.1.3.3 Conclusão

Em suma, ambas as *frameworks* são semelhantes. Foi escolhida a *framework* Bootstrap por ser um pouco mais conhecida e por já ter sido usada anteriormente.

3.1.4 Bibliotecas de JavaScript e aplicações de única página (SPA)

Na aplicação *web* é também usada uma biblioteca JavaScript. O jQuery é uma biblioteca (de código aberto) de funções JavaScript que interage com o HTML, desenvolvida para simplificar os scripts interpretados no navegador do cliente. A sintaxe foi desenvolvida para tornar mais simples a navegação no DOM (interface de programação que permite ver e tratar documentos HTML ou XML representando-os de forma hierárquica, numa árvore, onde cada nó é um objeto desse mesmo documento) do documento HTML. O seu objetivo é aumentar a facilidade de utilização do JavaScript. Esta biblioteca é já um dos requisitos para ter Bootstrap a 100% pelo que a sua utilização era óbvia.

O desenvolvimento *web* é algo que avança a elevada velocidade e atualmente a tendência é cada vez mais ter aplicações complexas a correr do lado do cliente, evitando muitas das viagens entre cliente e servidor. Esta solução é conhecida por Single Page Application (SPA) e existem já algumas *frameworks* para este efeito, nomeadamente React, Vue.js, Ember e Angular.Js. Seria interessante implementar uma solução destas, mas dada a natureza do projeto e a sua longevidade / carga de trabalho, foi impossível pensar neste tipo de solução.

3.2 Desenvolvimento de Aplicações Móveis

No desenvolvimento de aplicações móveis existem 3 grandes vertentes: aplicações nativas, aplicações híbridas e aplicações multiplataforma nativas.

As aplicações nativas, são desenvolvidas na linguagem específica do sistema operativo em que estão, ou seja, Objective-C ou Swift para iOS, Java para Android. Têm um desempenho muito elevado, são otimizadas, providenciam uma melhor experiência ao utilizador e traduzem assim uma confiabilidade superior. Acrescentando a isso ainda o acesso a uma panóplia de funcionalidades, tais como a câmara, o GPS, microfone, entre todos os outros. No entanto, isto implica uma aplicação e código diferentes para N plataformas, que por consequência aumenta em muito o custo e o tempo de desenvolvimento. A complexidade da linguagem também é muitas vezes superior em comparação com as aplicações híbridas.

As aplicações híbridas são aplicações em que o código é parcial ou totalmente partilhado entre diferentes sistemas. Para o efeito são utilizadas *frameworks* específicas que

reaproveitam o código. Uma das soluções mais comuns passava pela utilização de HTML, CSS, Javascript e *WebViews* (componente que trabalha como se fosse um mini *browser* que está dentro da aplicação). Desta forma, não era preciso desenvolver em linguagens mais complexas, além de não ser preciso desenvolver uma aplicação para cada plataforma, tornado este método mais barato e rápido. No entanto, o problema deste método é o desempenho da aplicação, que em comparação com as nativas era indiscutivelmente pior. Uma das razões era porque o Javascript, HTML e CSS trabalhavam todos na mesma *thread*, havendo depois transições lentas, *bugs*, e até paragens de aplicação.

Atualmente começam a aparecer *frameworks* que utilizam as APIs das componentes nativas, chamadas de aplicações multiplataforma nativas. Este novo método usa uma linguagem comum entre múltiplas plataformas, compilando depois grande parte do código da *framework* em código nativo do sistema. Isto torna o desempenho superior em comparação com as aplicações híbridas anteriores, ao mesmo tempo que o seu código é entre 80% a 90% reutilizável entre plataformas. Este novo método tenta ser o "melhor dos dois mundos" e tem vindo a ganhar uma grande quantidade de seguidores.

3.2.1 *Frameworks para aplicações móveis*

De seguida são apresentadas algumas das *frameworks* mais utilizadas atualmente.

3.2.1.1 Xamarin

O Xamarin [19] é uma *framework* da Microsoft para o desenvolvimento móvel. Pode criar aplicações nativas para Android, iOS e Windows Phone usando C# e suas funcionalidades. Além disso permite ainda usar funcionalidades específicas de .NET como por exemplo '*async/await*'.

Xamarin compila o código nos pacotes nativos de cada plataforma, em que no caso do iOS, o código é compilado para ARMS *assembly binary*, tornando-se puramente nativo.

Um desempenho perto das aplicações nativas, com *user interface* nativo e a habilidade de partilhar código entre plataformas, torna o Xamarin como uma das melhores *frameworks* no mercado. Além disso, conta com um suporte de uma grande comunidade. Segundo a empresa, em abril de 2017, acima de 1.4 milhões de desenvolvedores utilizavam produtos Xamarin.

No entanto, tem também alguns problemas, nomeadamente em projetos grandes, onde o desenvolvimento e a pré-visualização se torna bastante pesada durante, demorando muito a carregar, podendo chegar aos 5 minutos [20]. Há ainda um acesso limitado às bibliotecas *open-source*, além de erros de compatibilidade com algumas dessas bibliotecas. Outro ponto menos positivo, ainda que um pouco subjetivo, é o problema de a linguagem ser C#, não sendo muito apreciada entre a maioria dos programadores.

3.2.1.2 React Native

O React-Native [21] é uma *framework* para desenvolvimento móvel criada pelo Facebook em que se desenvolve maioritariamente em JavaScript, sendo o seu código partilhado entre plataformas. Ao contrário das *frameworks* híbridas que usam WebViews para renderizar HTML e CSS, React-Native usa componentes nativas, o que significa que o desempenho da aplicação fica mais próximo das aplicações nativas. No entanto apesar de ter acesso a variadas funcionalidades do dispositivo, que dão acesso à maioria dos componentes do dispositivo, precisa ainda assim de usar bibliotecas para ter acesso a funcionalidades específicas. Apesar de partilhar código entre plataformas, cabe ao programador definir certos componentes para cada plataforma. Em relação à sua linguagem, o React-Native utiliza JSX (JavaScript Expression - extensão da sintaxe do JavaScript) para editar as *views*, tendo por base a *framework* de desenvolvimento *web* SPA do Facebook, o React [22]. Uma das características mais fortes do React é a não-necessidade de compilar código, desde que não se adicione componentes. Isso permite a pré-visualização instantânea no dispositivo/emulador em qualquer alteração que se faça.

Por ser uma *framework* relativamente recente e em enorme crescimento, ainda não se encontra estabilizada e consequentemente sofre alterações a cada nova versão. Estas alterações ao longo de várias versões acabam por introduzir incompatibilidades e obrigar a trabalho extra do desenvolvedor para manter a aplicação atualizada e funcional. Apesar disso, React-Native é uma *framework* relativamente simples de usar, utiliza uma linguagem acessível, onde muitos programadores se sentem à vontade, tem um desempenho muito próximo das aplicações nativas, sendo por isso uma *framework* de grande potencial. Atualmente é dos projetos mais seguidos no GitHub [23].

3.2.1.3 Conclusão

Após estudar os diferentes tipos de *frameworks*, conclui-se que as aplicações nativas são claramente melhores em desempenho e em experiência de utilizador, mas por outro lado, caso se deseje ter a mesma aplicação em diferentes sistemas operativos, os custos e o tempo de desenvolvimento são muito mais elevados. No entanto se o objetivo for produzir para um único sistema operativo, a melhor solução são as aplicações puramente nativas.

Percebeu-se também que as aplicações híbridas, apesar de serem fáceis de desenvolver e permitirem ser multiplataforma, perdem por ter um desempenho relativamente fraco, o que pode vir a ser mitigado ao longo do tempo para aplicações mais básicas, uma vez que os telemóveis contêm cada vez melhores especificações.

Por fim, as novas *frameworks* em paralelo com o desempenho cada vez mais elevado dos smartphones de hoje em dia, trazem o melhor das aplicações nativas com o melhor das aplicações híbridas, ou seja, um desempenho elevado em multiplataformas.

Dentro dessa vertente, existem dois grandes candidatos: React-Native e Xamarin. Visto o React-Native trabalhar com Javascript e o Xamarin com C#, optou-se pelo React-Native, por achar que é uma linguagem mais simples de trabalhar e porque o React Native tem sido adotado para grandes aplicações, como é o caso do Instagram, Tesla, Airbnb, entre outras [24].

3.2.2 Bluetooth Low Energy (BLE)

Esta tecnologia foi selecionada para a aplicação desenvolvida em detrimento do GPS, permitindo detetar *beacons* colocados em POIs, uma vez que consome muito menos bateria e funciona por proximidade dentro e fora de edifícios.

Bluetooth Low Energy ou Bluetooth Smart como foi anunciado, é uma tecnologia de rede sem fios pessoal, desenhada para aplicações de saúde, *fitness*, *beacons*, segurança e indústrias de entretenimento em casa. Em comparação com o Bluetooth tradicional, o Bluetooth Smart consome muito menos recursos enquanto consegue manter uma distância relativamente semelhante de comunicação [25].

Em 2010 o Bluetooth Smart foi adicionado ao Bluetooth tradicional, com o nome Bluetooth Core Specification Version 4.0, e espera-se que até que até ao final deste ano,

mais de 90% dos dispositivos o suportem. Entre os sistemas móveis suportados estão o iOS, Android, Windows Phone, BlackBerry, como também macOS, Linux, Windows 8 e 10.

A tecnologia Bluetooth continua a ser melhorada, sendo que na versão atual (5) a distância de comunicação quadruplicou, a velocidade duplicou e a capacidade de transmissão aumentou oito vezes [26], sendo isso importante para aplicações IoT (Internet of Things).

3.2.3 *Beacons* BLE

Um *beacon* é um pequeno dispositivo, com ID único, que emite sinais por meio da tecnologia Bluetooth Low Energy. São dispositivos muito simples, apenas contêm uma CPU, rádio e baterias. Podem, no entanto, conter acelerômetros, sensores de temperatura ou outros sensores únicos. Tudo depende para qual é o efeito procurado, mas todos têm uma particularidade – transmitem um sinal.

Esses sinais podem ser capturados por dispositivos móveis, desde que possuam essa tecnologia (Bluetooth Low Energy). Desta forma, basta ter uma aplicação instalada no dispositivo móvel, que esteja conectada aos *beacons*, para se começar a fazer uma série de ações. Colocando-o num determinado local abre um conjunto de possibilidades, entre elas ler o identificador único universal (do inglês *universal unique identifier* ou UUID), contar as vezes que passamos por ele, ver a que distância está, entre outros. Esta tecnologia na aplicação serve precisamente para isso, ter um *beacon*, com um UUID único, associado a um determinado ponto de interesse e através desse UUID é possível saber onde está o utilizador.

No entanto, a tecnologia inerente aos beacons pode variar ligeiramente consoante o standard que se utiliza, em que os três standards mais reconhecidos são explicados de seguida.

3.2.3.1 iBeacon

Tecnologia *Beacon* introduzida pela Apple Inc. para oferecer um modo diferente de fornecer informações e serviços baseados em localização para iPhones e outros dispositivos iOS. Apesar do seu desenvolvimento ter sido focado nos dispositivos iOS, a tecnologia iBeacon funciona também nos dispositivos Androids. Quando está a enviar anúncios (em *broadcast*), os seus pacotes correspondem a:

- UUID: número identificador do *beacon*;
- Major: número de 2 byte para identificar um grupo de *beacons*;
- Minor: número de 2 byte que identifica um *beacon* dentro de um grupo (major);
- TX Power: Potência do sinal em dBm recebida quando o recetor se encontra a 1 metro de distância.

3.2.3.2 Eddystone

Tecnologia *open-source* da Google que vem competir com o iBeacon da Apple. Ao contrário do iBeacon, o Eddystone foi desenvolvido de raiz para iOS e Android, originando menos erros/falhas. As maiores diferenças, no entanto, são nos pacotes que o *beacon* envia no *broadcast*, que são:

- URL: Envia um url para o dispositivo, não necessitando de uma app;
- UID: fundamentalmente igual aos iBeacons;
- TLM: transmite a telemetria do beacon. Atualmente esse pacote inclui a vida da bateria, a temperatura, a data desde que foi ligado e o número de anúncios que enviou desde que foi ligado;
- EID: identificador encriptado que varia periodicamente para prevenir que utilizem o *beacon*. As aplicações autorizadas podem utilizar um '*trusted resolver*' para obter um identificador estável. É necessário haver uma ligação com o servidor para converter o identificador encriptado num identificador estável.

3.2.3.3 Altbeacon

Proposta da Radius Network para um mercado aberto para *beacons*. A ideia passa por ultrapassar os problemas dos protocolos favorecerem um determinado fabricante. Com os Altbeacons, o código é aberto, compatível com outras plataformas móveis, e permite uma maior flexibilidade a personalizar o código fonte.

3.3 Outras tecnologias

Para além das tecnologias referidas nas secções anteriores, foram ainda utilizadas certas ferramentas que são muito úteis e que vale a pena referir.

3.3.1 Ambiente de desenvolvimento virtual

O desenvolvimento da aplicação web foi feito utilizando um ambiente de desenvolvimento virtual. Para isso foi utilizado o Vagrant [27] e o Virtualbox [28]. Vagrant é um *software open-source* que tem o propósito de construir e manter ambientes portáteis de desenvolvimento de *software*. Este *software* pode ser instalado, por exemplo no VirtualBox, Hyper-V, Docker, VMware e também AWS. A ideia principal por detrás do *software* está no facto de que a instalação e manutenção de um ambiente virtualizado torna-se cada vez mais fácil num grande projeto de desenvolvimento de *software*, em comparação com um ambiente real entre equipas. Este simplifica a gestão das configurações do *software* necessário para aumentar a produtividade do desenvolvimento. Esta solução permite ter um ficheiro de instruções (Vagrantfile) onde está descrita a configuração do ambiente que é utilizado. Depois de criado o ficheiro e realizada a configuração da máquina virtual a utilizar, qualquer pessoa pode entrar rapidamente no projeto fazendo o comando "vagrant up", o que resultará num ambiente completamente configurado para o projeto em causa. Outra grande vantagem é que, independentemente do sistema operativo de qualquer programador, garantimos que o ambiente de desenvolvimento é o mesmo para todos.

3.3.2 Controlo / Revisão de código

Para a gestão e controlo de versão de código foram utilizados o Git [29] e o Github. Git é um sistema de controlo de versões distribuído. O objetivo deste sistema é a possibilidade de desenvolvimento de projetos nos quais diversas pessoas podem colaborar simultaneamente no mesmo. Ao mesmo tempo podemos ter uma equipa a trabalhar no mesmo projeto, a fazer o seu desenvolvimento em qualquer parte deste. A qualquer momento o programador pode submeter as suas alterações para o seu repositório local e quando desejar enviar para o repositório partilhado, ou puxar as de outros membros da equipa para o seu trabalho, neste caso fornecido pelo Github. Outra grande vantagem deste serviço é permitir a qualquer momento reverter o código para uma revisão anterior.

Github é um serviço *web* que oferece ainda inúmeras funcionalidades extras para o Git. É no Github que é possível alojar os projetos. Este serviço tem inúmeras funcionalidades:

- Existência de repositórios públicos e privados;

- Permite *pull requests*[1];
- Resolução de problemas (*issues*);
- Execução de *rollbacks*[2] e *commits*[3];
- Permite ter código atualizado e seguro;
- Permite a existência de ramos (*branches*[4]) em paralelo para que o desenvolvimento seja efetuado sem que o código atual seja afetado;
- Revisões de código;
- Possibilidade de fazer clone ou descarregar repositórios públicos.

[1] Notificar equipa que as alterações estão prontas para serem revistas e adicionadas a um ramo específico.
[2] Voltar atrás, desfazer alterações realizadas por submissões anteriores.
[3] Gravar alterações realizadas no repositório.
[4] Diferentes ramos de desenvolvimento.

4 Arquitetura do Sistema Movtour

A arquitetura do sistema Movtour é composta por três componentes, tal como ilustrado na Figura 9:

- *Beacons* – que estão presentes em cada ponto de interesse;
- *Backoffice* – responsável por toda a gestão, monitorização e estatística de dados;
- Aplicação móvel – elo de ligação entre os *beacons* e o *backoffice*, responsável por transmitir a informação ao utilizador e enviar meta-dados ao *backoffice*.

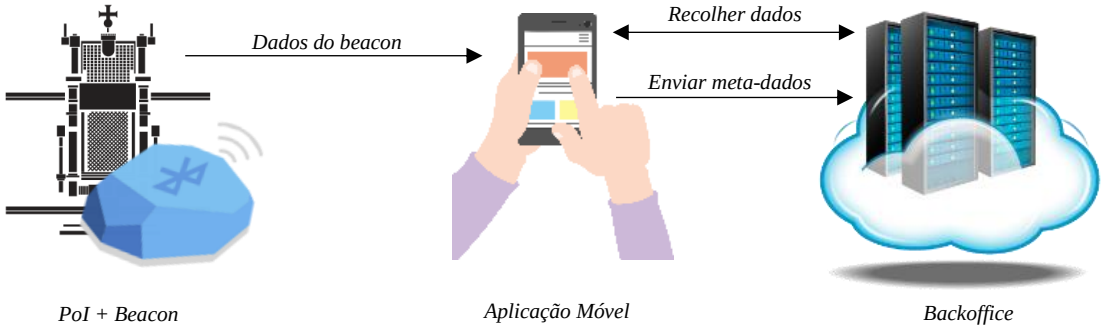

Figura 9 - Arquitetura geral do sistema Movtour

4.1 PoI e *Beacon*

Em cada monumento existem N pontos de interesse em que cada um desses pontos contém pelo menos um pequeno dispositivo que transmite sinais através de Bluetooth, chamado de *beacon*. Cada *beacon* contém um identificador único (entre outros dados, tais como a distância, major e minor) que permite ao ser detetado, identificar qual o ponto de interesse associado. A associação entre ponto de interesse e *beacon*(*s*) é feita através do *backoffice*, no momento de criação ou edição de um ponto de interesse.

4.2 Aplicação Móvel

A aplicação móvel (desenvolvida para Android e iOS), de uma maneira resumida, comunica com o utilizador sempre que este passa perto de um ponto de interesse, apresentando o ecrã que contém a informação do ponto de interesse em questão, ou caso o telemóvel esteja bloqueado, alertando para a presença do PoI através de uma notificação. Deste modo permite que mesmo que este não possua um guia turístico, possa conhecer a

história do monumento/ponto de interesse. A comunicação resulta da interação do dispositivo com o *beacon* que está presente no ponto de interesse, via Bluetooth. Adicionalmente, a aplicação contacta também o *backoffice* sempre que existe uma comunicação entre dispositivo móvel e *beacon* para o registo de informação sobre a visita (registando por exemplo a hora e o ponto de interesse visitado).

A aplicação tem suporte para diversos idiomas e diferentes níveis de conteúdo (ex.: Iniciação, Divulgação, etc). Permite ainda consultar a distribuição dos vários pontos de interesse existentes no mapa.

Por fim, esta não necessita de internet para o seu funcionamento, exceto na primeira utilização de modo a descarregar a informação no dispositivo ou então quando se quer atualizar a informação da aplicação. No entanto, sem internet os dados não são comunicados para o *backoffice*.

4.3 Backoffice

O *backoffice* consiste num sistema de informação sobre património natural e cultural, que permite gerir toda a informação utilizada pelo projeto. Entre outros, é possível gerir (criar, apagar ou editar) a informação de cada monumento e respetivos pontos de interesse, gerir informação sobre *beacons*, definir que monumentos estão visíveis na aplicação, organizar os mesmos por categorias ou escolher os níveis de descrição textual disponível de forma dinâmica. É ainda possível consultar informações sobre as visitas registadas permitindo extrair conhecimento sobre estas através de estatísticas, gráficos, entre outros. Tem suporte para várias línguas quer no *backoffice*, quer de forma dinâmica no conteúdo informativo que é disponibilizado à aplicação móvel (ver Figura 10).

De forma geral, em ambiente de produção, o serviço utiliza o nginx - servidor HTTP para responder aos pedidos para o *backoffice*, que podem vir através do navegador (por um administrador) ou através da aplicação móvel (através da API). Estes pedidos, quando para recursos estáticos como imagens ou ficheiros de estilo (CSS) são servidos diretamente pelo nginx, melhorando o desempenho da plataforma. Atualmente os recursos estáticos consistem em ficheiros locais (alojados no próprio servidor). No entanto, o sistema está preparado para usar serviços de alojamento em nuvem (*cloud storage)* tais como a Amazon S3 ou até redes de distribuição de conteúdo (CDNs – *content delivery networks*). Os pedidos para recursos

dinâmicos são reencaminhados para o Passenger – o servidor de aplicações web em Ruby. Neste é executado o código do projeto (que utiliza a *framework* Rails). A informação atualmente é guardada numa base de dados relacional PostgreSQL, embora o sistema esteja estruturado de forma a minimizar o trabalho necessário para migrar para outro serviço de bases de dados ou até usar uma base de dados não relacional. A aplicação utiliza ainda um servidor Redis, um serviço de armazenamento de estruturas de dados em memória, tipicamente utilizado como servidor de bases de dados não relacionais (NoSQL), para *caching* ou para troca de mensagens (*broker*). Neste caso, é utilizado para implementar filas de mensagem (padrão *publisher/subscriber*) e permitir a comunicação entre a aplicação de *backoffice* e o servidor de *websockets* (*actioncable*), onde o navegador do gestor se liga para receber informação de visitantes em tempo real.

De seguida, são abordadas com maior profundidade as características e funcionamento de todo o sistema.

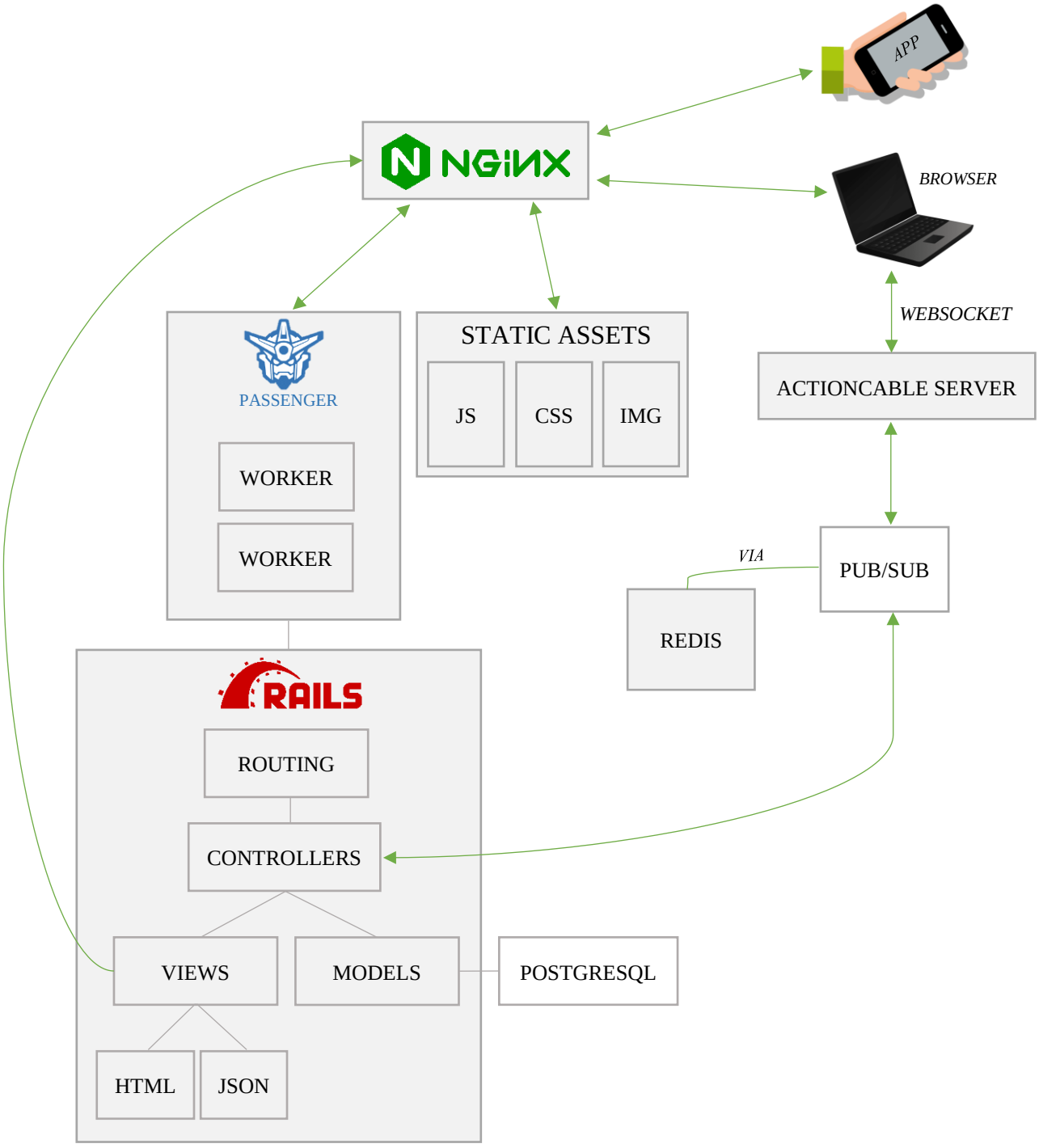

Figura 10 - Arquitetura do backoffice

5 Resultados

5.1 Backoffice

Como ferramenta de *backoffice*, foi desenvolvida uma aplicação *web* em Ruby on Rails que segue o padrão MVC (descrito na secção 3.1.1). A aplicação está apenas acessível aos administradores dos monumentos, pois é uma ferramenta de gestão de pontos de interesse. Esta permite além das funcionalidades base (adicionar, editar e remover pontos de interesse), adicionar *beacons* e associa-los a um ponto de interesse, registar os acessos dos turistas aos pontos de interesse e consultar estatísticas geradas automaticamente com base nos acessos registados.

5.1.1 Modelo de Dados

Os principais modelos que constituem a aplicação são:

- Category – representa a categoria a que um monumento pertence;
- Monument – objecto com os dados de um monumento;
- Beacon – representa o dispositivo Bluetooth colocado nos pontos de interesse;
- Poi (Point of Interest) – ponto de interesse de um determinado monumento;
- DescriptionType – representa os tipos de descrição existentes num ponto de interesse;
- PoiDescription – representa a descrição de um determinado ponto de interesse;
- Access – representa um acesso de um visitante a determinado ponto de interesse
- User – representa o utilizador (neste caso um dos administradores);

O diagrama apresentado na Figura 11 apresenta estes mesmos modelos e a relação entre eles. Estes são implementados usando classes em Ruby e os seus dados são mapeados diretamente numa base de dados relacional. Tal é feito através de um sistema de mapeamento de objetos-relações (ORM) fornecido pela biblioteca *ActiveRecord*. Através desta é possível manipular toda a informação sem o uso direto da linguagem SQL (a biblioteca cria uma camada de abstração entre a base de dados e o programador), evitando assim eventuais falhas de segurança introduzidas pelo programador.

Existem ainda modelos adicionais que servem para representar diversas traduções de alguns modelos (ex.: Category:Translation). Esses modelos contêm toda a informação traduzida.

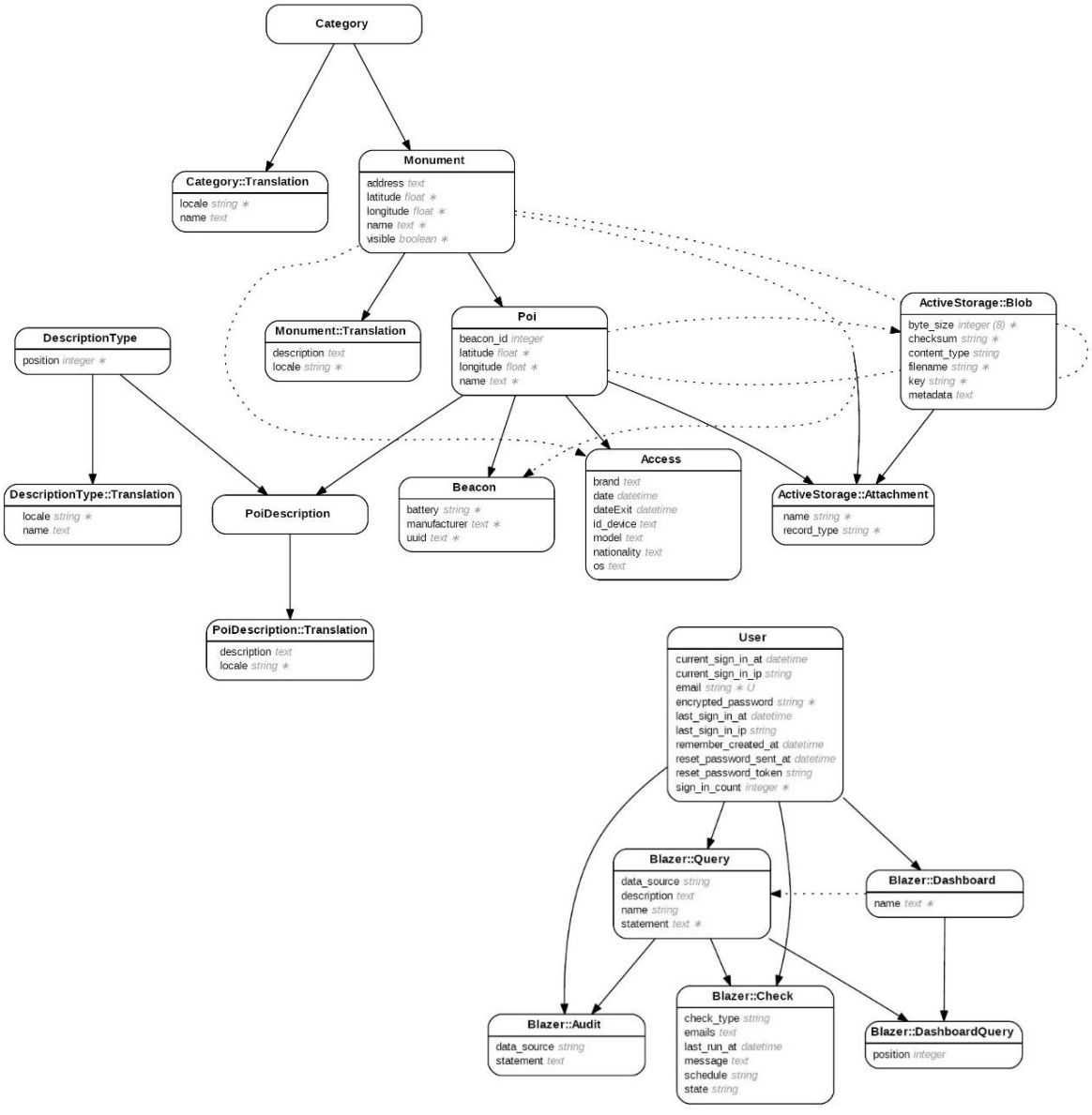

Figura 11 - Diagrama Entidade Relacional

5.1.2 Componentes do backoffice

O *backoffice* é destinado aos gestores dos monumentos, sendo por isso necessário efetuar login para se poder aceder às suas funcionalidades. A *interface* foi pensada e concebida com o objetivo principal de ser simples e intuitiva, permitindo que qualquer utilizador com conhecimentos básicos sobre computadores possa utilizar o sistema. Os formulários foram normalizados e, portanto, o método de criação de monumentos ou pontos de interesse são bastante similares, tal como acontece com a introdução de *beacons*, categorias dos monumentos ou tipos de descrição.

Nos pontos seguintes descreve-se cada uma das características quem compõem o *backoffice*.

5.1.2.1 Categorias dos monumentos

O *backoffice* permite que haja uma categorização de monumentos (ver Figura 12). O objetivo desta funcionalidade é permitir aos utilizadores da aplicação móvel e do *backoffice* ver e filtrar apenas por monumentos de uma determina categoria, oferecendo-lhe assim uma melhor experiência.

Figura 12 - Lista de Categorias

Para a sua implementação, existe a tabela 'Category' que contém todas as categorias dos monumentos, permitindo que o gestor possa adicionar, editar ou remover categorias de forma dinâmica. Essas categorias são disponibilizadas, através da API, para que a aplicação móvel as possa utilizar.

5.1.2.2 Categorias de descrições

Um dos problemas comuns na maioria das aplicações e de informações turísticas existentes é o facto de oferecerem um texto único para determinado ponto de interesse. No entanto, os visitantes são por norma bastante distintos, tendo diferentes graus de escolaridade, idade ou profissão. Para colmatar este problema, foi implementado um sistema de categorias de descrição que permitem para um mesmo ponto de interesse definir vários graus de informação, desde a mais básica até à mais aprofundada e erudita. Atualmente existem quatro níveis de conteúdo do património natural e cultural. Esses níveis são os seguintes:

- Iniciação: destinado a públicos pouco relacionados com o objeto de visita e capazes de lidar com um nível de complexidade básica (correspondente, comparativamente até ao 9º ano de escolaridade ou equivalente);
- Divulgação: correspondendo a uma apresentação e interpretação para públicos com capacidade para acolherem a complexidade média na abordagem ao objeto de visita (que corresponde até ao 12º ano ou equivalente);
- Aprofundamento: destinado a públicos com formação superior ou equivalente, procurando responder a necessidades de uma interpretação mais profunda e detalhada (que corresponde a licenciados, pós-graduados, e autodidatas com aptidões equivalentes a estes níveis de formação académica);
- Investigação: que se focará nos públicos científicos e técnicos que operam nos domínios onde os objetos de visitação se poderão contextualizar, destinados a ilustrar certos aspetos mais específicos e profundos (que toma corpo no que podemos designar por visitas técnico-científicas em sede de Investigação e Desenvolvimento).

Esta funcionalidade é implementada no *backoffice* através da tabela 'DescriptionType' e associação desta aos pontos de interesse. Tal como acontece com as categorias dos monumentos, o sistema foi desenhado para permitir ao gestor adicionar, editar e remover novos tipos de descrição (Figura 13). Estas alterações são disponibilizadas automaticamente através da API para a utilização na aplicação móvel.

As descrições de cada ponto de interesse ficam presentes na tabela ‘PoiDescription’ e contém dois atributos, a descrição e a categoria correspondente. Cada ponto de interesse tem um número de descrições correspondente ao número de tipos de descrição.

Figura 13 - Lista dos tipos de descrição

5.1.2.3 Visibilidade dos monumentos

O gestor dos monumentos pode definir a visibilidade (ou invisibilidade) de determinados monumentos aos utilizadores da aplicação móvel. Para isso, o gestor apenas tem que pressionar o ícone que está presente em cada monumento, como demonstra a Figura 14.

Figura 14 - Visibilidade dos monumentos

Essa funcionalidade está implementada através de um atributo presente no modelo ‘Monument’ e respectiva tabela, que define a visibilidade do monumento. Este atributo é depois usado para filtragem na geração da lista de monumentos a disponibilizar à aplicação móvel através da API.

5.1.2.4 Estatísticas de Acessos

Um dos objetivos principais do projeto é o registo de informação e extração de conhecimento sobre os turistas. Isso possibilita retirar dados estatísticos e otimizar a oferta turística. Para tal, quando o utilizador da aplicação móvel se encontra perto de um *beacon,* para além de receber informação desse mesmo ponto de interesse é, simultaneamente, enviado para o *backoffice* um registo de acesso ao mesmo.

Esse registo é guardado pelo *backoffice* na tabela ‘Access’ para efeitos de estatística, e é composto por dados que não identificam o turista, tais como: a data do acesso, a nacionalidade (baseada na linguagem do sistema operativo do dispositivo móvel), o sistema operativo do dispositivo, a marca do dispositivo, o modelo do dispositivo, o identificador do dispositivo (guardado de forma indireta, através de uma *hash* anonimizada, para salvaguardar a privacidade do turista) e o ponto de interesse visitado. A Figura 15 apresenta a lista dos últimos acessos.

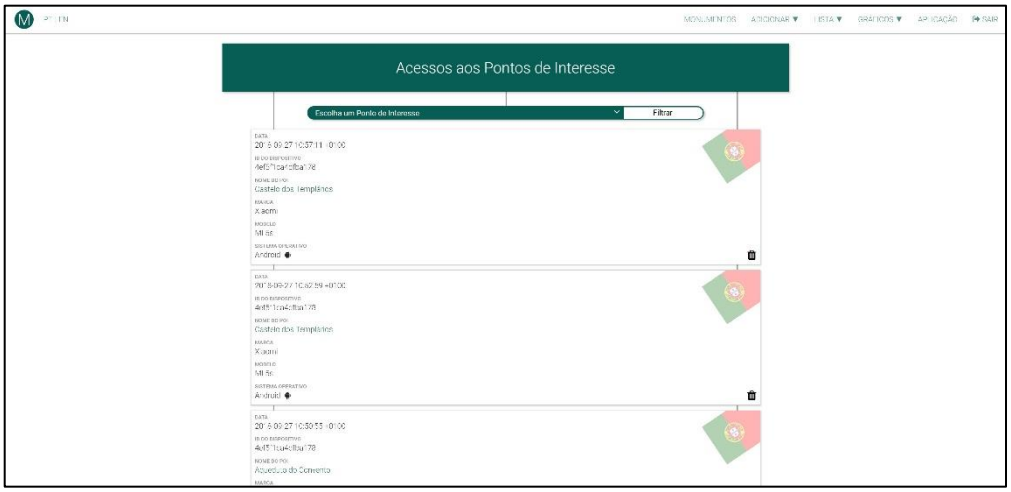

Figura 15 - Página de acessos por pontos de interesse

Com base nesta informação, são calculadas estatísticas e construídos vários gráficos e tabelas disponíveis no *backoffice* através das bibliotecas 'highcharts' e 'chartkick'. Estes gráficos permitem por exemplo, verificar quais os pontos de interesse com maior afluência (como demonstrado na Figura 16), ou que nacionalidades são mais frequentes (Figura 17), podendo-se assim ajustar conteúdos para ir de encontro às necessidades. É possível também retirar informação valiosa, tal como tendências. Sabendo os meses de maior afluência como demonstrado no gráfico da Figura 18, é possível por exemplo ajustar meios às necessidades do momento.

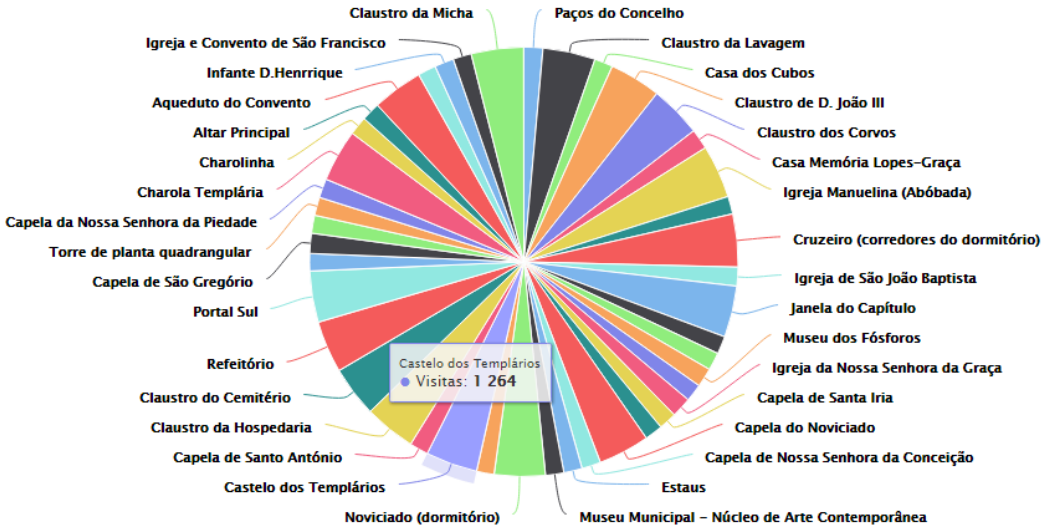

Figura 16 - Gráfico circular de visitas por ponto de interesse

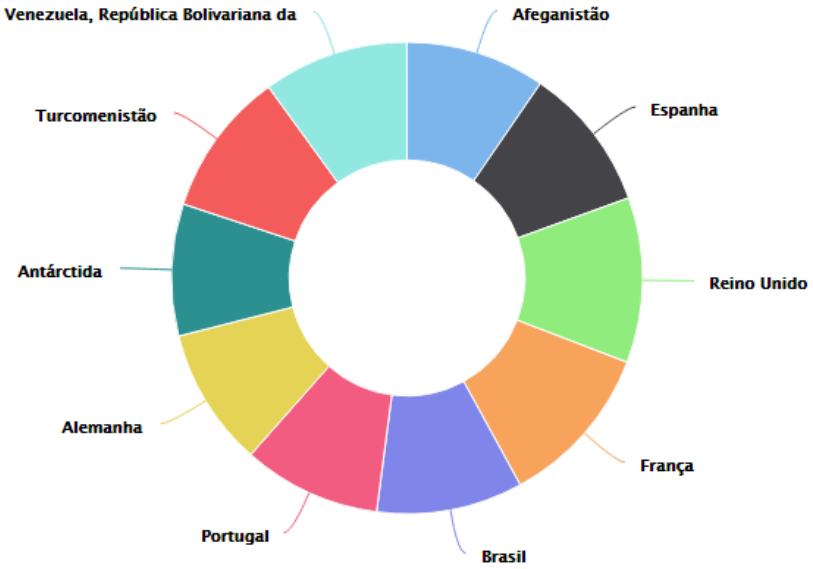

Figura 17 - Dashboard de visitas baseadas na nacionalidade

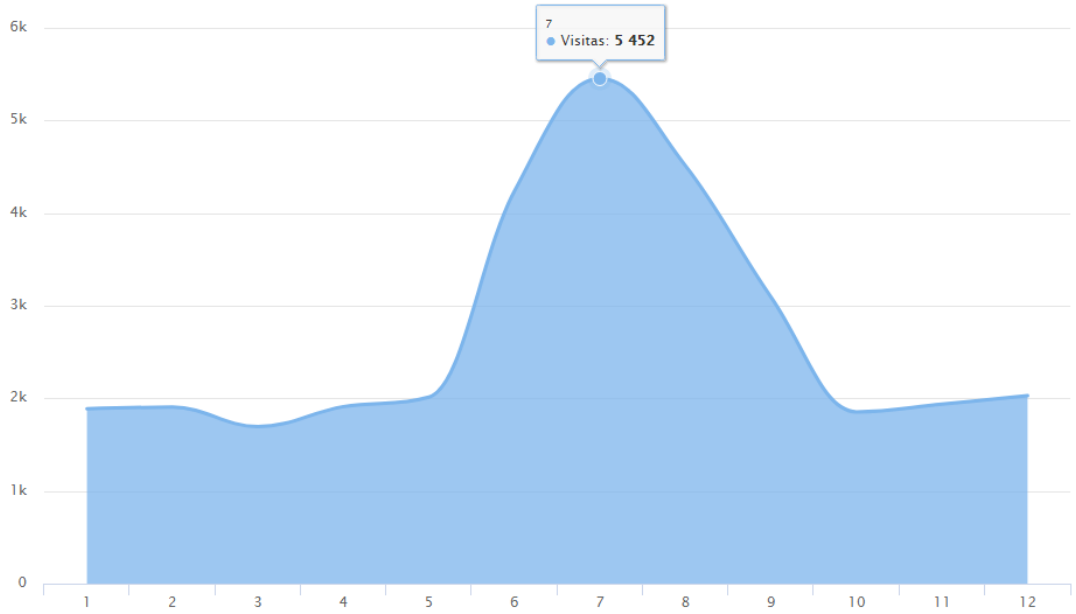

Figura 18 - Número de visitantes por mês

Além destes gráficos, também existe um *heatmap* que consiste num mapa (implementado sobre o Google Maps) com representações visuais de cores que mostram a concentração de utilizadores da aplicação que estão a interagir com os *beacons* (Figura 19).

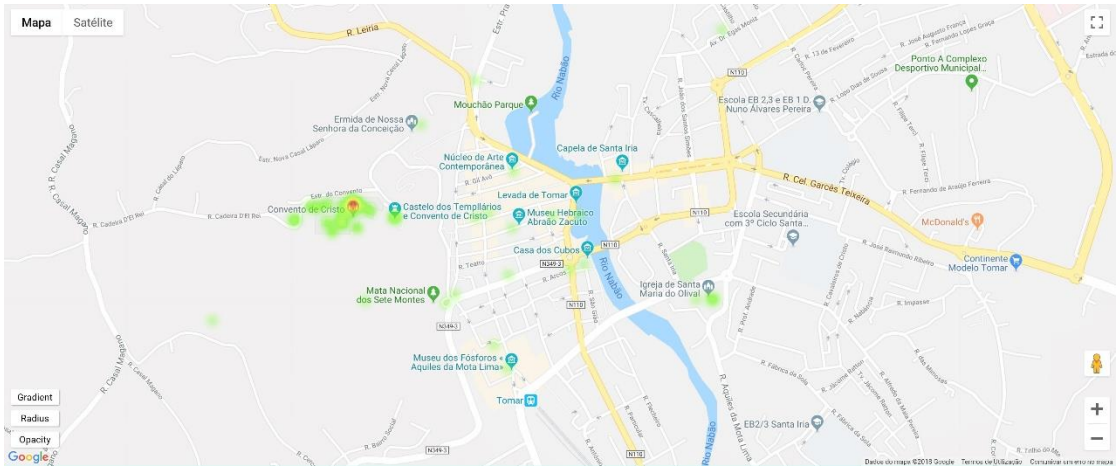

Figura 19 - Heatmap de acessos nos pontos de interesse

As estatísticas apresentadas são apenas um pequeno exemplo das possibilidades que o conjunto de dados registado nos proporciona, tendo sido geradas com dados fictícios por ainda não terem sido recolhidas visitas reais.

5.1.2.5 API – *Interface* de programação de aplicações

O *backoffice* tem como finalidade a gestão da informação turística e visualização de dados estatísticos por parte do gestor. No entanto é também necessário que informação sobre monumentos, pontos de interesse, *beacons* e categorias esteja acessível aos utilizadores da aplicação móvel ou qualquer outra aplicação que possa surgir no futuro. Mais ainda, é necessário que estas consigam também fornecer informação, de forma a registar os acessos de visitantes.

Para este efeito o *backoffice* expõe também um conjunto de pontos de acesso através de uma API *REST*, que permitem a qualquer aplicação comunicar com esta, enviando e recebendo informação no formato JSON (*JavaScript Object Notation*). Embora existam vários, os dois principais pontos de acesso da API usados atualmente pela aplicação móvel permitem:

1) Obter uma lista extensiva sobre monumentos, seus pontos de interesse e *beacons*, categorias do tipo de conteúdo, entre outros dados, tal como exemplificado na Figura 20.
2) Enviar para o backoffice, via HTTP *POST*, informação de acessos vindos da aplicação móvel, como é descrito na secção 5.2.2.7, Figura 36.

Adicionalmente, é fornecida a documentação da API de modo a permitir que novos desenvolvedores percebam como essa pode ser acedida e utilizada.

A API está disponível em: **http://movtour.ipt.pt/apipie/**

```
{
  "categories": [
    {
      "id": 3,
      "position": 1,
      "name_pt": "Iniciação",
      "name_en": "Initiation",
      "name_fr": "Initiation",
      "name_de": "Einleitung"
    },
    { … }, // 6 items
    { … }, // 6 items
    { … } // 6 items
  ],
  "monuments": [
    {
      "id": 1,
      "name": "Convento de Cristo",
      "category": "Castelos",
      "cover_image": "http://movtour.ipt.pt/rails/active_storage/blobs/eyJfcmFpbHMiOnsibWVzc2FnZSI6IkJBaHBWQT09IiwiZXhwIjpudWxsLCJwdXIiOiJibG9iX2lkIn19--f2d84f43ac88fdae2fd9c1750b4079eba390e4e9/ConventoCristo.jpg?locale=pt",
      "cover_image_md5": "a865d75ee01c1495799718aec920cd24",
      "description": "<p>O Castelo Templ&aacute;rio/Convento de Cristo foi sede da Ordem do Templo, at&eacute; 1314, e da Ordem de Cristo, a partir de 1357...</p>",
      "longitude": -8.41902494,
      "latitude": 39.60349765,
      "pois": [
        {
          "id": 1,
          "name": "Castelo dos Templários",
          "cover_image": "http://movtour.ipt.pt/rails/active_storage/blobs/eyJfcmFpbHMiOnsibWVzc2FnZSI6IkJBaHBWdz09IiwiZXhwIjpudWxsLCJwdXIiOiJibG9iX2lkIn19--0c62aaf42cc5b4143a9c193534f1a43b268fa019/CasteloDosTemplarios.jpg?locale=pt",
          "cover_image_md5": "2cd9e1666a6faf5408e5511bca60406d",
          "images": [
            {
              "image": "http://movtour.ipt.pt/rails/active_storage/blobs/eyJfcmFpbHMiOnsibWVzc2FnZSI6IkJBaHBBWDA9IiwiZXhwIjpudWxsLCJwdXIiOiJibG9iX2lkIn19--48109d9ebe11ff05e19a10fae4d8444d31556ca2/Castelo-de-Tomar-1.jpg?locale=pt",
              "image_md5": "7eff3203ceaaca94ad9740b3578d8cf5"
            },
            { … }, // 2 items
            { … }, // 2 items
            { … } // 2 items
          ],
          "longitude": -8.41765165,
          "latitude": 39.60355138,
          "poi_descriptions": [
            {
              "description_type_position": 1,
              "description_type_id": 3,
              "description_type_name": "Iniciação",
              "description_pt": "<p>Inicia&ccedil;&atilde;o. O Castelo Templ&aacute;rio/Convento de Cristo foi sede da Ordem do Templo, at&eacute; 1314, e da Ordem de Cristo, a partir de 1357. Do castelo (1160) faz parte a charola octogonal (finais do S&eacute;c. XII), santu&aacute;rio rom&acirc;nico de influ&ecirc;ncia oriental... </p>",
              "description_en": "<p>Initiation. The Templar Castle / Convent of Christ was the seat of the Order of the Temple until 1314, and the Order of Christ, from 1357. The castle (1160) includes the octagonal tray (late 12th century), Romanesque shrine of influence Eastern. This complex was convent 2 classified by UNESCO as World Heritage in 1983...</p>",
              "description_fr": "<p>Initiation. Le ch&acirc;teau des Templiers / Couvent du Christ &eacute;tait le quartier g&eacute;n&eacute;ral de Temple jusqu'&agrave; 1314, et l'Ordre du Christ, de 1357. Du ch&acirc;teau (1160) fait partie de la rotonde octogonale (la fin du si&egrave;cle. XII), sanctuaire roman d'influence oriental Ce complexe a &eacute;t&eacute; class&eacute; par l'UNESCO au patrimoine mondial de l'UNESCO en 1983...</p>",
              "description_de": "<p>Einleitung. Die Templerburg / Kloster von Christus war Temple Hauptquartier bis 1314 und der Orden Christi, von 1357. Von der Burg (1160) ist Teil des achteckigen Rotunde ist (das Ende des Jahrhunderts. XII), romanisches Heiligtum des Einflusses orientalisch Dieser Satz wurde von der UNESCO zum Weltkulturerbe Kloster 2classificado im Jahr 1983...</p>"
            },
            { … }, // 7 items
            { … }, // 7 items
            { … } // 7 items
          ],
          "beacons": [
            {
              "uuid": "25d3f51f-e930-466d-9253-a88b458af6bf"
            },
            {
              "uuid": "ad2b313d-f317-40bc-956d-becace9904ed"
            }
```


Figura 20 - Lista disponibilizada em formato JSON

5.1.2.6 Internacionalização

Para a internacionalização dos elementos estáticos do *backoffice* foi utilizada o módulo 'i18n' já incluído na *framework* Rails. Para tal, foram definidas as linguagens aceites pelo *backoffice* na configuração base da aplicação ('application.rb'), sendo o Português definido como a linguagem por defeito.

Com isto definido, sempre que se pretende introduzir texto internacionalizado nas vistas do *backoffice*, utiliza-se o método de tradução, t(…), do I18n, passando para este uma palavra-chave que identifica a frase a apresentar (ex.: *t('.palavra-chave')*).

Essas *palavras-chave* estão definidas em dois dicionários[5] criados para este propósito, um em Português e outro em Inglês, como ilustrado na Figura 21. A título de exemplo, a chamada para a função t('title') será substituída pela frase "Como instalar a aplicação" ou "How to install the application", dependendo da língua escolhida pelo utilizador.

```yaml
pt:
  pt: "Português"
  en: "Inglês"
  fr: "Francês"
  de: "Alemão"

  activerecord:
    attributes:
      monument:

  devise:
    sessions:
      new:
        login: "Entrar"
        password: "Palavra-passe"
  layouts:
    application:
      title: "Como Instalar a aplicação"
      1parag: "Nota: A aplicação está disponível apenas para dispositivos
      2parag: "Para instalar deve:"
```

```yaml
en:
  pt: "Portuguese"
  en: "English"
  fr: "French"
  de: "German"

  activerecord:
    attributes:
      monument:

  devise:
    sessions:
      new:
        login: "Login"
        password: "Password"
  layouts:
    application:
      title: "How to install the application"
      1parag: "Note: The application is available only for devices with Ar
      2parag: "To install it you should:"
```

Figura 21 - Dicionários de tradução

Sempre que o utilizador alterar a linguagem, o sistema automaticamente troca entre dicionários, alterando as palavras/frases estáticas do *backoffice*. Um diagrama deste processo é demonstrado na Figura 22. Esta preferência linguística é passada ao utilizador através do URL do site (http://movtour.ipt.pt/pt ou http://movtour.ipt.pt/en).

[5] Documentos em 'yml' que contêm palavas ou frases em cada idioma

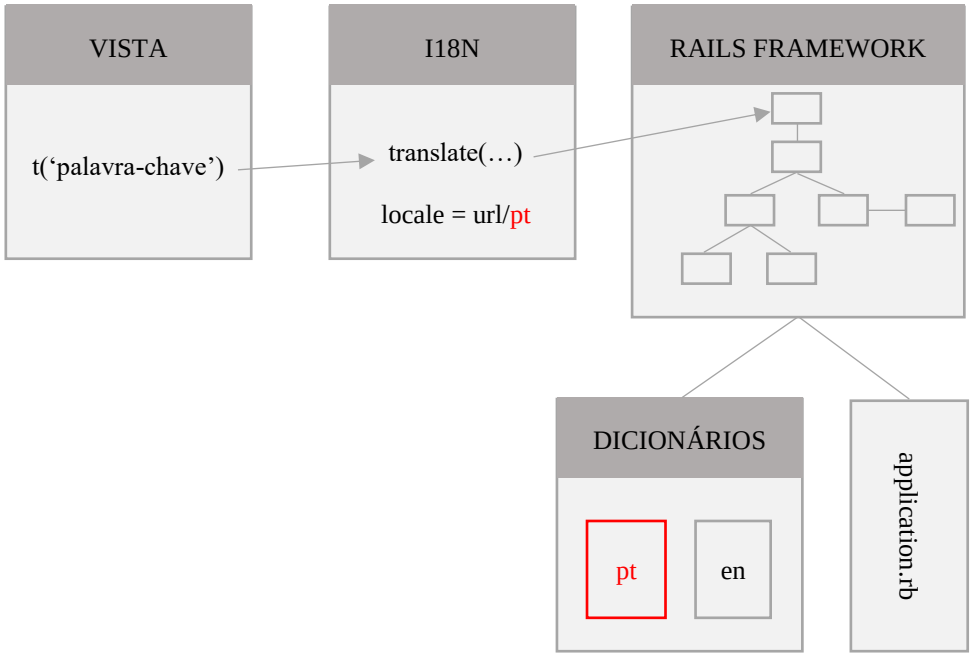

Figura 22 - Processo de tradução do I18n

5.1.2.7 Conteúdo multilingue

O *backoffice* permite que diversos tipos de conteúdo tais como as descrições dos monumentos e dos pontos de interesse, as categorias de descrição e categorias dos monumentos sejam definidas em diferentes idiomas. Atualmente estão definidos quatro idiomas, que são: Português, Inglês, Francês e Alemão.

Para este efeito foi utilizada a biblioteca ‘Globalize’, sendo criadas classes e tabelas de tradução associadas aos modelos que contêm atributos que necessitam de tradução. Por exemplo, o modelo ‘Category’ representa a categoria dos monumentos e contém o atributo ‘name’ que necessita de tradução. Em vez de guardar o atributo ‘name’ no modelo principal, esta informação é guardada no modelo de traduções associado, contendo o *locale* (língua) e o atributo ‘name’ na respetiva língua, tal como ilustrado na Figura 23.

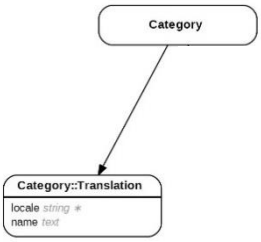

Figura 23 - Tabela de tradução das Categorias

Todo esse conteúdo fica presente no JSON (Figura 24) que é disponibilizado à aplicação móvel através da API e permite que a aplicação seja usada em diversos idiomas.

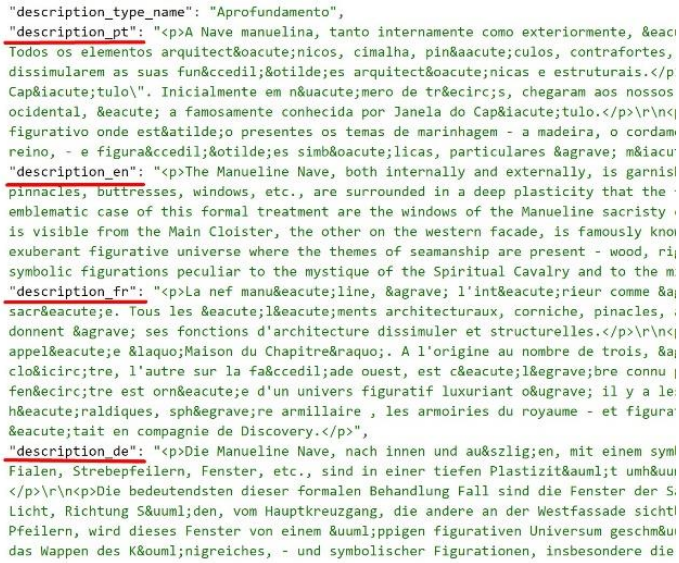

Figura 24 - Parte do JSON com as descrições

5.1.3 Bibliotecas (Gems)

Na grande maioria das linguagens de programação são utilizadas bibliotecas que permitem estender a funcionalidade base da linguagem e Ruby não é exceção. Em Ruby (e Ruby on Rails) as bibliotecas são denominadas de *gems*.

O RubyGems.org é um repositório destes pacotes Ruby que facilita a criação, partilha e instalação de *gems*. Esta instalação é automatizada no projeto através do *Bundle*, um gestor de pacotes para Ruby e ficheiro de configuração respetivo (*Gemfile*). As bibliotecas mais importantes utilizadas no desenvolvimento deste projeto são descritas na Tabela 2.

Tabela 2 - Bibliotecas do backoffice

Gem	Descrição
bootstrap	Acesso à *framework* bootstrap em alternativa a adicionar manualmente ficheiros JS e CSS [30].
capistrano, capistrano-rails, capistrano-passenger, capistrano-rvm	Capistrano [31] é uma biblioteca para automatizar todo o processo de *deployment*. Permite criar scripts para este fim. As restantes *gems* contém configurações específicas para tecnologias utilizadas tais como rails, o servidor *passenger* e o gestor de versões de ruby de nome rvm.
chartkick	Criação de gráficos e *dashboards* [32]

devise	Criação de contas de utilizadores [33]
faker	Geração de dados fictícios lógicos em teste e desenvolvimento, usado nas *seeds* (sementes – dados) [34]
font-awesome-rails	Utilização de ícones e imagens [35]
globalize	Biblioteca para tradução de dados dinâmicos [36]
groupdate	Fornece funções para agrupar dados temporais dos modelos [37]
highcharts-rails	Criação de gráficos e *dashboards* [38]
jquery-rails	Permite usar Jquery como biblioteca padrão de Javascript [39]
pg	Biblioteca para acesso ao SGBD PostgreSQL [40]
rails-erd, railroady	Utilizada para gerar diagramas a partir de modelos e controladores [41]
tinymce-rails	Integra o editor de texto TinyMCE para formulários (WYSIWYG) [42]

5.1.4 *Deploy* – Execução em Produção

Para a colocação do *backoffice* em produção utilizou-se uma *Virtual Machine* (VM) alojada no servidor do Instituto Politécnico de Tomar que onde foi instalado e configurado o seguinte conjunto de serviços:

- Ubuntu 16.04.3-LTS: sistema operativo;
- Rbenv: gerenciador de versões de Ruby;
- Ruby v2.5.x: interpretador da linguagem de programação utilizada;
- Rails: *framework* de desenvolvimento do backoffice;
- Bundler: Gestor de bibliotecas Ruby;
- Postgres: Base de dados SQL;
- Nginx: servidor HTTP que serve de *proxy* para o conteúdo estático como imagens ou o conteúdo dinâmico através do Passenger;

- Passenger: servidor de aplicações Ruby que permite executar aplicações *web* e APIs com excelente confiabilidade, desempenho e controlo. O Passenger é o responsável por todo o conteúdo dinâmico do *backoffice*;
- Redis: serviço de armazenamento de estruturas de dados em memória, tipicamente utilizado como servidor de bases de dados não relacionais (NoSQL), para *caching* ou para troca de mensagens (*broker*).

Fazer o *deployment*[6] de uma nova versão de uma aplicação complexa para um servidor de produção é um processo complexo e suscetível a falhas. Neste caso, é necessário copiar a nova versão do projeto para o servidor, instalar ou atualizar novas bibliotecas utilizadas, aplicar eventuais transformações (migrações) nas bases de dados, recompilar ficheiros de estilos e *javascript* (via *uglifier*) e reiniciar os serviços. Ao mesmo tempo ter o cuidado de manter a informação existente no sistema de dados e ficheiros de imagens. Para facilitar e automatizar o processo *deployment* para o servidor foi utilizado o Capistrano, uma biblioteca para o efeito. Depois de configurada, permite que com um único comando na máquina de desenvolvimento (*"cap production deploy"*) o servidor vá diretamente ao repositório (GIT) descarregar o projeto, execute eventuais migrações, instale novas bibliotecas, faça a otimização de imagens, ficheiros de JavaScript e compilação de ficheiros SASS em CSS, permitindo até fazer *rollback* – retroceder para uma versão anterior caso seja necessário.

A Figura 25 demonstra a colocação de uma das versões do projeto em produção.

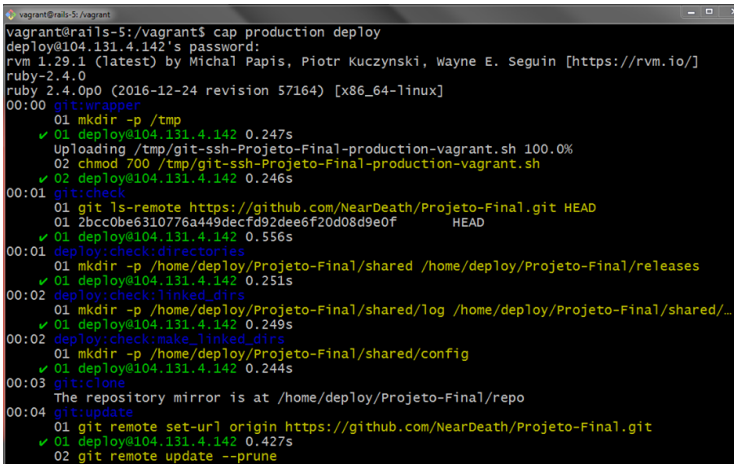

Figura 25 - Importação do projeto do GitHub para o Servidor

[6] Processo de colocação (implantação) de um sistema em funcionamento num ambiente de produção real.

5.1.5 Testes Automatizados

Para verificar a integridade do código e evitar a introdução de defeitos no *backoffice*, foram implementados testes automatizados. Para isto foi usada a biblioteca de testes 'Minitest'. Embora existam outros, tais como o 'RSpec', a biblioteca 'Minitest' é o sistema de testes padrão instalado no Rails, não sendo necessária configuração adicional e razão pela qual foi escolhido.

Os testes implementados encontram-se na pasta test/ do projeto e vão desde testes unitários aos modelos do sistema, até aos testes de integração de vários componentes. A grande vantagem de implementar e automatizar o processo de testes é a de garantir que algo continua a ser testado a cada nova versão, sendo mais fácil evitar alterações que levem à introdução de defeitos durante o desenvolvimento.

Este processo de automatização de testes do *backoffice* está, no entanto, dependente da execução dos mesmos por parte dos programadores. Para automatizar ainda mais este processo foi configurado um servidor de execução de testes, compilação e integração contínua através do serviço Travis-CI. Após ser configurado, a cada submissão de código para o git (*commit*), o servidor de testes vai descarregar esse mesmo código, instalar todo o ambiente necessário e executar todos os testes automatizados que estão implementados. Caso algum destes falhe esta informação será enviada por correio eletrónico para os responsáveis e o aviso aparecerá no GitHub.

Para além do Travis-CI, foi ainda configurado o serviço CodeClimate que permite fazer testes estáticos ao código. Este serviço analisa também o código a cada alteração no repositório e permite identificar falhas de segurança, duplicações de código ou outros problemas de estilo de formatação tanto em Ruby como em JavaScript. Esta ferramenta analisa ainda o resultado dos testes dinâmicos executados pelo Travis-CI, calculando a taxa de cobertura dos testes (percentagem de código que é executado pelos testes implementados).

A informação resultante destes testes automatizados, juntamente com a informação da análise estática de código e sobre as bibliotecas usadas está agrupada e apresentada sob a forma de logotipos (*badges*) na página do projeto no GitHub, tal como mostrado na Figura 26.

Figura 26 - Página principal do GitHub (retirado de [55])

5.2 Desenvolvimento da Aplicação Móvel

Como ferramenta para os turistas, foi desenvolvida uma aplicação móvel utilizando React-Native, que permitiu ter apenas uma base de código para diferentes plataformas (Android e iOS). Esta permite visualizar a distribuição de pontos de interesse no mapa, assim como a informação inerente a cada um desses pontos de interesse. Esta aplicação tem a particularidade de alertar o turista sempre que esse se encontrar num ponto de interesse, através de notificações geradas pela aplicação ao detetar um *beacon*.

5.2.1 Funcionamento da *framework* React-Native

No passado, as aplicações híbridas usavam exclusivamente JavaScript e WebViews[7]. Essas aplicações trabalhavam unicamente em HTML, CSS e JavaScript vindas da WebView, como demonstrado na Figura 27. Isto significava que, entre outras limitações, todo o processo de interação e renderização era controlado por uma única *thread* do interpretador de JavaScript. Este facto muitas vezes levava a que os componentes de JavaScript e HTML não acompanhassem a interação do utilizador com a aplicação, causando desfasamento, sensação de saltos entre a WebView, *bugs*, e por vezes até a terminação (*crash*) da aplicação. Portanto, apesar de ser possível criar aplicações através desse método, a perda de desempenho fazia com que a maioria dos programadores optasse por desenvolver utilizando linguagens nativas.

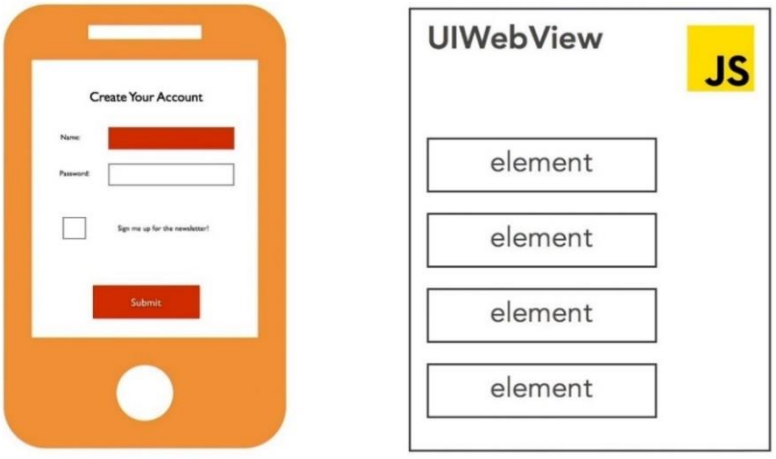

Figura 27 - Representação de uma single-thread que trabalha com as componentes HTML/CSS/JavaScript

[7] Componente que trabalha como se fosse um mini *browser* que está dentro da aplicação

React-Native por outro lado usa as componentes e APIs nativas de cada plataforma (Android e iOS), componentes essas que estão tipicamente escritas em Object-C e Swift para iOS e Java para Android. Nesta abordagem, o código desenvolvido em JavaScript corre na sua própria *thread*, sendo que as restantes componentes nativas trabalham independentemente deste. Não sofre assim dos problemas associados a uma abordagem *single-thread*, uma vez que os vários componentes e código comunicam assincronamente com o núcleo do JavaScript (como exemplificado na Figura 28). O resultado disso é uma renderização sem atrasos e uma experiência de utilização próxima ou igual a uma aplicação nativa, em parte por estar a utilizar os mesmos componentes.

Figura 28 - Representação de uma thread JavaScript que trabalha assincronamente com as componentes nativas

5.2.2 Componentes da aplicação móvel

Sendo a aplicação móvel destinada aos turistas, foi feito um esforço considerável a nível de design e UX (*User Experience*), de forma a criar um interface simples e que garantisse uma experiencia de utilização fácil e agradável (ver Figura 29). O objetivo foi reduzir ao mínimo possível a interação necessária por parte do utilizador na aplicação para usufruir ao máximo do que esta tem para oferecer: informação sobre os pontos de interesse.

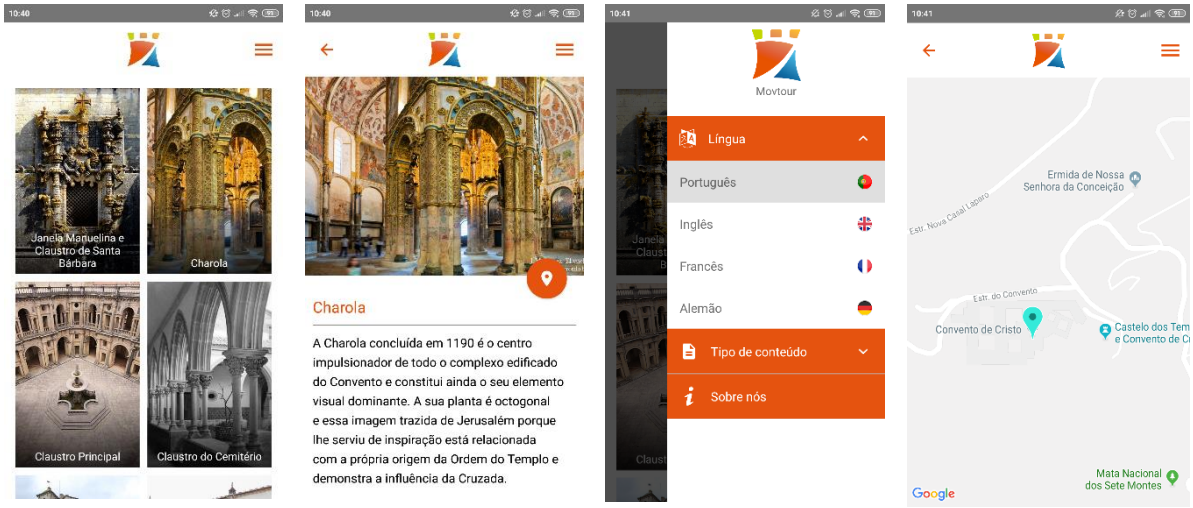

Figura 29 - Screenshots da aplicação MovTour

Nos pontos seguintes descreve-se cada uma das características que compõem a aplicação móvel.

5.2.2.1 Recolha de dados

Todos os dados que a aplicação móvel utiliza provêm da API disponibilizada pelo *backoffice*. Sempre que a aplicação é iniciada e caso possua internet, esta pede à API os dados necessários para a aplicação. A resposta vem em formato JSON, que é depois guardado de forma persistente no armazenamento do dispositivo, como ilustrado na Figura 30, por meio de uma biblioteca interna do React-Native, o AsyncStorage [43] (sistema que permite guardar pequenas quantidades de dados).

```
// -- FETCHING DATA ------------------------------------------- FE
fetch('http://movtour.ipt.pt/monuments.json', {timeout: 10 * 1000})
  .then(res => res.json())
  .then(res => {
    if(res.status == 500) {
      console.log("----------------------------------Erro 500-------
      Alert.alert(
        'Falha na recolha de dados',
        'Pedimos desculpa, mas de momento o servidor não está a envia
        [
          {text: 'Compreendi'},
        ],
      )
      this.getSavedData();
    } else {
      this.setState({data: res})
      // Vai guardar os dados na memória do telemóvel
      try {
        AsyncStorage.setItem('@Data', JSON.stringify(res));
        console.log("@Data saved");
      } catch (error) {
        console.log("Error saving data -> " + error);
      }
```

Figura 30 - Recolha dos dados

5.2.2.2 Guardar e apagar imagens

As hiperligações (URLs) para as imagens dos pontos de interesse estão presentes no ficheiro JSON que é descarregado ao iniciar a aplicação. Durante a primeira utilização, a aplicação descarrega e guarda as imagens no dispositivo utilizando uma biblioteca de terceiros (React-Native FS - permite efetuar leituras e escritas no *DocumentDirectory*[8]). Como este processo não é instantâneo existe na vista inicial um indicador do progresso da transferência.

De forma a poupar tráfego e tempo de transferência, no futuro a aplicação vai transferir uma nova versão da informação em JSON, só transferindo as imagens de monumentos e pontos de interesse caso estas tenham sido alteradas. Para isto, a aplicação vai comparar as informações do novo JSON com as imagens guardadas no dispositivo, verificando se as imagens são as mesmas através do uso de *hashes* MD5, como é demonstrado na Figura 31.

```
saveImages(){
  const x = this.state.data.monuments.length-1;
  const y = this.state.data.monuments[x].pois.length-1;
  // console.log("x: ", x, "; y:", y);
  return (
    this.state.data.monuments.map((i, indexMonuments) => {
      if (i.cover_image_md5 != undefined) {

      RNFS.exists(RNFS.DocumentDirectoryPath + '/images/' + i.cover_image_md5 + '.jpg')
        .then( success => {
          if (success == false){ // Se a imagem ainda não existir
            console.log("Vai gravar a imagem do monumento: ", i.cover_image_md5);
            this.setState({totalOfImages: this.state.totalOfImages+1, willDownload: true, showSpinner:false})
            RNFS.downloadFile({
              fromUrl: i.cover_image,
              toFile: `${RNFS.DocumentDirectoryPath}/images/` + i.cover_image_md5 + `.jpg`,
            }).promise.then(r => {
              console.log('Resposta ao guardar imagem do monumento: ', r);
              this.setState({
                imagesDownloaded: this.state.imagesDownloaded+1,
                progressBar:this.state.imagesDownloaded/this.state.totalOfImages
              })
              if(this.state.imagesDownloaded >= this.state.totalOfImages-1){
                this.setState({showProgressBar:false, willDownload: false})
                this.getMovtourBeacons();
              }
            })
            .catch((error) => {
              console.log("Erro ao descarregar imagem do monumento: ", error);
            })
          }
        })
```

Figura 31 - Função que guarda as imagens

[8] Pasta da aplicação que contém os dados do utilizador: /data/user/0/com.movtour/

Um processo semelhante é feito para eliminar imagens desatualizadas (imagens que foram alteradas no backoffice e, portanto, já não estão no JSON fornecido, mas que ainda estão no dispositivo móvel). A aplicação percorre todas as imagens guardadas no dispositivo, comparando-as com a informação das imagens que está no JSON. Qualquer imagem do dispositivo que já não esteja em utilização é automaticamente eliminada, mantendo assim toda a informação atualizada.

5.2.2.3 Modo off-line

Como visto no ponto anterior, a aplicação guarda toda a informação necessária para o seu bom funcionamento. Isso permite que a aplicação funcione quando o utilizador não possui ligação de dados (exceto na primeira utilização), pois vai reutilizar os dados que foram guardados na última utilização com ligação de dados. Para isso, utiliza-se o mecanismo *AsyncStorage.getItem('<palavra-chave>')*. Esse método recolhe todos os dados que dizem respeito à *'palavra-chave'*, inserindo-os de seguida numa variável de estado, que vai ser utilizada pela aplicação (Figura 32).

```
getSavedData(){
  return (
    AsyncStorage.getItem('@Data',(err, dados) => {
      if(err) {
        console.error('Error loading monuments', err)
      } else {
        if(!dados){
          console.log('Os dados estão vazios');
          Alert.alert(
            'Dados insuficientes',
            'Por favor reinicie a aplicação e ligue a internet para a aplicação descarregar os dados necessários. Obrigado.',
            [
              {text: 'Ok, fechar aplicação.', onPress: () => RNExitApp.exitApp()},
            ],
          )
        } else {
          console.log('Vai carregar os dados que estão na memória')
          const monuments = JSON.parse(dados)
          this.setState({
            data: monuments,
            showProgressBar: false,
            showSpinner: false,
          })
          this.getMovtourBeacons();
        }
      }
    })
  )
}
```

Figura 32 - Recolha dos dados guardados

Assim que o utilizador possuir internet e a aplicação for reiniciada, os dados para o seu bom funcionamento serão atualizados (caso necessário).

5.2.2.4 Deteção de *Beacons*

A ligação entre o dispositivo do turista e um *beacon* é feita através da tecnologia Bluetooth Low Energy (BLE). Ao contrário de soluções como a utilização do sinal de GPS, esta solução permite que a aplicação funcione mesmo dentro de edifícios, alertando o utilizador sempre que um *beacon* presente do ficheiro JSON é detetado. Além disso, a utilização do BLE permite que se economize bastante bateria em comparação com outras tecnologias, como o GPS.

Para a deteção de *beacons* foi utilizada a biblioteca 'react-native-beacons-manager'. Quando a aplicação inicia, é chamado o método que procura especificamente os *beacons* Movtour, como é demonstrada na Figura 33.

```
Beacons.detectIBeacons();

Beacons
 .startRangingBeaconsInRegion({identifier: 'Movtour', uuid:null})
 .then(() => console.log('Beacons ranging started succesfully'))
 .catch(error => console.log(`Beacons ranging not started, error: ${error}`));
```

Figura 33 - Deteção de beacons

A partir desse momento um evento é despertado quando é detetado um *beacon* (é necessário que o Bluetooth esteja ativo no dispositivo). No entanto, o processo que segue à deteção depende do estado da aplicação:

- *Foreground*: Quando a aplicação móvel está em primeiro plano e essa deteta um *beacon*, é automaticamente aberta a vista que contém a informação do ponto de interesse associado àquele *beacon*.
- *Background*: No caso da aplicação móvel estar em segundo plano e detetar um *beacon*, a aplicação alerta o utilizador através de uma notificação com o nome do ponto de interesse, sendo apenas necessário tocar na notificação para abrir a aplicação na página que contém a informação relativa ao ponto de interesse associado.

O processo de deteção de *beacons* (Figura 34) utiliza vários dados que os *beacons* transmitem, como o seu UUID e distância. Quando um *beacon* é detetado, é verificado se o seu UUID está presente na lista de *beacons* do Movtour. Caso isto seja verdade, então é efetuado um dos dois processos descritos acima, dependendo do estado da aplicação (se está

em primeiro plano ou segundo plano). Na possibilidade de ser detetado mais do que um *beacon* ao mesmo tempo, então a aplicação irá verificar qual o que se encontra mais perto. Após identificar o que está mais próximo é seguido um dos dois processos acima mencionados.

```
// Beacons events
DeviceEventEmitter.addListener('beaconsDidRange',(data) => {
    // Se for detectado algum beacon, verifica se é dos Movtour e encontra qual o mais próximo, adicionando-o à variável closerBeacon.
    if (data.beacons.length > 0){
      let movtourBeacons = data.beacons.filter(beacon => this.state.movtourBeacons.includes(beacon.uuid));
      this.setState({detectedBeacons: movtourBeacons});

      // Encontra o beacon que está mais perto
      let tempBeacon = movtourBeacons.find(x=> x.distance == Math.min(...movtourBeacons.map( y => y.distance)));

      if(!this.state.closerBeacon || tempBeacon.distance < this.state.closerBeacon.distance){
        if (this.state.data.monuments !== undefined){
          this.state.data.monuments.map(i => i.pois.map(j => j.beacons.map(b => {

            if (tempBeacon.uuid == b.uuid){
              let poi_id = j.id;
              this.setState({
                closerBeacon:{uuid: tempBeacon.uuid, poi: poi_id}
              });
            }
          })))

        }
      }
    } else { //Se não for detectado nada atualiza a variavel detectedBeacons
      this.setState({detectedBeacons: null});
    }
```

Figura 34 - Processo de deteção de beacons

5.2.2.5 Internacionalização

Atualmente a aplicação móvel Movtour suporta a internacionalização de conteúdo estático e dinâmico, através da biblioteca I18n, para quatro idiomas: Português, Inglês, Francês e Alemão.

Quando a aplicação inicia, verifica o idioma do sistema operativo e se for um dos quatro idiomas suportados, altera o conteúdo (estático e dinâmico) em conformidade. Caso o idioma do sistema operativo não for nenhum dos suportados pela aplicação móvel, é utilizado por defeito o Inglês.

Para a tradução de conteúdo estático é necessário chamar a função de tradução nas vistas dos componentes (ex.: *I18n.t('<palavra-chave>')*). Essa função recolhe a tradução correspondente à *palavra-chave* e ao idioma selecionado/detetado. Essa tradução está presente nos dicionários de cada idioma, como demonstrado na Figura 35.

```
I18n.translations = {
  pt: {
    map: 'Mapa',
    monuments: 'Monumentos',
    convent: "Convento",
    museum: "Museus",
    church: "Igrejas",
    other: "Outros Locais",
  },
  en: {
    map: 'Map',
    monuments: 'Monuments',
    convent: "Convent",
    museum: "Museum",
    church: "Church",
    other: "Other Places",
  },
  fr: {
    map: 'Carte',
    monuments: 'Monuments',
    convent: "Couvent",
    museum: "Musée",
    church: "Église",
    other: "D'autres lieux",
  },
  de: {
    map: 'Karte',
    monuments: 'Monumente',
    convent: "Kloster",
    museum: "Museum",
    church: "Kirche",
    other: "Andere Orte",
  }
```

Figura 35 - Dicionários I18n

O utilizador tem ainda a opção de substituir o idioma que pretende, utilizando para isso a página de opções da aplicação móvel. Ao selecionar uma opção, essa será guardada na memória do dispositivo móvel, através do *AsyncStorage*, para que nas futuras utilizações o utilizador tenha a aplicação no idioma preferencial.

5.2.2.6 Níveis de conteúdo turístico

Como referido na secção 5.1.2.2, a solução implementada suporta diferentes níveis de conteúdo do património natural e cultural. É através da página de definições da aplicação móvel que é possível alterar o nível de conteúdo que é apresentado em cada ponto de interesse. Tal como o idioma, a aplicação móvel guarda esse nível na memória do dispositivo para que numa futura utilização o nível se mantenha.

5.2.2.7 Envio de dados para estatística

Um dos objetivos principais do projeto é a recolha de dados para efeitos de estatística. Sempre que um utilizador entra num ponto de interesse, o seu dispositivo móvel deteta o(s)

beacon(*s*) associados a esse ponto e envia dados sobre o dispositivo móvel para o *backoffice* através da API, tal como demonstrado na Figura 36. Isso permite que se possa estudar esses dados posteriormente, dando assim a possibilidade de melhorar o serviço turístico.

```
postFunction(k){
  fetch('http://movtour.ipt.pt/accesses', {
    method: 'POST',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      date: moment().format(),
      poi_id: k.id,
      nationality: DeviceInfo.getDeviceCountry(),
      id_device: DeviceInfo.getUniqueID(),
      os: DeviceInfo.getSystemName(),
      brand: DeviceInfo.getBrand(),
      model: DeviceInfo.getModel(),
    })
  })
}
```

Figura 36 - Envio dos dados recolhidos

5.2.2.8 Distribuição dos monumentos pelo mapa

A aplicação móvel disponibiliza uma vista, através da biblioteca 'React-Native-Maps' e a API da Google, onde se pode visualizar todos os pontos de interesse distribuídos pelo mapa.

Essa vista utiliza as coordenadas de cada ponto de interesse para os distribuir pelo mapa, através de marcadores (Figura 37). Esses marcadores ao serem pressionados encaminham a aplicação móvel para a vista que detalha o ponto de interesse pressionado.

```
<MapView.Marker
  coordinate={{
    latitude: j.latitude,
    longitude: j.longitude,
  }}

  pinColor='#009688'
  key={j.name}
>
  <MapView.Callout tooltip style={styles.callout} onPress={
    () => navigate({
            routeName: 'MonumentDetails',
            params: {monumento:i, poi:j, description_types:data.categories},
            key:'detail'
          })
  }>
```

Figura 37 - Inserção dos marcadores no mapa

5.2.2.9 Gerar Android *Package* (APK) para distribuição

Para distribuir a aplicação final é necessário compilar a mesma em modo *release*, gerando um APK assinado digitalmente (ver Figura 38). Para esse efeito, o primeiro passo foi gerar uma chave. Para isso é utilizado o *keytool* (ferramenta que gere chaves e certificados) presente no Java Development Kit (JDK) e executado o seguinte comando:

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias my-key-alias -
keyalg RSA -keysize 2048 -validity 10000
```

Figura 38 - Comando para criar uma chave assinada

De seguida, move-se a chave acabada de gerar para dentro do projeto, sendo necessário depois alterar a configuração da aplicação no ficheiro *gradlew.properties* e *build.gradle* para utilizar a chave. Por fim, corre-se um comando específico do React-Native para gerar a APK, como demonstrado na Figura 39.

```
$ cd android && ./gradlew assembleRelease
```

Figura 39 - Comando para gera um APK

5.2.3 Bibliotecas utilizadas

Tal como referido na secção 5.1.3, na maioria das linguagens de programação (e *frameworks*) são utilizadas bibliotecas que permitem estender a funcionalidade base destas. React-Native não foge a regra, sendo nesta também necessário utilizar bibliotecas adicionais para aceder a certas funcionalidades. A título de exemplo, a *framework* em causa não oferece de base mecanismos simples (através da sua API) para interação com dispositivos Bluetooth e a utilização deste tipo de hardware tem por norma diferenças entre plataformas (Android e iOS). Deste modo, foi necessário utilizar uma biblioteca adicional para obter esta funcionalidade e harmonizar o acesso a esta entre diferentes sistemas. Na Tabela 2 apresentamos as bibliotecas que foram utilizadas na aplicação móvel.

Tabela 3 - Bibliotecas da aplicação mobile

Biblioteca	Descrição
Navigation	Navegação para o react-native [44]
Maps	Biblioteca para uso do mapa da google [45]
Push Notification	Para notificações locais e remotas [46]
Beacon Manager	Biblioteca para detetar *beacons* [47]
Device Info	Biblioteca para obter informação do dispositivo [48]
FS	Para obter acesso ao sistema de ficheiros do dispositivo (ex.: gravar, carregar dados) [49]

6 Conclusões

Este projeto teve como objetivo principal o desenvolvimento de uma solução capaz de mitigar os problemas relacionados com o acesso e recolha de informação turística. Esta lacuna existe tanto para o turista que visita a cidade como também para as entidades responsáveis pelo turismo. O sistema resultante deste trabalho permite aos turistas receber informação sobre monumentos e pontos de interesse de forma facilitada e gratuita através do seu dispositivo móvel. Por outro lado, o gestor de turismo poderá obter estatísticas e tendências sobre os turistas. Tais informações eram até este ponto desconhecidas ou muito limitadas, sendo agora possível extrair conhecimentos que permitem por exemplo otimizar o serviço oferecido.

O *backoffice* encontra-se em funcionamento estando alojado num servidor do Instituto Politécnico de Tomar. A aplicação móvel foi desenvolvida tanto para dispositivos Android como para Apple iOS, estando a versão Android já disponível no Google Play. Foram também iniciados os primeiros testes experimentais do sistema em ambiente real, consistindo em testes de captura de sinais dos *beacons* no Convento de Cristo, em Tomar. Está previsto para o futuro próximo a instalação de *beacons* nos pontos de interesse do Convento, iniciando-se de seguida os primeiros testes do sistema MovTour num ambiente real que é visitado anualmente por aproximadamente 350 mil pessoas [50].

No entanto e como em qualquer outro projeto, ainda mais em contexto académico, mesmo que os objetivos principais tenham sido atingidos, existem algumas funcionalidades e melhorias a ser implementadas em versões futuras. Algumas destas são a disponibilização da aplicação móvel para iOS na Apple Store; a deteção de utilizadores nos monumentos através de WiFi; a recolha de *feedback* por parte do utilizador no fim da visita; melhorias ao sistema de mapas da aplicação, por exemplo com a disponibilização de rotas entre o utilizador e o monumento que deseja visitar; disponibilização de informação ao turista do local em que este se encontra (dentro do monumento), através do uso de *beacons* e plantas dos monumentos; e por fim, suporte para guias de áudio.

7 Bibliografia

[1] "Digital in 2017: Global Overview," We Are Social, 2017. [Online]. Available: https://wearesocial.com/special-reports/digital-in-2017-global-overview. [Acedido em 09 2018].

[2] INE, "Instituto Nacional de Estatística," 2018. [Online]. Available: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_publicacoes&PUBLICACOESpub_boui=320462327&PUBLICACOESmodo=2. [Acedido em 23 10 2018].

[3] "Ruby on Rails," Ruby on Rails, [Online]. Available: http://rubyonrails.org/.

[4] "Bootstrap," GetBootstrap, [Online]. Available: http://getbootstrap.com/.

[5] "jQuery - write less, do more," The jQuery Foundation, [Online]. Available: https://jquery.com/.

[6] R. Safitri, D. S. Yusra, D. Hermawan, E. Ripmiatin e W. Pradani, *Mobilte Tourism Application Using Augmented Reality,* 2017.

[7] A. Smirnov, A. Kashevnik, A. Ponomarev, M. Shchekotov e K. Kulakov, *Application for e-Tourism: Intelligent Mobile Tourist Guide,* 2015.

[8] G. Sato, G. Hirakawa e Y. Shibata, *Push Typed Tourist Information System based on Beacon and Augumented Reality Technologies,* 2017.

[9] M. Kenteris, D. Gavalas e A. Mpitziopoulos, *A Mobile Tourism Recommender System,* 2010.

[10] "Tripadvisor," Tripadvisor, [Online]. Available: https://www.tripadvisor.pt/.

[11] D. Gravity, "Descubra Tomar," Google Play, [Online]. Available: https://play.google.com/store/apps/details?id=pt.descubra.tomar.

[12] "Discover your destination," Lonely Planet, [Online]. Available: https://www.lonelyplanet.com/guides.

[13] D. M. Selfa, M. Carrillo e M. d. R. Boone, *A Database and Web Application Based on MVC Architecture,* 2006.

[14] “The web framework for perfectionists with deadlines,” django, [Online]. Available: https://www.djangoproject.com/.

[15] “The PHP framework for web artisans,” Laravel, [Online]. Available: https://laravel.com/.

[16] “ASP.NET,” Microsoft, [Online]. Available: https://www.asp.net/.

[17] M. Bächle e P. Kirchberg, *Ruby on Rails,* 2007.

[18] “Foundation - The most advanced responsive front-end framework in the world,” ZURB Foundation, [Online]. Available: http://foundation.zurb.com/. [Acedido em 12 2017].

[19] “Ferramentas do Visual Studio para Xamarin,” Visual Studio Microsoft, [Online]. Available: https://visualstudio.microsoft.com/pt-br/xamarin/. [Acedido em 12 2017].

[20] P. Reichelt, “Why I left Xamarin behind in favor of React Native,” Medium, 15 11 2016. [Online]. Available: https://medium.com/@reicheltp/dev-diary-4-why-i-left-xamarin-behind-35817ccd07b2. [Acedido em 01 2018].

[21] “React Native - Build native mobile apps using JavaScript and React,” Facebook React Native, [Online]. Available: https://facebook.github.io/react-native/.

[22] “React - A Javascript library for building user interfaces,” ReactJS, [Online]. Available: https://facebook.github.io/react/.

[23] “Maiores Repositórios do GitHub,” Github, [Online]. Available: https://github.com/search?o=desc&q=stars%3A%3E1&ref=searchresults&s=stars&type=Repositories&utf8=%E2%9C%93.

[24] “Who's using React Native,” Facebook React Native, [Online]. Available: https://facebook.github.io/react-native/showcase.html.

[25] M. Siekkinen, M. Hiienkari, J. K. Nurminen e J. Nieminen, “How Low Energy is Bluetooth Low Energy? Comparative Measurements with ZigBee/802.15.4,” 2012.

[26] “Bluetooth Low Energy,” Wikipedia, 13 09 2018. [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth_Low_Energy.

[27] “Development Environments Made Easy,” HashiCorp Vagrant, [Online]. Available: https://www.vagrantup.com/.

[28] Virtual Box, [Online]. Available: https://www.virtualbox.org/.

[29] “Version Control System,” Git, [Online]. Available: https://git-scm.com/.

[30] “Bootstrap 4 rubygem for Rails,” GitHub, [Online]. Available: https://github.com/twbs/bootstrap-rubygem.

[31] “Remote multi-server automation tool,” GitHub, [Online]. Available: https://github.com/capistrano/capistrano.

[32] Ankane, “Create beautiful Javascript charts with one line of Ruby,” GitHub, [Online]. Available: https://github.com/ankane/chartkick.

[33] “Devise - Flexible authentication solution for Rails with Warden,” GitHub, [Online]. Available: https://github.com/plataformatec/devise.

[34] B. Curtis, “A library for generating fake data such as names, addresses, and phone numbers,” GitHub, [Online]. Available: https://github.com/stympy/faker.

[35] D. Bock, “The font-awesome font bundled as an asset for the rails asset pipeline,” GitHub, [Online]. Available: https://github.com/bokmann/font-awesome-rails.

[36] “Rails I18n de-facto standard library for ActiveRecord model/data translation,” GitHub, [Online]. Available: https://github.com/globalize/globalize.

[37] A. Kane, “The simplest way to group temporal data,” GitHub, [Online]. Available: https://github.com/ankane/groupdate.

[38] P. C. B. Viken, “Simple gem to include Highcharts in a Rails,” GitHub, [Online]. Available: https://github.com/PerfectlyNormal/highcharts-rails.

[39] “A gem to automate using jQuery with Rails,” GitHub, [Online]. Available: https://github.com/rails/jquery-rails.

[40] “PostgreSQL,” RubyGems, [Online]. Available: https://rubygems.org/gems/pg.

[41] “Generate Entity-Relationship Diagrams for Rails applications,” GitHub, [Online]. Available: https://github.com/voormedia/rails-erd.

[42] S. Pohlenz, “Integration of TinyMCE with the Rails asset pipeline,” GitHub, [Online]. Available: https://github.com/spohlenz/tinymce-rails.

[43] “AsyncStorage,” Facebook React Native, [Online]. Available: https://facebook.github.io/react-native/docs/asyncstorage.html.

[44] “Routing and navigation for your React Native apps,” ReactNavigation, [Online]. Available: https://reactnavigation.org/.

[45] “React Native Mapview component for iOS + Android,” GitHub, [Online]. Available: https://github.com/airbnb/react-native-maps.

[46] J. Trujillo, “React Native Local and Remote Notifications,” GutHub, [Online]. Available: https://github.com/zo0r/react-native-push-notification.

[47] E. Datin, “React-Native library for detecting beacons,” GitHub, [Online]. Available: https://github.com/MacKentoch/react-native-beacons-manager.

[48] B. Hughes, “Device Information for React Native,” GitHub, [Online]. Available: https://github.com/rebeccahughes/react-native-device-info.

[49] H. Hübel, “Native filesystem access for react-native,” GitHub, [Online]. Available: https://github.com/itinance/react-native-fs.

[50] M. R. Fonseca, “Tomar - Convento de Cristo teve 350 mil visitantes em 2017,” mediotejo, 13 01 2018. [Online]. Available: http://www.mediotejo.net/tomar-convento-de-cristo-teve-350-mil-visitantes-em-2017-um-novo-recorde-de-turistas/.

[51] TripAdvisor, “Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=com.tripadvisor.tripadvisor. [Acedido em Janeiro 2018].

[52] D. Tomar, “Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=pt.descubra.tomar&hl=pt_PT. [Acedido em 01 2018].

[53] G. b. L. Planet, “Google Play,” [Online]. Available: https://play.google.com/store/apps/details?id=com.lonelyplanet.guides. [Acedido em 01 2018].

[54] R. Peres, “Model-View-Controller (MVC) in iOS: A Modern Approach,” [Online]. Available: https://www.raywenderlich.com/1073-model-view-controller-mvc-in-ios-a-modern-approach. [Acedido em 02 2018].

[55] Movtour, “Movtour - Informação sobre turismo em Tomar,” [Online]. Available: https://github.com/NearDeath/Movtour. [Acedido em 11 2018].

[56] oPortoInsight, “oPortoInsight,” [Online]. Available: http://oportoinsight.com/. [Acedido em 12 2017].