UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Play JBT – Mobile Application for the Tropical Botanical Garden of Lisbon

Stefan Postolache

Mestrado em Informática

Trabalho de Projeto orientado por:
Prof. Doutora Ana Paula Afonso
e Prof. Doutor António Manuel Silva Ferreira

2019

Acknowledgements

This thesis would not have been possible without the support and patience from Prof. Doctor Ana Paula Afonso and Prof. Doctor António Manuel Silva Ferreira. Overcoming the challenges related to development of the information system presented in this Master Thesis would not have been possible without their support and understanding. I am very grateful for their supervision.

Next, I need to thank the Rectory of the University of Lisbon, for the opportunity to be part of this project and the research grant, and the LASIGE centre for the space provided for the execution of this thesis.

My special thanks to Prof. Maria Dulce Pedroso Domingos and Prof. José Manuel Pinto Paixão, as it was with their management of the different parts involved in this project that this application was successfully completed. I would additionally like to express my gratitude to all DI FCUL team members, Rafael Torres, Prof. Maria Beatriz do Carmo and Prof. Ana Paula Cláudio that contributed for the realization of this work. In addition, I must also give my thanks to Cristina Duarte, César Garcia, Ana Leal, Ana Godinho, Raquel Barata, Paula Redweik, Palmira Carvalho and Cecília Sérgio and Tiago Ribeiro. Without their expertise and time for invaluable directions this project would not have been possible. I am deeply and sincerely grateful for all of your help.

I must also sincerely thank my great friends and partners, Ana Nunes, Catarina Cavique, Pedro Carvalho, Ricardo Subtil and Ricardo Jorge, for all the good times and experiences we've shared from the first year of the Bachelor's all the way up to this moment, and for giving me strength to go on.

Finally, I want to thank Nuno Henriques, for all the support and the mentally stimulating discussions on programming and philosophy.

Thank you all so much.

Dedicatória.

Resumo

Com o progresso das tecnologias de informação e comunicação (TIC), as instituições culturais diversificaram as modalidades de interação com as pessoas. TIC permite hoje as várias instituições culturais de assumir papeis diferentes perante a comunidade (por exemplo, educação dos cidadãos e das suas associações; formador de várias competências; e de perito em vários programas governamentais para desenvolvimento de comunidades). Neste documento está apresentado o trabalho de desenvolvimento de uma aplicação móvel para Jardim Botânico Tropical de Lisboa. Técnicas diversas foram utilizadas no desenvolvimento de aplicação móvel (por exemplo, entrevistas, listagem de conteúdos, prototipagem, avaliação heurística, testes de usabilidade). São apresentados detalhes das tecnologias usadas (software e hardware), procedimentos de implementação, como também sobre arquitetura final do sistema desenvolvido. A aplicação móvel permite aos visitantes de Jardim Botânico Tropical interagir de formas diferentes com os componentes de jardim (plantas, aves e edifícios). Vários recursos educativos são incluídos na aplicação de modo a ser adaptados de modo automático ao perfil do utilizador. A aplicação permite também captar e armazenar os dados produzidos por utilizadores da aplicação de modo a serem utilizados para melhoria de experiência dos visitantes do jardim. Vários serviços Web foram incluídos para melhorar apresentação dos conteúdos e para melhorar os serviços do jardim. Foram também realizados testes com peritos no jardim e recolhido feedback dos utilizadores dos quais recebemos boas críticas e sugestões que foram integradas na aplicação. Foram também realizados um conjunto de testes de desempenho do servidor.

Palavras-chave: Aplicações móveis, Jardins botânicos, Percursos.

Abstract

Through the progress of information and communication technologies (ICT), cultural institutions have diversified the modalities of interacting with people. Today, ICTs allow various cultural institutions to take on different roles in the community (e.g. educating citizens and their associations; shaping various skills; supporting government programs for community development). This document introduces the process of development of a mobile application, which acts mainly as a helping guide for visitors of the Lisbon Tropical Botanical Garden. This mobile application allows these visitors to interact in different ways with garden components (plants, buildings and birds), as well as to have access to the several educational resources included in it, which are to be adapted to the user's profile. The application also allows them to capture and store the data produced, data which is also used for help with improving garden services. Web services have been developed to provide content and to centrally store data on the visitor's trajectory in the garden and demographics. Furthermore, various techniques were used in the process of development (e.g. interviews, content listing, prototyping, heuristic evaluation, usability testing). Details on the technologies used (software and hardware), implementation procedures, as well as the final architecture of the developed system will be demonstrated. Finally, a set of usability tests is presented, from which we received positive feedback from the users as well as the performance tests executed on the server.

Keywords: Mobile applications, Botanical gardens, Tours.

Resumo alargado

Com o progresso das tecnologias de informação e comunicação (TIC), as instituições culturais diversificaram as modalidades de interação com as pessoas. Atualmente, as TIC permitem às várias instituições culturais assumir papéis diversos perante a comunidade (por exemplo: educação dos cidadãos e das suas associações; formação de competências; peritagem em vários programas governamentais para desenvolvimento de comunidades). Neste documento é documentado o processo de desenvolvimento de uma aplicação móvel para o Jardim Botânico Tropical de Lisboa.

Os objetivos deste projeto de mestrado foram: design e implementação de uma aplicação móvel para o Jardim Botânico Tropical de Lisboa, que integra um conjunto percursos, adaptados para diferentes perfis de utilizadores; criação de uma aplicação para servidor, encarregue de armazenar conteúdos e dados utilizados na aplicação, como também dados produzidos pelo utilizador; e avaliação do sistema implementado.

Numa primeira fase, foi feito um trabalho de investigação sobre a literatura existente, na base de dados ACM sobre aplicações para jardins botânicos, museus e guias de cidades e estudadas um conjunto de aplicações disponíveis nas lojas *Google Play* e *App Store*. A estrutura, propriedades e funcionalidades das aplicações, como também as suas diferenças, vantagens, desvantagens e limitações foram identificadas e analisadas, resultando desta análise uma grande variedade de perspetivas sobre como se deve estruturar uma aplicação móvel para este tipo de instituições. Para além disto, a análise das aplicações para os museus permitiu a documentação de diversos tipos de interações, as quais podem ser usadas nas plataformas digitais das instituições culturais.

Nesta fase também foi feito o levantamento de requisitos. Para este efeito, vários peritos do Jardim Botânico Tropical de Lisboa foram consultados, no início do projeto de desenvolvimento da aplicação e durante a avaliação dos protótipos. De seguida, para a elaboração dos protótipos iniciais de baixa fidelidade, foi utilizado o *software Balsamiq Wireframes*.

Já durante a implementação de um protótipo funcional, para a implementação da base de dados local, foram utilizados o *SQLite* e *Room Persistence Library*. A *Room Persistence Library* foi escolhida, para resolver alguns problemas de segurança dos dados, e para simplificar o acesso a estes. Para haver sincronização dos objetos com a base de dados, foi integrado *LiveData*, uma estrutura de dados *lifecycle-aware* e observável.

Para representação dos percursos na aplicação, um mapa do jardim botânico foi utilizado. Vários *SDKs* para mapas interativos foram considerados para a implementação do mapa interativo: *Google Maps*; *Mapbox*; *NextGIS*; *Osmdroid* e *Maps.me*. No entanto foi adotada uma solução baseada numa imagem do mapa que permite aplicação e movimento do mapa implementado de raiz.

O *web server* utilizado foi *NGINX*, considerando a sua alta performance, e a sua escalabilidade, que permite milhares de acessos simultâneos, sem um impacte considerável na velocidade de resposta.

Na fase seguinte, para implementar a base de dados remota foi utilizado *PostgreSQL*, com a extensão *PostGIS* (extensão de suporte a tipos geográficos), de modo a armazenar dados de

percursos e seus pontos de interesse e os conteúdos da aplicação. Para visualização dos dados geográficos, foi utilizado o sistema de informação geográfica *QGIS.*

Foram implementados também serviços *web* para aceder a dados destes percursos e seus pontos de interesse, desenvolvidos utilizando *Lumen, PHP micro-framework*, baseada em componentes da *framework Laravel*. Estes serviços são acedidos pela aplicação através da biblioteca *Retrofit* que converte a *API REST* utilizada para aceder aos serviços numa interface Java. Os serviços web foram testados, com uso do browser *Mozilla Firefox*, e a ferramenta de teste de *APIs, Postman.* A ferramenta *logcat* do *Android Studio* foi também utilizada para visualização de mensagens que a aplicação envia para o *IDE* quando a aplicação é instalada num um dispositivo real ou virtual, tais como sobre pedidos HTTP.

Para o controlo das versões de software foi utilizado *Git*, e como repositório central remoto foi utilizado o *GitHub service Pro version*, que permite a criação de um repositório privado. A aplicação móvel foi desenvolvida em *Java*, com utilização do *Android Studio IDE*.

A aplicação foi desenvolvida em computadores pessoais (correndo o sistema operativo Windows), e testada num conjunto de cinco dispositivos móveis com características distintas. Neste relatório estes dispositivos são listados juntamente com as suas especificações.

Para identificar problemas e melhorar a aplicação, foi feita a avaliação de vários protótipos, cada versão resolvendo problemas identificados em avaliações prévias como resultado do uso da técnica de test-fix-test-fix. A equipa de projeto para esta aplicação incluiu especialistas de vários domínios: representação geográfica, botânica, ornitologia, história e informática. Cada trajeto turístico incluído na aplicação foi desenvolvido com a colaboração dos investigadores da respetiva área de especialização. A avaliação da interface da aplicação foi feita em várias etapas, com a ajuda de peritos do jardim botânico e membros da equipa de projeto para detetar os problemas de usabilidade e melhorar a aplicação. Como resultado destes testes também foram corrigidas as posições dos pontos de interesse recorrendo ao QGIS.

Foram realizados também testes de usabilidade com participantes do Encontro Ciência 2019. Nestes testes, foi feita uma observação dos participantes ao usarem a aplicação, para se poder entender de que modo estes realizam as diferentes tarefas incluídas na exploração de um percurso. Tentaram-se identificar também as diferentes emoções sentidas, tais como surpresa, confusão e satisfação, como também a ocorrência de dificuldades no uso. Em geral, os participantes demonstraram interesse e deram *feedback* positivo quanto à qualidade de aplicação, sugerindo algumas alterações para melhoria dos processos na aplicação, que foram documentadas. Os resultados de todos os testes e coleção do feedback dos utilizadores, como também a origem de erros e soluções para correção estão todos descritos na discussão desta tese.

São apresentadas também estimativas do desempenho do servidor, considerando acessos à aplicação de números elevados de utilizadores, que revelam um crescimento linear na resposta, sendo apresentadas funções para esse crescimento resultantes da modelação dos dados colecionados.

Em conclusão, foi implementada a primeira aplicação móvel para o *Jardim Botânico Tropical de Lisboa*, com três trajetos turísticos "*Botânico (Especialistas)*", "*Histórico*" e *"Aves"*, cada um com um tema diferente e direcionado a um certo perfil de utilizadores, permitindo navegação geográfica e interação com diferentes pontos de interesse de jardim.

O mapa, desenvolvido baseado na imagem do mapa do jardim, ao qual foram adicionados dados GPS de alta precisão, apresenta um percurso, com os seus pontos de interesse e um caminho recomendado. A posição do visitante e as orientações são também representados no mapa, sendo utilizados para isto os dados de GPS e dos sensores de telefone – giroscópio, acelerómetro e magnetómetro. Para cada posição marcada, é apresentada informação sobre um

objeto (no caso de ser um ponto de interesse) ou conjunto de objetos (no caso de ser uma área de observação) das coleções do jardim.

Esta aplicação permite deste modo interação dos visitadores com diferentes componentes do jardim (plantas, prédios, pássaros), como também o acesso a diversos recursos educativos incluídos na *app*, adaptados a cada perfil de utilizador. Estes conteúdos são utilizados para descrever os vários objetos contidos no jardim, e são descarregados do servidor aquando da primeira vez que o visitante explora o percurso e são armazenados localmente no dispositivo. A *app* também armazena dados de utilizador que podem ser utilizados para melhorar os serviços do jardim.

Por fim, são sugeridos como trabalhos futuros diversas funcionalidades, que podem ser incluídas para melhorar as propriedades analíticas e lúdicas da aplicação.

Keywords: Mobile applications, Botanical gardens, Tours.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The present work for master's degree in Informatics at the Faculty of Sciences, University of Lisbon (FCUL) was proposed by rectory of the University of Lisbon to be accomplished at the LASIGE – Large-Scale Informatics Systems Laboratory, of the Department of Informatics. The development of a mobile application for the garden – Jardim Botânico Tropical de Lisboa (Tropical Botanical Garden of Lisbon) - with a curated list of tours oriented to different visitor profiles, was proposed. This application aims to attract interest from Lisbon citizens as well as tourists, to visit the garden and learn more about it and its components.

In order to feed contents to the application and to collect data from the visitors in a centralized manner, data which can be after used to improve the services provided by the garden, a system had to be implemented based on the client server architecture.

1.1 Motivation

The University of Lisbon is involved in the management of several public gardens: Ajuda Botanical Garden, Tapada da Ajuda, Lisbon Botanical Garden and the Tropical Botanical Garden [5].

The Tropical Botanical Garden of Lisbon, classified as a National Monument in Portugal, is situated in a region of Lisbon with many museums and institutions of great interests for visitors (e.g., Mosteiro dos Jerônimos, Centro Cultural de Belém, MAAT, Museu do Oriente, Palácio Nacional de Belém). The Garden was created in 1906 in the context of the organization of the colonial agricultural services for teaching Tropical Agronomy as a complement to the Agronomy and Veterinary Institute. In 1907, it was installed in the greenhouse at "Quinta das Laranjeiras" where the Zoological Garden is currently located. The current location was established in 1914 – near the Jerónimos Monastery and Belém Palace. The Tropical Garden is home to a collection of approximately 600 species [6] belonging to more than 100 botanic families that generally grow in the tropical and sub-tropical region. Since 2015, the management of the garden is under the National Museum of Natural History and Science of the University of Lisbon.

The advances in information and communication technologies already contributed for the implementation of tools that increase people's interests and engagement with collections from museums and botanical gardens. Different software applications (apps) were developed that combine information on objects of collections with navigational tools and social networking. An app was studied in more details – Jobim Botanic – which is an app that was used in Rio de Janeiro's Botanical Garden, providing practical and recreational information about the garden; a navigational tool that allows users to explore the garden; and tools to record, produce and share data on garden [7].

The knowledge and skills acquired during my bachelor's degree in Information Technology with a minor in Design and Multimedia, and the courses I attended during my master's degree's first year,

contributed to my decision to research and develop the app for the Tropical Botanical Garden of Lisbon. It was an interdisciplinary work where knowledge from different areas such as Botany, Ornithology, History, Data Visualization and Software engineering were combined.

This work allows to identify challenges of using information and communication technologies for presenting information about objects and their meaning to people. Also, this work will enable me to increase my knowledge of designing and implementing mobile applications for the *Android* operating system. This work is also an opportunity to deepen my knowledge of object-oriented programming, and design patterns. Finally, as the work will enable me to learn more about botany, and the rich history of the garden.

1.2 Objectives

The project presented in this thesis aims to implement a mobile application for the Android operating system, for the Tropical Botanical Garden of Lisbon. Through a survey of the characteristics of related applications and collaboration with garden staff and other members involved in the project (designers, biologists, etc.), an application will be developed that will support the aforementioned features. Therefore, this dissertation has three main objectives **(O)**:

- **Objective 1 (O1)** - design and implementation of a mobile application for Tropical Botanical Garden of Lisbon that integrate navigation tools and a curated set of tours, oriented for different types of user profiles: students and tourists. Each tour must offer a map with a selection of points of interest and a recommended path. In some tours filtering of points of interest must be available, and the user must be aware of the already viewed point of interest. When the user clicks a point of interest details about the point of interest must be presented.
- **Objective 2 (O2) –** creation of an application server layer, to store contents and data that are used in the mobile application, and visitor data
- **Objective 3 (O3) -** evaluation of the implemented system.

1.3 Contributions

This project contributed with some software components to the Tropical Botanical Garden of Lisbon. The first contribution was the creation of a mobile application, with a curated set of tours, navigation tools, with the aim of attracting different profiles of visitors to the botanical garden, such as students, tourists and families. This application is supported currently only by smartphones running from version five up to the most recent version of *Android*.

The second contribution was the creation of a central repository with information about plants, buildings and birds contained within the garden, and also data on the visitor demographics and trajectories, aiming to improve the garden's services. This repository is contained within a virtual machine running on one of the universities servers. Web services were also implemented to enable the communication between the mobile application and the central repository of contents and data.

The final contribution was the evaluation of the system. To evaluate the system, tests with experts were made in order to evaluate the usability and acceptance of the system and also a tool was implemented to test the performance of the server. From the tests it was concluded that the system was well accepted by the users. With the performance tests it was also possible to relate the duration per request with the number of parallel requests and identify possible issues that may occur when the server needs to respond to a great number of parallel requests.

1.4 Document structure

This document is organized as follows:

- **Introduction –** work context, motivation, objectives and contributions.
- **State of the Art** – presentation of results of literature research on published articles related to development of software applications for cultural institutions (public garden and museums) and analysis of the features and functionalities of the applications from Google Play and Apple App Store for museums, public gardens and city guides.
- **Analysis and Design –** description of the features and requirements of the application, resources, tools and methodologies used, system architecture, data model and prototypes.
- **Implementation –** details on the implementation of software functionalities such as the interactive map, content download, data synchronization, and web services.
- **System evaluation –** exposing the fulfilled activities with the aim of evaluating the system, such as heuristic, user and system performance tests.
- **Conclusions** – summarization of the accomplished work, challenges and acquired skills, as well as various predictions and suggestions on the future of the mobile application.

Chapter 2

Related Work

In order to establish requirements for the app and also in order to design the application, both visually and functionally, several applications have been studied. Some of these applications have the same orientation as the app that is being developed, i.e, to Botanical Gardens. Other applications also presented in this section, although not oriented to Botanical Gardens, are oriented to environments that share similar challenges to the ones oriented to Botanical Gardens, such as museum guides and city guides. We close this section with a comparison of the features of all the presented applications. Additionally, it is also presented a comparison of the technologies used in the development of the studied applications (for instance location tracking technology).

2.1 Mobile applications for botanical gardens

In many cities, public gardens are spaces that include different recreational and educational resources. Parks and community gardens that contain ornamental and edible plants meet the definition of a public garden if they employ a professional staff, have a database on their plants, and adopt a mission statement that drives their efforts [1]. In addition to their main role to maintain collections of plants for purpose of education, research and conservation, currently public gardens collaborate with community in waste reduction, food provision, water conservation, and community development [2]. A number of public gardens assume different roles in community development: educator of citizens and associated organizations; trainer of community members, city workers, and volunteers in specific skills; and technical expert supporting local governments and community development initiative. Landscape beautification is an important contributor to the revitalization of existing and older neighborhoods, and to improvement of property economic value [3]. Several public gardens are perceived as a driving force for change in communities that may increase region vitality and support sustainable development.

The development of mobile digital technologies like smartphones and tablets incentivized the creation of new ways of communicating between institutions and communities. Improvements in the capacity of smartphones for computing, sensing and data storage as well as the advances in communication technologies (e.g., 4G, 5G) and social networking turn these devices important tools for institutions to improve their services. The most attractive feature of apps for smartphone is the capacity to reach a large number of individuals through a personal device they have chosen and are familiar with, wherever the user chooses to be. These apps may increase the possibilities not only for one-to-one communication between the public gardens and the user, but also for social networking.

With advancements in the field of Augmented Reality, the set of functionalities of mobile apps for cultural institutions has increased. One of definition, considered being the first definition, define Augmented Reality as a medium “combining the real and virtual, interactive in real time, registered in

3D" [8]. Augmented reality can also be defined as "a live direct or indirect view of a physical, real-world environment whose elements are *augmented* (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data" [9]. Algorithms developed in the field of computer vision enabled the distinction of surfaces, allowing a more precise positioning of the virtual objects in the real world.

Applications oriented to botanical gardens started to be published in 2011. Examples of applications that were developed in this year are the Kew Gardens, and the Chicago Botanical Garden [7]. These systems are based on GPS signal, and in most cases show the position of the visitor and provide automatic routing, enabling visitors to explore the garden in a more flexible way.

Table 1. Analyzed applications for botanic gardens.

Name	**Institution**	**Android**	**iOS**
Jardim Botânico do Recife	Recife Botanic Garden, Recife Brazil	✓	
Jardim Botânico do RJ	Rio de Janeiro Botanic Garden, Rio de Janeiro, Brazil	✓	✓
Madeira Botanical Garden	Madeira Botanic Garden, Madeira, Portugal	✓	
GardenGuide	Chicago Botanic Garden, Chicago, USA	✓	✓
Royal Botanic Garden Sydney	Royal Botanic Garden Sydney, Sydney, Australia	✓	✓
Tohono Chul Park	Tohono Chul Park, Tucson, USA	✓	✓
Norfolk Botanical Garden	Norfolk Botanic Garden, Norfolk, USA	✓	✓
Botanical Garden of Padova	Padova Botanic Garden, Padova, Italy	✓	✓
Memphis Botanic Garden	Memphis Botanic Garden, Memphis, USA	✓	✓
Adelaide Botanic Garden	Botanic Gardens of South Australia, Adelaide, Australia	✓	✓
"Plants with Bite!"	Royal Botanic Garden Sydney, Sydney, Australia	✓	✓
Santa Fe Botanical Garden	Santa Fe Botanical Garden, Santa Fe, USA	✓	✓
RJB Live Museum	Real Jardín Botánico, Madrid Spain	✓	✓
Botanischer Garten Graz	Graz Botanic Garden, Graz, Austria	✓	
Botanic Nearby	Royal Botanic Garden Edinburgh, Edinburgh, UK	✓	✓
Botanischer Garten Frankfurt	Frankfurt Botanic Garden, Frankfurt, Germany	✓	✓
BotanyAR	Queen Sirikit Botanic Garden, Chiang Mae, Thailand	✓	
Hortus Haren / Laarmantuin	Haren Botanical Garden, Haren, Netherland	✓	✓
Leśne Arboretum Warmii i Mazur	Lesne Arboretum Warmii i Mazur, Olsztyn, Poland	✓	
Ogród Botaniczny UJ	Botanical Garden of the University of Wrocław, Wrocław, Poland	✓	
Ogród Botaniczny UW	Botanical Garden of the University of Wrocław, Wrocław, Poland		✓
Olds College Botanic Gardens	Olds College Botanic Gardens	✓	✓
Rancho Santa Ana Botanic Garden	Rancho Santa Ana Botanic Garden, Claremont, USA	✓	✓
Déclic botanique	Klorane Botanic Garden Foundation, France	✓	✓
SZTE Füvészkert Szeged	Botanical Garden of the University of Szeged, Szeged, Hungary	✓	
Botanischer Garten Wien	Vienna Botanical Garden, Vienna, Austria	✓	✓
CloudGuide	CloudGuide	✓	✓
Cannes Jardin	Ville de Cannes, Cannes, France	✓	✓
Jobim Botanic	Rio de Janeiro Botanic Garden, Rio de Janeiro, Brazil		✓
Kew	Kew Royal Botanic Gardens, Richmond, UK		✓

By searching on Google Play and iTunes app (short for application) store, from last year, a number of mobile applications for public garden were identified. In Table 1 for each of the identified applications oriented to botanical gardens, the institution is presented as well as whether they are available for Android or iOS. Online information related to different tools and features of these applications were analyzed.

Particular attention was given to the applications *Jardim Botânico do RJ* app, created by Jardim Botânico do Rio de Janeiro [10], *Royal Botanic Garden Sydney* app developed by Sydney Royal Botanical Garden [11], *RJB Museo Vivo* app developed by Real Jardín Botánico (CSIC) [12], *Kew* by Kew Gardens [13], and Jardim Botânico da Madeira by urbanXcode Team Developer [14]. The

Jardim Botânico do RJ application was further detailed as it can be considered the most successful implementation of these types of applications, based on its score on the Google Play Store. The *Royal Botanic Garden Sydney*, was chosen for its implementation of the "capture" use mode, which enables users to collect notes and pictures of the garden and share them on social media; this feature was considered a good extension to the sets of functionalities provided by the application. The *RJB Museo Vivo*, is presented for its alternative tracking technique used. Next the Kew app was detailed for being one of the first implementations of a mobile application for botanical gardens. Finally, as a national example the Jardim Botânico da Madeira was detailed for being the only existent application available in the Google Play application market, made for a Portuguese botanical garden.

The *RJ Botanic Garden* application, initially launched in 2013, exclusively for the iOS system, with the name *Jobim Botânic*, and now supported also for the Android system, was one of the first botanic garden applications to be released. Initially, this app provided garden visitors with an extensive set of features as a curated set of garden tours, tools for saving and sharing notes and photos taken while using the app, and tools based on Augment Reality. Initially, the functionalities of this app could be sorted into three modes of use:

- **Info** – where the user gets practical information about the garden, such as directions, schedule, parking, activities taking place in the garden, app documentation and configuration. In this mode multimedia contents that include poems and photographs that may inspire people to visit the garden are also included.
- **Visit –** the main use mode of the application. The visitor accesses a set of tours offered by the garden, choosing the one that most interests him, later being assisted in it by interactive navigation tools. Resources to take photos and make notes with geolocation are also included in this mode.
- **Data** – allows the visitor to collect data during their visit, whether these are notes, photos or contacts. After collection, it is also allowed to share this data, which can be done at the same time as visiting the garden, or after visiting the garden.

In Figure 2.1 these modes are can be seen and also some screens from the info mode.

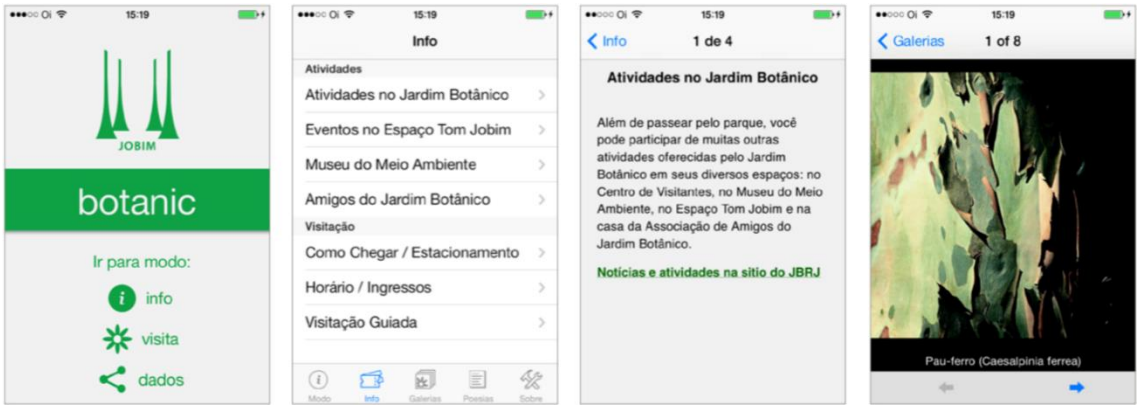

Figure 2.1. Jobim Botanic app [7]. From left to right: the home screen where the use mode is chosen; info mode where the user can view activities taking place in the garden, and also images of the garden.

The new version no longer has the data mode. The *Info* mode in the new version was changed in two new modes, the *Information* mode and the *Programming* mode. The *Information* mode partially retains the contents from *Info* mode, but the activities taking place in the garden are now on schedule. The *Visit* mode in new version changed in two modes, the *Tours* mode and *Garden Map* mode. The *Tours* mode comprises a curated list of tours. The *Garden Map* mode allows users to view and filter points of interest on the map and navigate to that point of interest without having to choose a route.

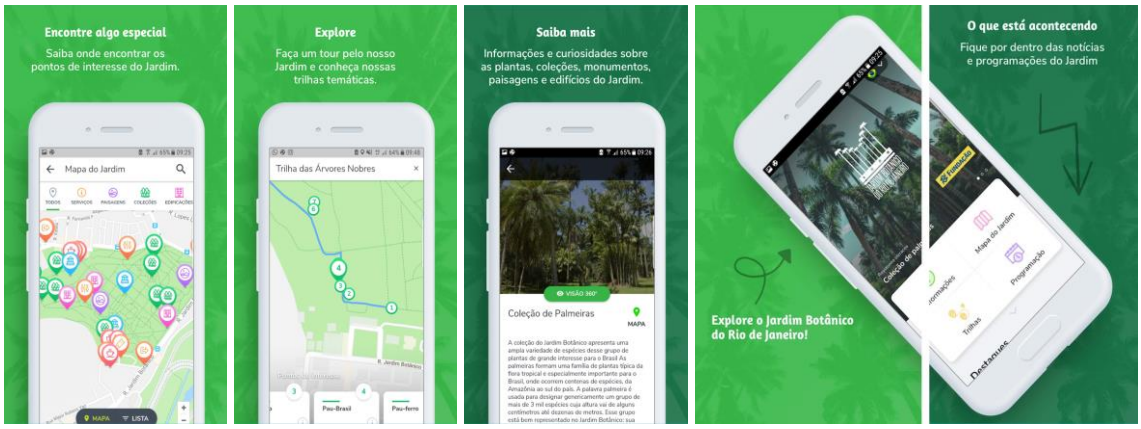

Figure 2.2. Jardim Botânico do RJ [10] - new version of the Jobim Botanic app, Rio de Janeiro Brasil. From left to right: *Garden Map* mode; *Tours* mode; presentation of a point of interest; Home screen, where the use mode is chosen.

The *Visit* mode of the first version of the application can be decomposed into four components: the search component - allowing visitors to find specific points on the map; the tour component - allowing the visitor to choose a route to follow through the garden; the map - which allows the visitor to locate itself, the route and points of interest. Also Augmented Reality was included to offer a more immersive navigation, the images of user's surroundings being enhanced with geolocated directions and landmarks. However, the Augmented Reality functionality has been replaced by 360-degree views, as it was easier to correctly give direction and mark points in the view than to change the camera image in real time. The 360-degree view in the new version is a feature that could be used also after or before the garden visit.

In the original version, for each point of interest included in a tour poems and music from *António Jobim* would be presented to visitors (a very important musician in Brazil, who appreciated the botanical garden and wrote many poems and songs about it), having this feature removed in the development of the latest version of the application.

Fast advances in mobile technologies imposed not only changes in functionality, but also in the technologies used to develop those features. Initially the map service of choice was Apple Maps. As map services at the time did not have a detailed representation of the botanical garden (Figure 2.3), with its paths and streets, this choice was made solely because Apple maps offered more tools than the other navigation services. However, the map had to be modified, based on official maps of the botanical garden. A layered geographical database of nodes was created, making it easier to establish paths (Figure 2.3). In the new application, however, Google Maps is used, as the map was corrected with the paths that were not previously visible.

(a) (b)

Figure 2.3. Official Map and Apple Maps comparison. (a) official map and (b) Apple map [7].

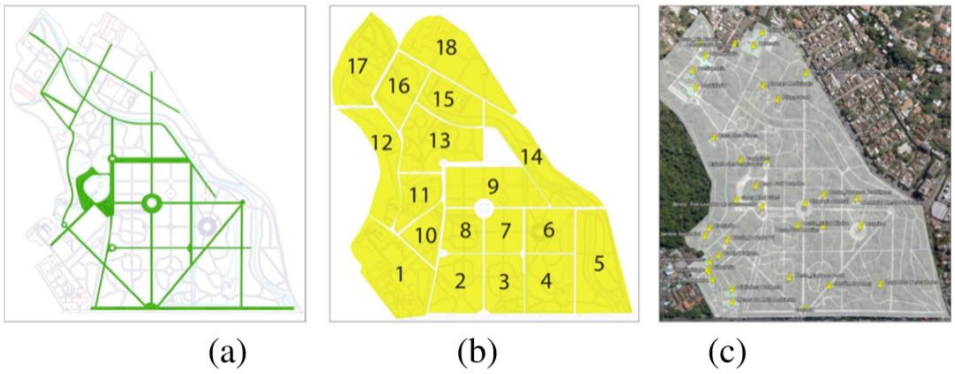

Figure 2.4. Visual representation of the layered database. (a) lines that make up the garden's paths; (b) plains which divide the map; (c) polygon that makes up the area of the garden [7].

The Rio de Janeiro Botanical Garden app can be considered the best implementation of a mobile application for Botanic Gardens of our time, having a five-star rating on the Google Play store. The great success of this application came from the maturation of the features, and the constant search by developers to keep the application as simple as possible to use and trying to provide users both with practical garden information as well as navigation information as faithful as possible.

Another application that stands out, as previously mentioned, is the *Royal Botanical Garden Sydney*. This application offers, as well as the Rio de Janeiro Botanical Garden application, practical information about the garden (opening hours, prices, directions to the garden, events, etc.), various themed tours, offering interactive navigation tools to assist visitors on their route (using Google's location service and also the official garden map), point-of-interest search and point of interest filtering on the map.

Figure 2.5. Royal Botanical Garden of Sydney app, Sydney, Australia [11]. From left to right: home; where the options map, "What's on?" and Tour are presented; *Tour* mode, which is a curated list of tours; map with tour marked; search functionality for points of interest in the map.

The application allows users to record audio, video, take photos or even write about a point of interest (similar to the *Data* mode of the initial versions of the Rio de Janeiro Botanical Garden application). The data is saved in the user's diary. All elements saved in the journal can be shared. Important feature of this APP is the tool that allows planning the visitors route based on their favorite points of interest. In addition to the previously mentioned features the *Royal Botanical Garden Sydney* application offers the *Plants with Bite* [15] feature. This feature consists of the use of Augmented Reality to complete the *Plants with Bite* path (a path centred on the exploration of garden carnivorous plants).

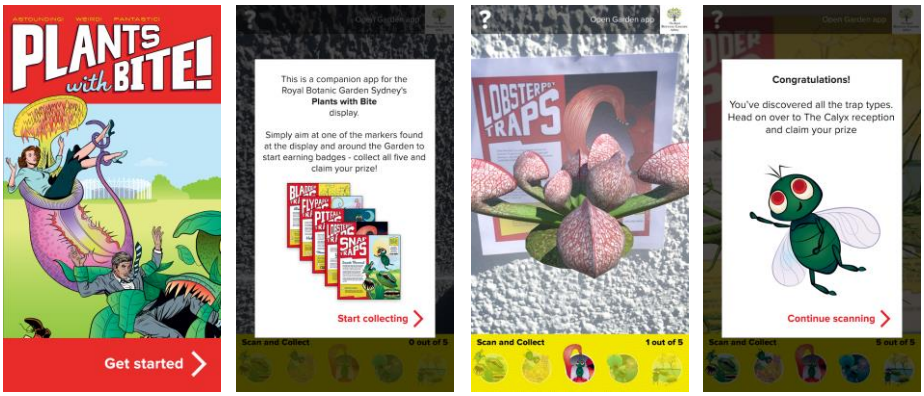

Figure 2.6. "Plants with Bite!" app [15] by Royal Botanical Garden of Sydney, Sydney, Australia.

In this tour the user must collect a set of five plants using Augmented Reality markers and finally, if children are included in the group, a prize is offered. This feature aims to attract families to the garden visit, offering a playful component to the tour.

RJB Living Museum, the application for the Real Jardín Botánico de Madrid, is similar to the other applications mentioned above, that is, it includes practical information about the garden and guided tours. However, standing out is the system used in navigation, which is the Aruba Meridian system, a location and map system, commonly used indoors and for tracking goods, but adapted to the garden context. This system provides its users with extremely accurate turn-by-turn navigation using Aruba Meridian beacons to correct GPS location data. This system is proprietary, as such we did not use this system in our application.

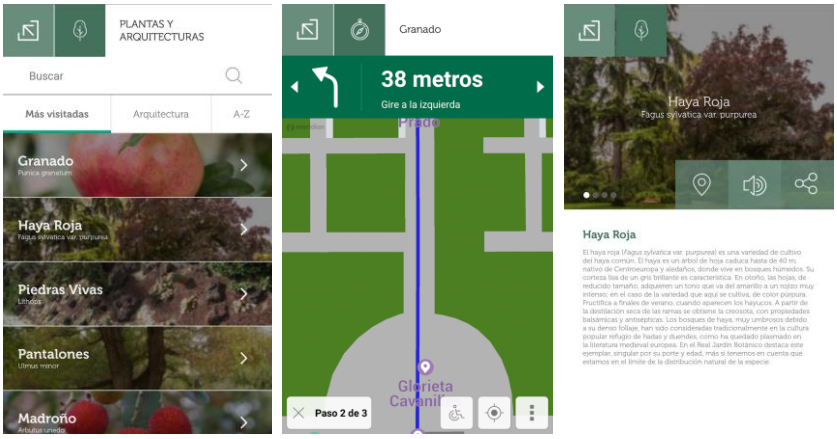

Figure 2.7. *RJB Museu Vivo* app [12] by the Real Jardín Botanico of Madrid, Spain.

The *Kew* application offered by Kew Gardens had also interesting features when it was first launched in 2011. An overview of these features is included in the study of Mann [13]. This application is no longer available and has been replaced by a *CloudGuides* application, guide. In the above presented study were indicated the reasons visitors have for visiting the garden. Paraphrasing an audience survey study conducted by Natasha Waterson, Senior Producer of Mobile Services at Kew, the most visitors visit the garden for social, emotional and spiritual reasons, as opposed to intellectual ones. So, most visitors are looking for a visiting experience consisting of wandering around the garden, and spontaneously discovering it. Based on the audience study, the application was then developed following the *delightfully lost* metaphor (i.e., not establishing a fixed route structure, only triggers and recommendations were used to guide the visitor through garden). The study conducted by Mann et al. was then conducted on families who regularly visited the garden, and visitors who visited the garden once or twice for social reasons. Each of the visitors or groups of visitors was asked about different features that the application offers: *See today at Kew*, the map and GPS, augmented reality

and QR reader. This application received positive reactions from visitors. However, in certain cases the experience has been degraded by the failure of the GPS component, making it difficult to navigate through the garden, and fully appreciate the functionalities offered by the app.

The *See today at Kew* feature is a feature mainly relevant in the pre-visit garden context, consisting of the presentation of highlights trough pictures on Flickr, and notifications which recommended places to visit, allowing the user to know the exact location of the point in the garden. This functionality was well received however due to the fallible GPS signal, users sometimes had a hard time finding the location of the points of interest.

The representation of points on the map in *Kew* APP, was indistinguishable, each of them being a pin marked with an icon of an eye. This approach of not highlighting points of interest was used to maintain consistency with the metaphor chosen for the application design. The map was also based on Apple Maps, which was not very well received by most users as they used Google Maps. Google maps offers a different interface from Apple maps. Google offers a more crowded interface, with lots of options visible from the start, which differs from the Apple maps design philosophy, which contained all the options hidden in button with “customize” written on it. This difference misled users used to Google maps, to think they couldn’t customize the map.

Figure 2.8. Kew Gardens app [13], map with "See today at kew" points of interest marked.

The Augmented Reality feature in the application uses the smartphone camera, GPS location and compass to provide a turn-by-turn navigation experience, with a blue ball pointing the position of the plant species. This feature was well received by young people. However, the elderly, having not understood the feature, did not have a good experience. This application has also suffered from GPS coverage issues, making it difficult to get a good indication of the way forward. However, the enthusiasm of the users did not impact reduced the impact of this factor, as opposed to the location using solely the map.

The QR reader, a feature implemented experimentally, in certain areas of the garden, was not very successful either. Although QR were growing rapidly in popularity, many users did not know what to do with these. Even for those that correctly used the QR reader, the experience provided (audio presentation on the point of interest) did not arouse much interest. The study also points out that the use of Wi-Fi and 3G to update content in certain cases was not very well received, as people were not expecting to have to wait as much for content to be presented to them.

Finally, the use of the Flickr social network to present content was very well accepted in general as the information used to describe the plants was not meaningful, as it did not tell what specie it was and what characterizes it.

The Mann et al. [13] study not only presents features that the Kew application offered, but also problems that the implementation of these features may rise. Most of these problems were related to the users not being familiarized with the technologies and concepts used to implement the functionalities.

In Portugal it was developed a mobile application for botanical gardens in Madeira - *Madeira Botanical Garden* [14]. Figure 2.9 presents some screens from this application. In *a* we can view the start screen for the application, containing the menu. The users can from this screen: view the Garden's History by clicking "História"; view the schedule for the busses that lead to the garden in "Transportes"; view a list of plants ordered by their content of origin by clicking "Plantas"; open the app's QR code reader, to obtain information about plants, by clicking "QRCode"; view the map of the garden an filter the information displayed on it by clicking "Mapa"; and finally, to obtain other information such as opening hours, pricing, accessibility and conveniences (ex. restaurants and souvenir shops), by clicing "Informações". The interface for the map (based on Google Maps [16]), can viewed in *b* with all the information such as sights for bird watching and parking. The interface contains two selection lists, for selecting the types and subtypes of the points of interest (see *d*). When the user clicks on one of the points of interest the user is shown information about it, as it can be viewed in *d*.

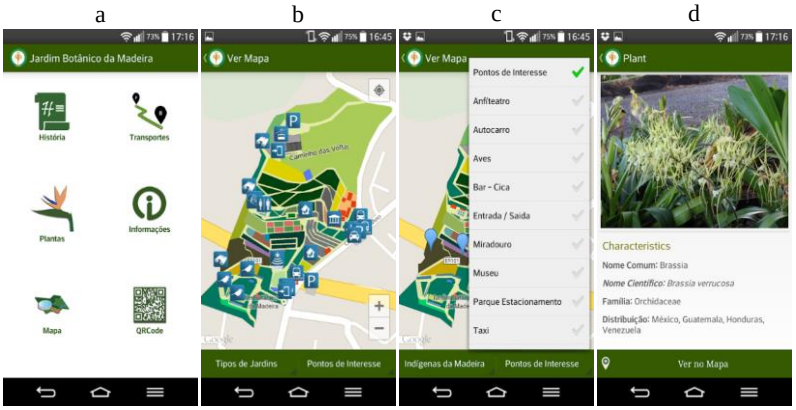

Figure 2.9. Jardim Botânico da Madeira app [14]: (a) initial screen; (b) garden map; (c) points of interest filter; (d) point of interest description.

This application is an example of an application for botanical gardens that is classifiable under the category of applications that handle navigation at the macro level, because it offers mostly practical information for navigation not providing very detailed information about points of interest. It offers a practical garden information component (transport to the garden, opening hours, prices, accessibility for the disabled), a list of plants that are in the garden, providing very brief information for each specie (i.e., common name, scientific name, family and distribution, in addition to being able to see the position of the plant on the map). The app also offers a map, although only to view the visitor's location and points of interest. Nevertheless, the app has a good rating in the Play Store (4.3 stars out of five stars).

2.2 Mobile tourist guides

Audio Guides have been the most popular method to engage and enhance the visitor experience in cultural institution. Digital guides for different cultural institutions (e.g., museums, public gardens) began to be used from 1990s. Mobile applications for cultural institutions however started to be created only starting from 2009. This was due to the advances in smartphone technologies, more precisely the appearance of the iPhone. The dominance of the iPhone as the target device of these applications may be attributed to the great popularity of the Apple's application store, in association with early stage of mobile development technologies, when cross platform development tools did not yet exist. The early stage of technologies and software packages available for mobile development, and the reduced processing power of devices at that time, did not enable the development of very

complex applications. Thus, most applications from 2009, had very simple functionalities such as object presentation through image, text and audio, and were often developed for closed spaces.

According to the creators of the Jobim Botanic APP [7] applications for general spaces can be classified into two types (or categories): **content-privileging applications** that deal with small, enclosed spaces, these applications typically being directed to museums or galleries; and **applications that deal with navigation on a macro level**, typically being city or country guides. Applications for botanical gardens can, however, in certain cases be distinguished into a third category, as they have **characteristics of both categories** mentioned above and cannot be classified as belonging to only one of the categories.

In the first category are included applications normally directed towards museums. As museums are generally enclosed spaces, and the technologies for location being expensive (GPS can't be used so other approaches using additional devices such as Bluetooth beacons must be used) and in many cases unjustifiable. This is the reason that many museums APPs are information-centred – being mainly a mean for transmitting information – interactive navigational tools not being included.

The second category covers applications such as city tour guides (e.g., *Visit London* mobile application), which focus mainly on a visitor's routing, offering interactive navigation tools, without providing very detailed and complex information, on the points included in the routes.

Garden-oriented applications (including botanical gardens) may in some cases be considered as belonging to a third category, as they attempt to provide the visitor with detailed information about the garden and its elements on display, while providing tools for interactive navigation. In general, garden applications can also be classified as belonging to only one of the two categories already mentioned.

Following different examples of APPs and their relevant features related with the first two categories are presented.

2.2.1 Applications that privilege content

These applications were developed mainly for the transmission of varied information about the place to which they are oriented, and about the objects on display, often transmitting the information using various types of media, and in some cases even games.

There is currently a large number of applications that privilege the contents. In one study was analysed a set of 71 museum-driven [17] mobile applications published between 2009 and 2010. Several findings from that study allow to better understand the objectives and features of these APPs. In their study it was found that the vast majority of applications (27 of 71 applications) were developed by museums in the United States of America, and most were developed for the iOS system (63 of 71 applications). A few were available both for iOS and other operating systems (four of those for Android operating system). The authors of the study suggested that the limited funds of museums could lead to the development of applications for only one platform (i.e., mainly iOS). The iOS although did not have the largest margin in the mobile device market at that time, was the most popular and profitable mobile app store. Most of the applications presented in the study, today are available for both iOS and Android. The Android application market has become progressively more profitable, however. By 2021, Android application development is expected to become much more profitable than iOS development, according to App Annie's latest report [18].

Applications were also classified in relation to museum content into six categories (see Table 2).

Table 2. Applications studied by Economou et al. [17] grouped by their types of contents.

Type of application	Number
Presentations – guided tours of permanent exhibitions and the museum in general	29
Presentations – guided tours of temporary exhibitions and practical information about the museum visit	20
Combination of the two above	5
Apps devoted to a single object or artwork from the collection	5
Content creation or manipulation from the user, inspired by artists' work	3
Games based on the exhibits	2
	64

Most of the analysed applications fall into the first three categories. These applications take the form of a guided tour, exploring the museum's exhibits and points of interest using buttons, expandable images, text and, in many cases, audio. An example of an application that stands out in this group is the *KHM Stories* application from the Museum of History and Art in Vienna (KHM-Museumsverband, KHM Stories – no Google Play). This application offers the visitor a series of tours that follow a certain theme, directed to a certain age group. Thus, there are tours made for all age groups. In all tours, the museum is navigated using images in which a character, depending on his position, indicates the direction to follow (e.g. if a character has his back to visitor, it means that he should moving forward). Sometimes arrows are used to point out the direction.

Upon arrival at one of the points of interest on the tour/route, the user is warned that he is in front of a point of interest, and on the next click, a story about the points of interest is told using videos, images, sound, among others.

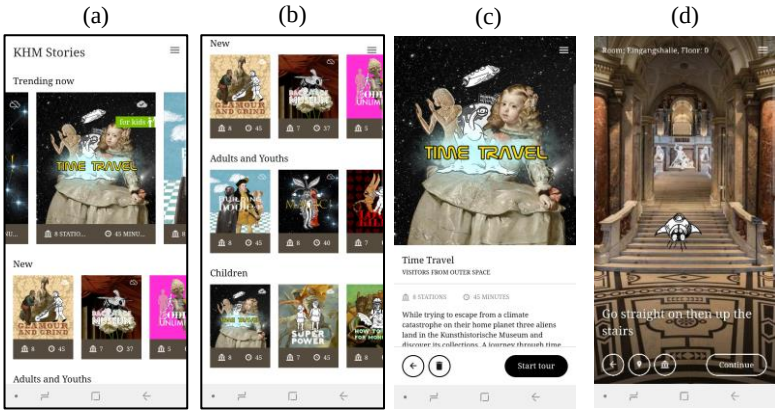

Figure 2.10. The KHM Stories app [19]: (a) and (b) Initial screens having list of tours; (c) example of a screen that initiate the tour; (d) image driven navigation.

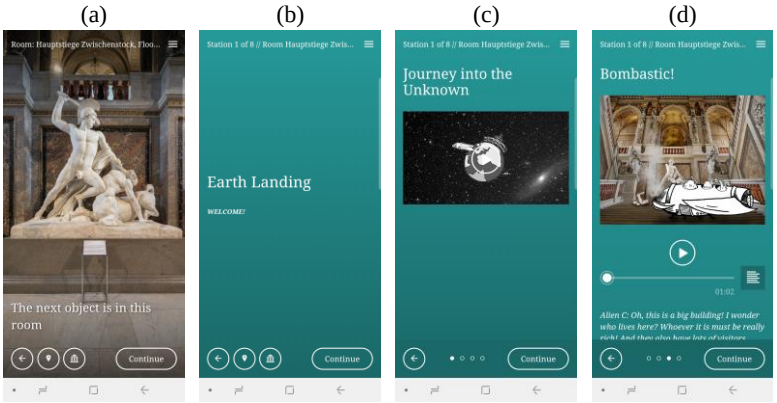

Figure 2.11. KHM Stories app [19]: the screen when a visitor is in front of point of interest; (b) stating the story; (c) video related to point of interest; (d) text and audio related to points of interest.

Other example of an application belonging first category that is the *Rijksmuseum* application of Rijksmuseum in the Netherlands (in Google Play store). This application stands out for containing a mode for when the visitor is in museum, and another mode for when the user is outside the museum. When outside the museum, the user can select images which he can enlarge to see more details and a textual description of the selected image. A widget (developed for the iOS system) that Rijksmuseum offered was presented in the study of Economou M and Meintani E (2011). It presented a daily piece on permanent display, giving both detailed information about the piece and the author and also an image of high resolution that the user can zoom in to see more details in order to increase users' interest in revisiting the museum.

Figure 2.12. Rijskwidget widget [17]. Example of a screen from Rijkswidget APP showing one piece from the collection daily.

Unfortunately, this widget is no longer available, and the new application does not implement this approach in the form of daily notifications. The new application allows the detection of arrival at a certain point of interest in the route via Bluetooth (using the Google Beacon system), to obtain information about the piece using audio or video. If visitors choose the video option, the audio narration is accompanied by a video that zooms in on the image at a certain point so the APP user can see the detail mentioned in the narration. The application also allows to save as much detail as the entire image of a piece, allowing the users to view later the piece as well as to create their collections.

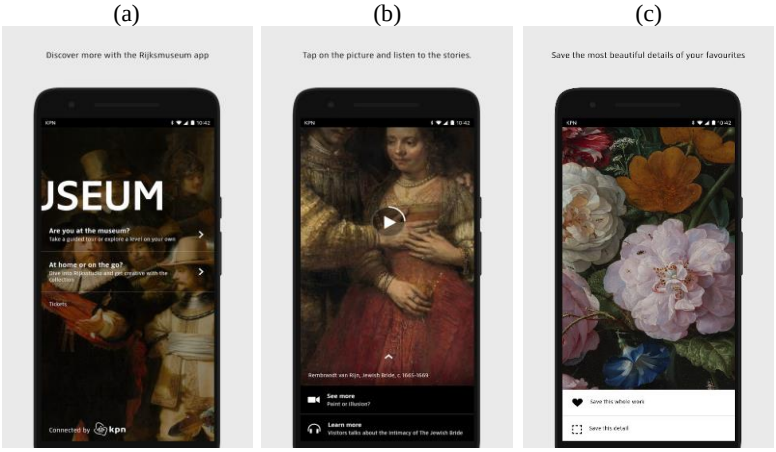

Figure 2.13. Rijksmuseum app [20] with the two mode of interaction - audio (b) and video (c) mode.

Another example of these types of applications, which offers a more interesting take, on the presentation of information about the garden is the *Magic Tate Ball* application [21] developed by Tate Gallery. The initial screen presents a “magic ball”. As the user shakes the phone factors such as date, time, GPS location, and local weather conditions are weighted, and an art piece is selected from the Tate Museum archive. The user can select what factors are used in this selection.

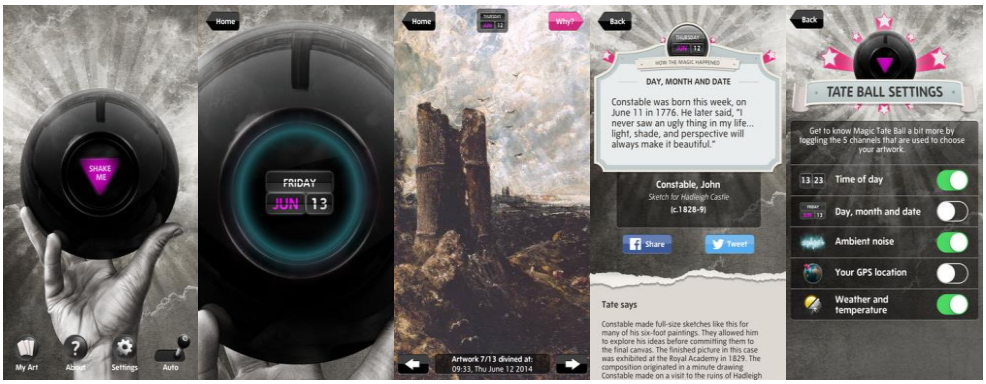

Figure 2.14. Magic Tate Ball app [21]. From left to right: Home screen; shake screen; artwork screen; description of the work and author screen; settings screen where the environment aspects are chosen in order to pick the artwork.

Following are applications directed to a single object or work in a collection. These apps present an object belonging to a temporary or permanent exhibition, a recent acquisition, or to promote an electronic publication (e.g., e-book) about a particular work on display, serving more as a marketing technique to promote a particular object. Only 4 applications under this category were identified in the study of Economou and Meintani [17]. However, the one that stands out most is the *Miroslaw Balka: How It Is* application [17] developed by the Tate Modern museum, which aims to promote the work of Miroslaw Balka. This application offered the visitor information on the work of artist using Augmented Reality. The art works exhibited at Tate Modern's Turbine Hall between 2009 and 2010, were integrated in a kind of mystery game. The artist's notes were spread around the space and videos describing the work and its inspiration were presented. Unfortunately, this application is no longer available.

Figure 2.15. Miroslaw Balka app [22]: How It Is application.

Next are applications aimed at user's creation and manipulation of content, inspired by the work of certain artists. Examples of such applications are the *Muybridgizer* app [17], also developed by the Tate Britain gallery, to complement a temporary exhibition on chrono photography by photographer Muybridge. This application allows the user to take photos that are transformed based on the artist style, i.e., combining different freeze frames with a sepia filter applied to them in a single picture, which the user can share on the Flickr social network.

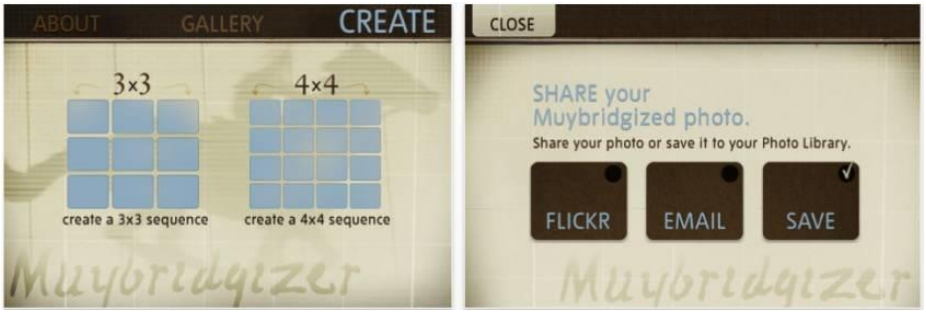

Figure 2.16. *Muybridgizer* app [17], by Tate Modern, London, England.

Other example is the *In Still Life* application from the LACMA museum in Los Angeles. This application, designed by contemporary artist John Baldessari [17], allows users to recompose a 17th century Dutch piece of art by rearranging 38 objects from the artwork.

Figure 2.17. In Still Life app by John Baldessari (2001-2010) [17]. Both images on the right, have the objects placed on the bottom bar, and user can drag them and place them in the way he sees feet creating a new artwork.

These types of apps are usually available exclusively for the iOS operating system, and only a few of these apps are currently available. Tate Gallery, however, continues to produce new experiences of this kind, such as the *Margritte Your World* application [23], which transforms any photograph by superimposing an animation into it, inspired by the most popular work of Magritte, the "Golconda". The surrealist video can be stored in the user smartphone and shared on social networks.

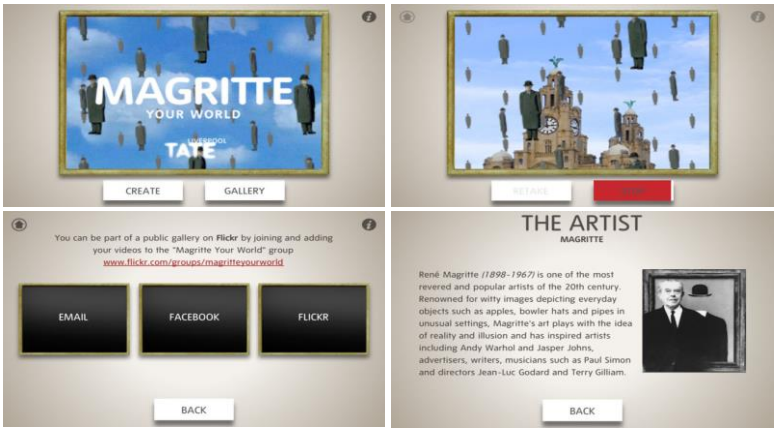

Figure 2.18. Magritte Your World app [23].

These applications allow the user to get actively involved in the exhibition, encouraging them to create their own works, thus promoting personal expression. Another advantage of using this type of applications is that the images produced are recorded on users' devices, allowing them to be shared on social networks, and thereby promoting the museum and its works and collections.

Serious Games, Gamification are other types of interaction with contents from apps promoted by different cultural institutions. Serious games, are defined as "games that are designed for a primary purpose other than pure entertainment" [24], or as "interactive computer applications, with or without a significant hardware component, that have a challenging goal, are fun to play and engaging, incorporate some concepts of scoring and impart to the user a skill, knowledge, or attitude that can be used in the real world" [25], "games that are designed to entertain players as they educate, train or change behaviour" [26] or "games with the purpose of improving an individual's knowledge, skills, or attitude in the real world" [27]. Gamification is defined as the "application of typical elements of game

playing (e.g., point scoring, competition with others, rules of play) to other areas of activity, typically as an online marketing technique to encourage engagement with a product or service.

For example, Tate Gallery developed *Trumps* application [28] which consists of a game for a maximum of three players, who at first must "capture" artworks by entering their number, adding them as cards to the player's deck. Finally, the players must battle each other by playing their cards. The battle is wone by the card with the highest score.

Figure 2.19. Tate Trumps app [28] by Tate Modern, London, UK.

In the addition to *Battle* mode, the application also contains the *Mood* and *Collector* modes. In *Mood* mode, the user should try to capture works that seem most threatening, exciting or absurd. The *Collector* mode consists of just collecting works, forming a personal collection with contents provided by the museum.

Nowadays, a new category of applications gain interest which can be called **universal applications**, as one application serves various institutions (museums or botanical gardens). An example of such applications is the *CloudGuide* application [29]. This application gives access to official information from all museums, art galleries and even gardens in the world. This type of application has the advantage of providing various museums with tools and templates to disseminate information about the establishment, including even directions for the user to follow. Location of many cultural institutions in one online place has brought benefits both for the users (they can be easily be informed about many cultural institutions near them or in the world) and the institutions by increasing the probability that their information could be seen online by a larger number of persons. The major disadvantage is, however, that establishments are very dependent on the features and tools offered by the company that produces this type of applications, diminishing the capacity to add improvements in the way that institutions communicate with people.

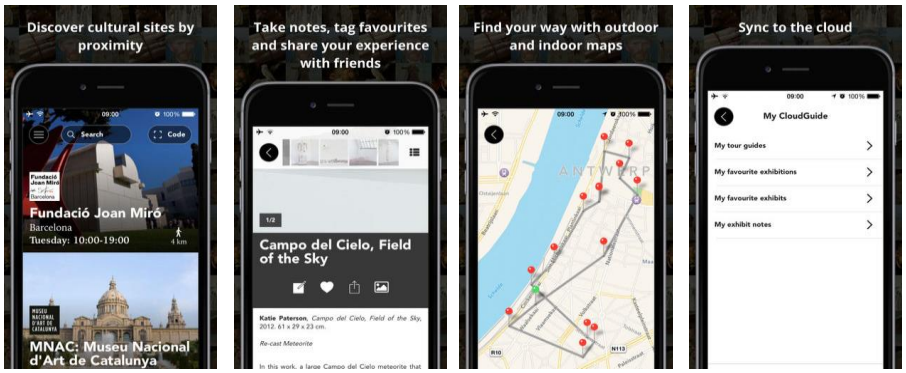

Figure 2.20. CloudGuide app [29].

Another other example of universal applications is *KeyARt - Museum Guide in Augmented Reality* by ARM23 LLC [17]. This application provides practical information of 25 museums, all over the world, such as opening hours and location. Information about the work in these museums is also

provided, using image recognition and Augmented Reality, which projects multimedia content over the image when the work is recognized by the camera.

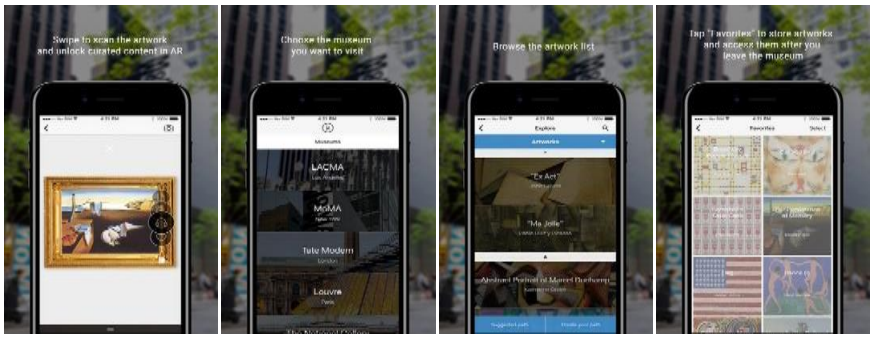

Figure 2.21. KeyARt app [30].

Most applications offer two use modes: a **pre-visit mode** and an **in-visit mode** [17]. In the pre-visit mode information such as directions to get to the museum is provided. In in this mode no navigation technologies are used such as interactive maps and automatic [17]. In the in-visit mode, details about objects of exhibits are presented. Additionally, some apps also have an interaction mode **after the visit**, allowing the user to view information about points of interest that he bookmarked during the visit.

Another observation in the study of Economou and Meintani was that few applications (19 out of 64 of the most closely analysed applications) offered the user the **sharing functionality on social networks** [17]. The vast majority of the applications that did offer this functionality (16 out of 19 applications) allow the user to **share, comment, rate and bookmark content**, and in most of these (13 of 16 apps) museum content or user-created content is shared using external social networks such as Facebook and Flickr. In this study only two applications offered the user the ability to connect to the museum's social networks.

The vast majority of applications studied by Economou and Meintani [17] have suffered from not adapting to the new paradigms, namely social networking and sharing, and creation of interactive contents with meaningful interactions (i.e., not only allowing for image enlargement, but also composition, modification and storage). Another problem of the applications studied at the time was also the low investment in navigation tools that would allow the visitor to be taken from where he was, following a route, or even arriving from one point to another. Although this study was conducted 7 years ago, the problems continue to exist in today's applications.

Although museum applications do not have the same navigation and location issues as garden applications, some navigation techniques that have been implemented for these spaces, as well as how visits were structured are worth to comprehensively being analysed. It is also interesting to adapt the games developed for exhibitions, in the exploration of the botanical gardens. The problems that most museum applications suffer today, namely the scarce interactive contents should be overcome by taking full advantage of the capabilities offered by advances in mobile technologies.

2.2.2 Applications that deal with navigation on a macro level

Application such as these with large geographical areas such as cities or even countries, typically serving as tourist guides, pointing out routes and places to visit on the map. For each point of interest, however, only practical information is presented (opening and closing times, ticket prices, contacts and a brief description, without going into too much detail). It is also very common that these applications also serve to buy tickets for points of interest. However, the main objective of these applications is to take users from point A to point B, often using pre-existing map services for that

purpose (e.g., Google Maps, MapBox and OpenLayers) and location information from GPS, Wi-Fi and Bluetooth.

A National example of this type of applications are the *Lisboa Cool: guia de viagem* application by Bloomidea (see Figure 2.22).

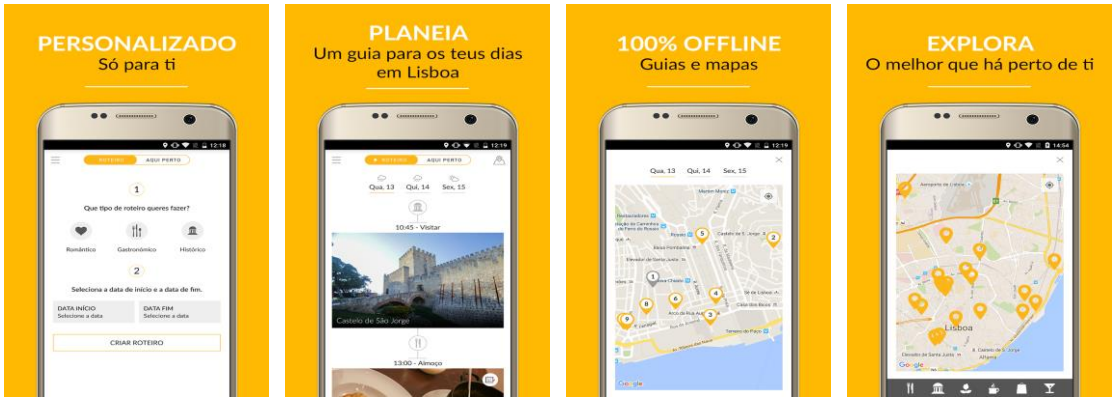

Figure 2.22. Lisboa Cool: guia de viagem app [31], Lisbon, Portugal.

These applications offer tourists and even citizens of Lisbon or Braga the possibility to create their own itinerary by selecting between the *Romantic*, *Gastronomic* or *Historic* options, as well as selecting the start and end date of the itinerary. A travel itinerary is then generated based on the user's choices.

The application provides practical information about each included point of interest, in text and image form, allows viewing the location of the point of interest on the map, as well as the user's position in relation to the point of interest. Weather information is also provided for the total number of days the route will last, and points of interest included in this route can be rated. The route can also be viewed in timeline or map mode (using the Google Maps service), where all points are pointed on the map, allowing the user to request directions to the points of interest. The user can click the points on the map to get more details about them. Finally, the APP also recommends attractions close to a given location.

Another example of an application that falls into this category is the *Visit London Official City Guide* application by London & Partners Ltd (Google Play store). This application offers recommendations of points of interest that can be visited by selecting a theme, or simply by the user location, which can be viewed on the map (using the Mapbox map service). This APP also allow saving points of interest for viewing latter offline. It is also presented information about the Underground schedule and directions to reach certain point, as well as ticket price.

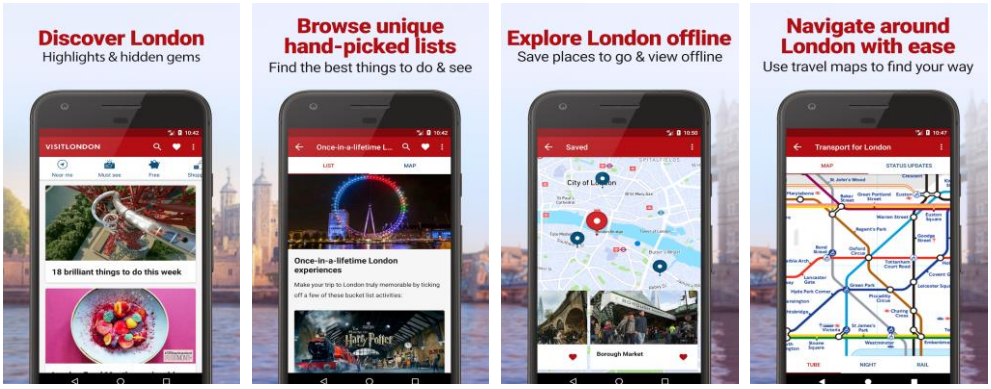

Figure 2.23. Visit London app [32].

As in museums, there are also universal applications for visiting cities. One example is *Guides by Lonely Planet* application by Lonely Planet [33]. This application allows the user, through a single application, to get guides for different cities (including Lisbon), all concentrated in a single application. All features work offline except for the guide download. This application offers a selection of points of interest that should be visited, and users can filter them. For each point of interest, the user can get general information about the point in question, information about points near that point, and directions to that point. Information is also provided on the different areas (neighbourhoods) in which the city is divided, on transport to get to and out of the city, and even a currency converter and money-saving recommendations. This app also enables saving points of interest.

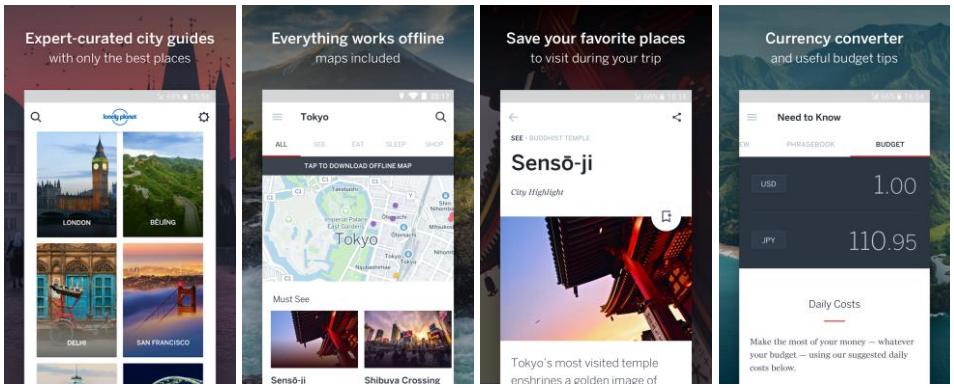

Figure 2.24. Guides by Lonely Planet app [33].

Furthermore, an example of a digital guide for a larger area such as a country is the *Visit Korea: Official Guide* created by the Korea Tourism Organization [34]. This application provides information about Korea, namely, traditions and historical cities. It also offers recommendations on what to visit in Korea, presenting practical information about each point of interest, and recommending related points of interest. The application provides directions to a certain point including transport information. Also, the APP provides information about attractions near the user, allows filtering of points of interest and getting directions to those points of interest. Storage of points of interest for later viewing and discount coupons are also provided by this APP.

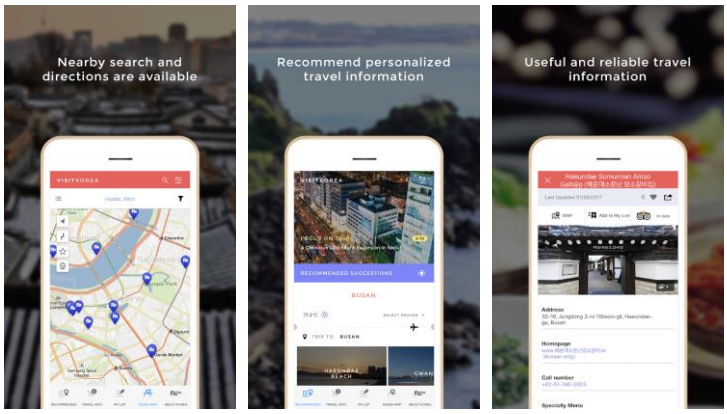

Figure 2.25. Visit Korea: Official Guide app [34].

The *TripAdvisor* APP represents a category of places-driven APP focused primarily on user content creation. Users recommend places among themselves and evaluate the places they visit. This application also allows to find restaurants, hotels, routes, etc. The APP has tools for creation, saving, and sharing user-created trips (tours) based on recently searched and visited places, as well as places that user bookmark.

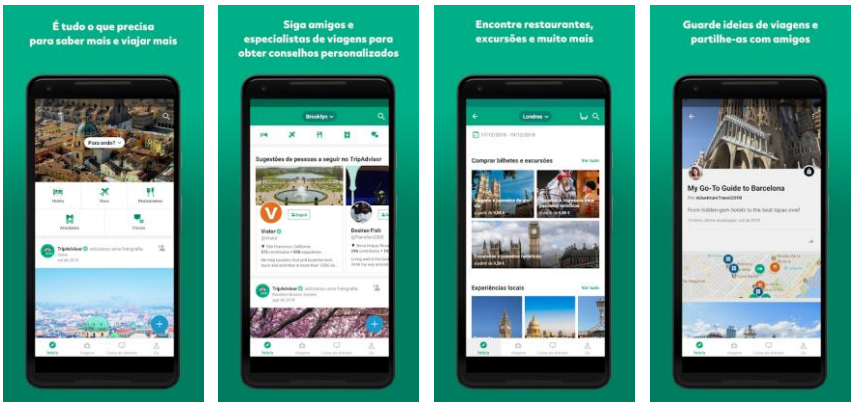

Figure 2.26. *TripAdvisor* app [35].

Although these applications deal with larger areas than botanic garden areas, technologies used for location, navigation and recommendations are similar. While applications for botanical gardens will only deal with one garden area, they should offer the user interactive navigation tools that allow them to see points of interest on the map and get directions to those points of interest. Features such as in the *Lisboa Cool* application that allow to generate a route based on the options that user choose, and like in the TripAdvisor application to create a route and share it with friends, were considered of great interest to implement in the application developed in the present work for the development of a APP for Tropical Botanical Garden from Lisbon.

2.3 Comparison of application features and technologies

A comparison was made of the identified features and technologies used by the applications presented in this section. Table 2 and Table 3 summarize features of the analysed applications and some technical details. In both of these tables the applications are separated into groups based on the categories proposed by Velho et al. [7], with the addition of the "Applications for botanical gardens" category.

In Table 3 the presented applications in this section are compared relative to the features and contents they possess. They the first column following the "Name" starting from the left, "Tour", compares the application in terms of whether they provide a curated set of tours. The automatic "Automatic Routing" concerns whether the application can calculate a trajectory automatically that takes visitors from any location A to any location B. The "Map", column indicates whether the application offers a map of the institution. If an application offers automatic routing, naturally it will also offer a map of the institution, however not all application that contain map also contain automatic routing. Next the "AR" table, indicates whether the applications use AR in any manner. Finally, under the "Contents used in POI presentation" section of the table, are indicated the contents that are used to present the points of interest featured on the applications.

Table 3. Features of related APPs

Name	Tour	Automatic Routing	Map	AR	Contents used in POI presentation				
					Text	Image	Audio	Video	Game
Applications that privilege content and deal with small enclosed spaces									
KHM Stories	✓	✓	✗	✗	✓	✓	✓	✓	✗
Rijksmuseum	✓	✓	✓	✗	✗	✓	✓	✓	✗
Rijkswidget	✗	✗	✗	✗	✓	✓	✓	✗	✗
Magic Tate Ball	✗	✗	✗	✗	✓	✓	✗	✗	✓
"How It Is"	✗	✗	✗	✓	✓	✓	✓	✓	✓

Muybridgizer[1]	✗	✗	✗	✗	✗	✗	✗	✗	✗
In Still Life[2]	✗	✗	✗	✗	✗	✗	✗	✗	✓
Margritte Your World[3]	✗	✗	✗	✗	✗	✗	✗	✗	✗
Tate Trumps[4]	✗	✗	✗	✗	✓	✓	✗	✗	✗
CloudGuide	✓	✗	✗	✗	✓	✓	✗	✗	✓
KeyARt	✓[5]	✓	✓	✓	✓	✓	✓	✓	✗
Applications that deal with location at a macro level									
Lisboa Cool	✓	✓	✓	✗	✓	✓	✗	✗	✗
Braga Cool	✓	✓	✓	✗	✓	✓	✗	✗	✗
Visit London	✓	✓	✓	✗	✓	✓	✗	✓	✗
Visit Korea	✓	✓	✓	✗	✓	✓	✗	✗	✗
Trip Advisor	✓[6]	✓	✓	✗	✓	✓	✗	✗	✗
Applications for botanical gardens									
Jardim Botânico RJ	✓	✓	✓	✗	✓	✓	✓[7]	✓	✗
Sydney Royal Botanical Garden	✓	✓	✓	✓[8]	✓	✓	✓	✗	✓[8]
RJB Museu Vivo	✓	✓	✓	✗	✓	✓	✗	✗	✗
Kew	✓	✓	✓	✓	✓	✓	✗	✗	✗
Jardim Botânico da Madeira	✓	✗	✓	✗	✓	✓	✗	✗	✗

In Table 4 are presented technical aspects of the studied applications. First the used Maps SDK by the application as well as the tracking technique used to locate the user in the map are presented. As we can see for most of the applications that privilege content and deal with small enclosed spaces, no Maps and tracking technique is used.

The Mafic Tate Ball uses GPS, however this is not used to locate the user in the Museum, instead the location of the user is added to the parameters combined in order to select an Art piece from the Tate Museum collection. In the case of KeyARt the tracking technique depends on the museum, as it is not just an application but a platform in which the various museums can publish their own AR experiences.

Next the operating system on which the application is (or was) supported is presented. As we can see the applications may not always be available for android, but they are always available for the iOS operating system, with the exception of the Rijskwidget which is a desktop widget. With the data from the next column we can observe that the most recently released applications are almost always available from both Android and iOS. Finally, in the last column, on the right, are presented the last published versions of the mobile application. For rows that are marked with a "-" it was not found their version since they weren't available anymore on the market.

Table 4. Technical details of related APPs

Name	**Maps SDK**	**Tracking Technique**	**Operating System**	**Publish Year**	**Last Version**
Applications that privilege content and deal with small enclosed spaces					
KHM Stories	✗	✗	Android and iOS	2016	2.0.13
Rijksmuseum	✗	Bluetooth	Android and iOS	2013	3.0.12

1 Create chronographs in the style of Eadweard Muybridge.

2 It allows to reconstruct a work of the seventeenth century.

3 Applies an animation of Magritte's most popular work to a user video.

4 Cards Game.

5 Allows users to create their own route or choose a route.

6 It allows the construction and sharing of routes.

7 Not present in new version but present in Jobim Botanic version.

8 Available with the *Plants With Bite* APP that complements the path of the same name.

Rijkswidget	✗	✗	MacOS and Windows	2010	2.0
Magic Tate Ball	✗	GPS	iOS	2012	2.2
How It Is	✗	✗	iOS	2009	1.0
Muybridgizer	✗	✗	iOS	2010	1.2
In Still Life	✗	✗	iOS	2001	-
Margritte YourWorld	✗	✗	iOS	2011	1.0.1
Tate Trumps	✗	✗	iOS	2009	-
Cloud Guide	Google Maps[9]	GPS	Android and iOS	2014	4.0.7
KeyARt	✗	Depends on the Museum	Android and iOS	2016	Varies with the operating system
Applications that deal with location at a macro level					
Lisboa Cool	Google Maps	GPS	Android and IOS	2018	1.1.7
Braga Cool	Google Maps	GPS	Android and IOS	2018	1.0.7
Visit London	Mapbox	GPS	Android and IOS	2017	2.34.0.253
Visit Korea	Google Maps	GPS	Android and IOS	2013	4.2.20
Trip Advisor	✗	GPS	Android and IOS	2016	29.2
Applications for botanical gardens					
Jardim Botânico RJ	Google Maps	GPS and Bluetooth	Android and IOS	2018[10]	1.1.0
Sydney Royal Botanical Garden	Garden's Map	GPS	Android and IOS	2017	1.20
RJB Museu Vivo	Aruba Beacons	Bluetooth	Android and IOS	2018	1.1.9
Kew	Google Maps	GPS	IOS	2011	-
Jardim Botânico da Madeira	Google Maps	GPS	Android	2014	1.0

The analyzed apps could be categorized in different ways:

- Content privileged apps
- Apps with or without navigational tools
- Apps with different type of interaction (i.e., use of visual, auditive, movements information for interaction)
- universal apps
- Apps with and without features for customization
- Apps that use or not Virtual Environments (i.e., use of Augmented Reality)
- Apps that allows or not social networking or sharing the data recorded during using the app
- Apps that have tools or not for generation of contents by the user (i.e., sharing the user story related to cultural institution or object of collection)
- Apps that have or not analytical tools to optimize the communication with users (i.e., to analyze user preferences, needs and desires, for user experience and usability evaluation, to identify the most engaging activities)
- Apps that used serious games or gamification, or not.

Different apps for botanical gardens or other cultural institution could be also categorized considering the context of use – outside institution, inside or both. By considering the temporal classification the apps for cultural institutions can be classified on apps that should be used before, during or independent of the visit.

[9] Used only to direct user to museum

[10] In fact, the first version can be considered in 2014.

The fast development of technologies for mobile apps increases the challenges related to customization and personalization of the apps. Integration of social networks, blogs, wikis in the apps for botanical garden may create a diverse and engaging way of communicating with visitors. An app for botanical gardens should not only contain information on plants, educative information related to ecosystem and navigation tools but also ludic, recreational features that would increase the users' interactions with app. The app should provide a multi-sensory, contextualized and as immersive as possible experience for the user. A balance should be found as the app would be not only "about something" but also "for somebody" [36].

2.4 Summary

This chapter the most important applications that are oriented to botanical gardens and other tourist guides such as apps for museums and city-guides. The applications for botanical gardens offered a great variety of perspectives on how to structure a mobile application oriented to this type of institutions. The applications oriented to museums offered a rich set of concepts that could be adapted to create more interesting interactions than the ones that are currently available.

By studying these apps, we also got a general understanding of what technologies are normally used to implement applications similar to the one that was implemented, and we were able to establish a set of requirements and use cases for the JBT app.

The following chapter presents the resources and methods used in the project, followed by a brief listing of the core requirements identified through the study of the app. It will also be presented the architecture of the system, an abstract representation of the structure of the mobile application, the data model. Furthermore, is presented all the work done designing the UI of the app starting from wireframes up to the current version of the app interface. Finally, the web services used for the application to communicate with the server are listed.

Chapter 3

Analysis and Design

This project consists on the development of a mobile application for the Tropical Botanical Garden of Lisbon (*Jardim Botânico Tropical de Lisboa*), with a curated list of tours tailored to the various visitor profiles.

In this chapter are listed the resources and methods used during the development of this project. Also, the artefacts resulting from analysis and design requirements executed during the development process are presented. First the core requirements of the system are listed, followed by the presentation of the system and mobile application architectures. Next the Data Model for the databases in the system are presented. After that the User Environment Design (UED) diagram, used to separate the different user actions into focus areas is presented, and also the wireframes and prototypes of the mobile application interface based on the conceived UED diagram. Finally, are listed the web services used to transfer data between the mobile application and the remote database.

3.1 Resources, tools and methods

To develop this project a study of the available solutions, for implementing a functionality or implementing part of functionality, was made. In this sub-section all the resources and tools that were available to us or used are presented. A tool is a software package, or a piece of hardware used in order to execute some operation, while a resource is an element, a person or a software library, that is directly integrated in a project. Alternatives considered while choosing each of the software resources are also presented.

3.1.1 Resources and tools

Human Resources

The project involved a team comprised of 15 participants. The vice-rectors of the University of Lisbon, Professor José Manuel Pinto Paixão and Professor Maria Dulce Pedroso Domingos coordinate the project for the project *Jardim XXI,* which aims to rehabilite the Tropical Botanical Garden of Lisbon and also the proponents of the JBT mobile app.

The development team from Faculdade de Ciências of Universidade de Lisboa (FCUL) includes six people. This team consists of two Master students – Stefan Postolache (me) and Rafael Torres and their respective advisors Professor Ana Paula Pereira Afonso and Professor António Manuel Silva Ferreira, and Professor Beatriz Carmo and Professor Ana Paula Boler Cláudio.

Additionally, for each tour featured by the app, a team of two or more people was in charge of providing contents to the mobile application. These elements were also used to evaluate the design of the tour. As such five teams were integrated in this project:

- *Botânico* tour team: Maria Cristina Duarte
- *Histórico* tour team: Ana Godinho Coelho Dotti De Carvalho, Raquel Barata
- *Aves* tour team: Ana Leal, César Garcia
- *Sensores da natureza* tour team: César Garcia, Cecília Sérgio and Palmira Carvalho
- *Lúdico* tour team: Ana Godinho Coelho Dotti De Carvalho, Raquel Barata

Furthermore, Professor Paula Maria Ferreira de Sousa Cruz Redweik from the Geographical and Geophysical Engineering and Energy department (*"Departamento de Engenharia Geografica Geofisica e Energia"*) of the Faculty of Sciences, contributed to the project with geographical data on the locations of plants and buildings in the garden.

Software resources

Software resources are packages used to support the application namely a database, a library or a simple script. It was necessary, at the beginning of the project and whenever a new component was added to the system, to research the available software resources in order to speed up the implementation of that same component.

For none of the components were found full-experience (functionalities developed in other projects that could be used in this) or partial-experience (functionalities developed in other projects that could be modified for use in this). However, several off-the-shelf resources were considered (software resource developed by third parties that could be used in the project, as they are). For each component it was necessary to choose between considered alternatives, which best fits the requirements of each component bringing more advantages.

As a **local database**, used to cache data needed for the application to work offline, the *SQLite* database [37] was chosen along with Room persistence library [38]. *NoSQL* bases such as *Oracle Berkeley DB* [39], *Couchbase Lite* [40] - a lightweight, document-oriented (*NoSQL*), syncable database engine for .NET or *Level DB* [41] - a fast key-value storage library written at *Google* that provides an ordered mapping from string keys to string values, weren't chosen as it is very difficult to implement relationships between database tables.

Realm object database [42] that claims creation of "reactive mobile apps in a fraction of the time" has not been adopted despite it offers higher performance than *SQLite* because it isn't as tightly integrated with the *Android* development environment as the *SQLite* database. Moreover, many of the advantages that the *Realm* database offers have been matched by *SQLite* with the introduction of *Room.*

Room Persistence Library solved most *SQLite* security issues. It is now possible to synchronize objects with the database due to their direct integration with *LiveData*, a lifecycle-aware observable data structure. Migrations are also simpler to implement and concurrent access to the database is supported [42]. Also considering that the app development team has experience on using *SQLite* in *Android* development, no time was spent to familiarize with the technology.

To represent the available tours' paths, we have decided to use a raster map of the garden. A set of interactive map *SDKs* were considered while making this decision: *Google Maps* [43]; *Mapbox* [44]; *NextGIS* [45]; *Osmdroid* [46] and *Maps.me* [47]. All these *SDK*'s except *Google Maps* are based on *OpenStreetMaps* [48] - a collaborative project designed to create a free editable world map.

Figure 3.1 shows a satellite view of the garden's area (a), the Google's map representation of the area (b), the OsmAnd [49] (maps application based on OpenStreetMaps geographical data)

representation map representation of the area (c). It is possible to observe that with the Google maps service the pedestrian paths within the garden are not represented.

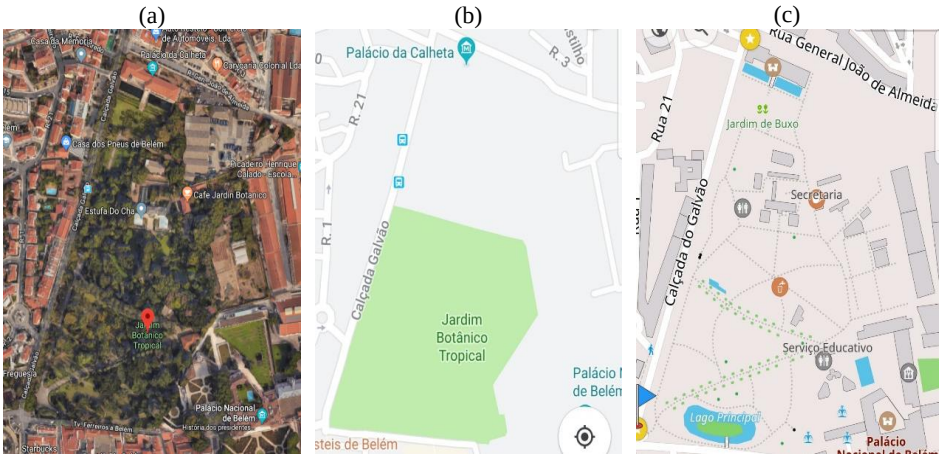

Figure 3.1. Map service comparison. Comparing satellite view (a), with the provided map by Google Maps (b) and the provided map by OpenStreetMaps (c).

Although *Google Maps* has implicit pedestrian paths in the garden, these do not include much of the existing paths. It would be possible to fix the *Google Maps* and add the missing paths, however the process of adding these takes time, and currently the changes made to the map are paid. For these reasons *Google Maps* was discarded.

In *OpenStreetMap-based SDKs* (*Osmdroid Map*), pedestrian paths are represented with dashed lines, and are represented most of the streets in the garden. Based on the criteria of the pedestrian paths of the garden being visible on the map, we proceeded to use *OpenStreetMaps*-based SDKs.

However, during testing was discovered, through the visualization of high precision data provided by the garden's staff involved in the project, that in OpenStreetMaps the shape and paths of the garden were wrongly positioned. The differences between the map obtained with *OpenStreetMaps SDKs* and garden official geographical data can be observed in Figure 3.2. We can observe in the closeup on the right that the paths provided by OpenStreetMaps (dotted orange), are slightly deviated from the correct paths (solid blue).

The changes needed to correct *OpenStreetMaps*-based maps, although not entailing any additional monetary cost, would be very time-consuming, as not only the orientation of the paths was wrong, but also the proportions and orientation of the tiles were wrong as well. It would also be necessary to correct the positioning of the buildings near the garden. For this reason, *OpenStreetMaps SDKs* were excluded from the app resources

Figure 3.2. Overlay of geographical data onto OpenStreetMaps map. Overlaying of the map based on official geographical data from the botanical garden onto the map provided by open street maps. In the close up on the right we can better see that the paths provided by OpenStreetMaps (dotted orange), are slightly deviated from the correct paths (solid blue).

Although map SDKs make it easier to implement certain features such as marking map locations and auto-routing, since all the studied SDK's did not meet the requirements of this project, and the garden comprises a small geographical area, it was decided to implement our own map.

Web services that retrieve data regarding tours, points of interest and their respective contents were developed. The mobile application invokes these services through a *REST API*. Thus, for the application to take advantage of *Web* services, two components are essential: an *HTTP* client, a component that accesses a given *API* address and receives the results of its access in *JSON* format; and a *JSON* interpreter, component that maps *JSON* objects to *Java* ones.

The *Retrofit Library* [4] was adopted as an approach to these components. This library adds an abstraction layer in communication with the server, turning the *REST API* into a *Java* interface. This library is built on top of *ok Http HTTP* client [50]. The mentioned library also allows the addition of a *JSON* interpreter, and for this purpose the *Moshi* interpreter was adopted [51]. This interpreter, being developed by the same team that developed the Retrofit library and the *okHttp HTTP* client, offers a better integration and thus better performance in interpreting the server response.

The web server chosen was *NGINX* [52] for its high performance, offering a higher performance and endurance than *Apache* [53], and its scalability, allowing thousands of simultaneous accesses without considerable impact on response speed [54].

To implement the **web services** was used *Lumen* [55] *PHP* micro-framework, based on *Laravel* [56] framework components. This framework was chosen due to the fact that is very lightweight, offers great performance, while at the same time offering the convenience of Laravel [55]. In addition, since this framework is powered by Laravel components, and I have used Laravel in previous projects (for example in the Information Technology Project course of my bachelors), it required a short adaptation time.

As the **remote database**, used to store data regarding tours, points of interest and contents of the application, as well as geographical data of all application users, *PostgreSQL* [57] with the *PostGIS* [58] extension, was adopted. PostgreSQL is a relational DBMS, which is consistent with the local database (*SQLite*). This DBMS is also open-source and free, with all its features accessible under the FLOSS license.

Additionally, since we need to store data related to locations of points of interest and the trajectory of the visitor, and also to easily calculate the shortest route that connects the points of interest in future versions of the application, support for geographical types are necessary. *PostGIS* is a spatial database extender for *PostgreSQL* database. It adds support for geographic objects allowing

location queries to be run in SQL. In addition to basic location awareness, PostGIS offers many features rarely found in other competing spatial databases [59] such as Oracle Locator/Spatial [60] and Microsoft SQL Server [61]

Software tools

Software tools are products, packages or languages used to execute certain operations aiming at the development or testing of system functions.

The *Web* services were implemented in *PHP* using the *Lumen* framework. The code for these services was developed using the *Visual Studio Code* text editor, with the following extensions: *Laravel Helpers*, which helps the editor better understand the framework structure and syntax; *Laravel Extra Intellisense* which allows the editor to more easily complete the commands you are typing.

The services were tested using the *Mozilla Firefox* browser, *Postman* [62], an *HTTP API* testing tool, and *Logcat* an *Android Studio* component used to log messages that the application sends to the *IDE* when the application has been sideloaded on a real or virtual device.

To inspect the *SQLite* database data from mobile devices, the *sqlite3* package was used through the terminal.

To access the remote database (*PostgreSQL*), the server was accessed using the *ssh* command of the terminal. Database management was done using the database command line (*psql*) on the terminal. Migration files (files used to define database tables) [63] and seeds (files used for entering data) [64] were also used, which were executed using the artisan command line interface included with the *Lumen* framework.

Balsamiq Wireframes [65], a wireframing tool was used to create wireframes of the different screens of the application, and to create low fidelity prototypes.

To view geographical data it was used the *QGIS* [66] geographical information system, as it is free and offers most features that commercial systems such as *ArcGIS* [67] provide.

To control the versions of the software it was used *Git* [68], and as remote central repository was used the *GitHub service* Pro version, which allows the creation of private repositories.

The mobile application was developed in *Java* using the *Android Studio IDE* [69]. With the help of this IDE also various devices were emulated, whose specifications are available on Table 5. Devices with different screen resolutions were created in order to guarantee that the interface looks well even in devices with lower screen resolution. The lowest supported *Android* version supported by the developed application is version 5 (*Lollipop*), and the latest version of Android, when the application was developed, was android 9 (*Pie*). Considering the lowest supported version and the latest version of the Android operating system at the time, these devices were created in order to cover most of the spectrum of devices in which the app may be installed.

Table 5. Emulated devices and their specifications.

Name	Screen Resolution	Android version
Pixel 3	1080 x 2160	9
Pixel	1080 x 1920	8
Nexus 5X	1080 x 1920	6
Galaxy Nexus	720 x 1280	6
Nexus One	480 x 800	6
Nexus S	1080x1920	5

Hardware resources

All code in this project was developed on two laptops: one with Intel Core ™ i7-8750H CPU @ 2.20Hz 2.21Hz, 16GB RAM, with Windows 10 Pro x64 operating system; another with an Intel Core ™ i7-7700HQ CPU @ 2.80Hz, 16GB RAM and Windows 10 Home x64 operating system.

To test the application were used mobile devices with various specifications. These specifications can be seen in Table 6. Similar to the emulated devices, each device has a different Android version and screen resolution, in order to cover most of the spectrum of devices that the application would be supported in.

Table 6. Physical devices used to debug the application.

Name	Chipset	CPU	RAM memory (GB)	Android version	Screen resolution (pixels)
Samsung Galaxy S9	*Exynos* 9810 Octa	octa-core (4x2.7 GHz Mongoose M3 & 4x1.8 GHz Cortex-A55)	4	9	1440 x 2960
Huawei P10 Lite	HiSilicon Kirin 658	octa-core (4x2.1 GHz Cortex-A53 & 4x1.7 GHz Cortex-A53)	4	8	1080 x 1920
Samsung Galaxy S4	*Exynos* 5410 Octa	octa-core (4x1.6 GHz Cortex-A15 & 4x1.2 GHz Cortex-A7)	2	6	1080 x 1920
Motorola Moto E (Gen2)	Qualcomm Snapdragon 200	quad-core 1.2 GHz Cortex-A7	1	6	540 x 960

As an application **server** a virtual machine was used in one of the machines of the rectory of the University of Lisbon, with a two-core processor and 3GB of memory RAM. The application server hosts all the data and files related to the tours and points of interest featured in the mobile application.

3.1.2 Methods

In this subsection are presented the methods that were used to develop the mobile application.

Literature research

Search and analysis of published works about apps for public gardens and other tourist guides were performed on the ACM digital library. In addition, apps were downloaded and studied from *Google Play* and Apple's *AppStore*. Features and functionalities of the studied apps were analysed, and differences, advantages, drawbacks and limitations of these apps were identified and analysed.

Interviews

Interviews with specialist from Tropical Botanical Gardens were realized both for requirements elicitation, content organizations, and evaluation of app prototypes.

Content Inventory

Sessions of discussion were realized with the two Professors that supervised this thesis, as well as with specialist in Botany, Ornithology and History from Tropical Botanical Garden from Lisbon in order to create an inventory contents available to be included in the APP. A list of contents and their attributes was realized.

Prototyping

Low-fidelity prototypes was realized using paper drawing as well digital tools for performing prototyping (for example *Balsamiq Wireframes*). The prototyping was mainly used for organizing the interface of the application and to identify requirements for the application.

Rapid iterative testing with experts

Evaluation of various protypes was realized in order to identify problems and fix them. A new prototype with solutions to the identified problems in previous evaluations would then be verified using a rapid test-fix-test-fix approach.

The evaluation of the developed interface of the app was carried out repeatedly during iterative design process in collaboration with members of the team, in order to detect the baseline usability problems and to improve the app. A multidisciplinary team, with researchers from the areas of geographical data visualization, botany, ornithology, history and computer science, was involved in the development of this project. Each of the tours available in the developed app was implemented in collaboration with researchers with a corresponding area of expertise.

The rapid iterative testing and prototyping techniques were included in the development process to help the researchers with whom we are collaborating to define the requirements for mobile app. Considering the feedback given to us about the content inventory and prototypes, the planning (workflow) for the next iterations was modified, solution were found and new version of the app was implemented.

User feedback

Feedback about how to improve the usability of the app interface were obtained from presenting the app to participants at the *"Encontro Ciência 2019"*. Participants were observed while they used the app to understand if exists different approaches to completing different tasks, such as navigating back to the home screen. Additionally, it was observed if the participants give up or resign from the process, express surprise, delight or frustration or confusion, as well as if something is wrong or doesn't make sense. Suggestions for interface improvement or the flow of events were documented.

Development Process

This project was developed using a process similar to the Unified Process [70], an iterative process, in this case a comprising small number of iterations, with activities in these iterations not being executed for all workflows.

All iterations took approximately two weeks and ended with monitoring meetings involving my advisors, as well as the student implementing the AR experiences (Rafael Torres), and his respective advisors. In these meetings the progress made in the development of the application was presented, issues and challenged identified would be discussed and was planned the work to be made in following iterations.

Similar to the unified process, most to the activities executed were made in the **Requirements**, only some being made towards the **Analysis and Design** workflows and **Project Management**. In subsequent iterations more activities began to be directed towards the **Implementation** and **Analysis and Design** workflows.

The application currently is not ready for **Deployment** as some tours are still to be implemented such as the Nature's Sensors (*Sensores da Natureza*) tour.

3.2 Requirements

A set of functional requirements (FR) were defined based on the objectives of the project as well as literature research and interviews with other members of the Tropical Botanical Garden of Lisbon.

- **FR1** – offer the users structured tours. Structured tours follow a certain route and along that route points of interest are presented in a certain order. For each point, information

about the object should be presented. An indication of the progress the visitor also has to be made available.

- **FR2 –** for some tours is expected to have filters that correspond to parts of the tour, identified by their age (for example the Historic tour). This type of tours follows the same philosophy of the structured, and only the filtering functionality is added.
- **FR3 –** offer users a tour that does not have a structure where users can filter points of interest by their type. Additionally, a route should be calculated that leads the visitor to the points of interest.
- **FR4 –** offer users with families a tour where only one of the points of interest is visible at the beginning. The visitors then need to pass the challenge presented, which can be a puzzle, quiz or a team orientation challenge, for the next point of interest to be revealed. This process is further repeated until the team reaches the end of the tour.

In addition, a non-functional requirement (**NFR)** was instituted in the project's description that dictates that the app must be developed for the *Android* operating system.

These requirements represent the core functionalities and system features of the conceived mobile application.

3.3 System architecture

Figure 3.3 shows the main components of the system and the dependencies between them: the mobile app, the *Nginx* web server, web services, the public storage and the geographical database.

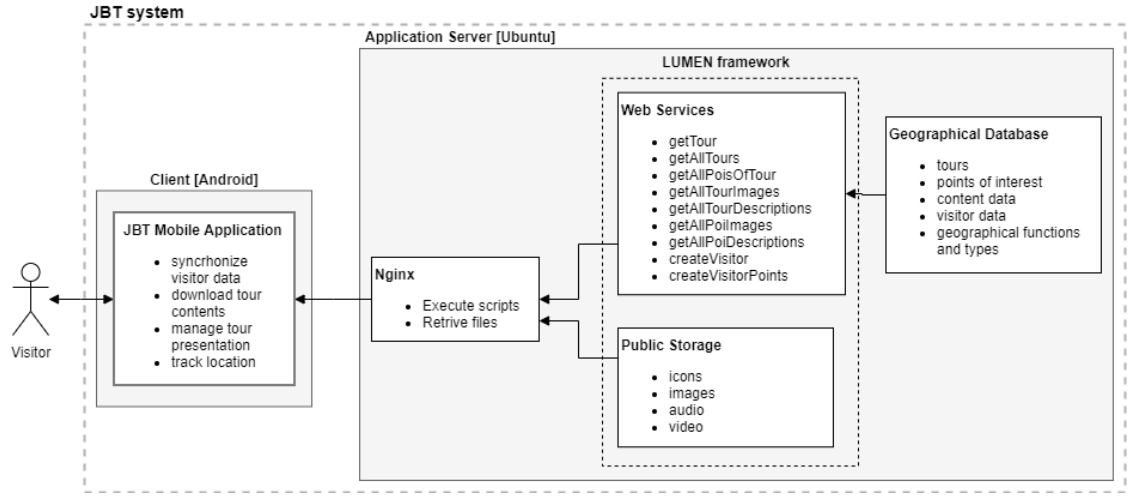

Figure 3.3. JBT system architecture.

The system follows a client server architecture. The client comprises all the devices of the garden's visitors running the Android operating system, and JBT mobile application. The server is a virtual machine hosted in one of the universities' server.

In this architecture the client contains the necessary resources to execute most of the work in the system, without connecting to the server. The client connects to the server to synchronize visitor data, such as demographic data or location data, and to download data and contents of the featured tours. The rest of the tasks such as tracking the location of the visitor and managing the presentation of content about the tour, and its points of interest, are done offline. This makes the system more scalable as the server has less payload, and so it can serve more clients. Additionally, the client makes the system more performant as most operations are processed locally, as such are not subjected to network latency. Finally, as contents of the application are downloaded and stored locally, the user experience of the mobile application is not influenced by the Wi-fi coverage, making the system more reliable.

The mobile application communicates with the **application server** by sending HTTP requests to it. For each request the **NGINX** server software tries to execute the script that interprets the requested URL. Since the NGINX server does not know how to compile and execute PHP, it opens a socket to PHP-FPM (PHP Fast-CGI Process Manager) [71], which assigns one of its managed processes to execute the script that is invoked by the URL. Once the request has been processed the response is delivered through the socket back to NGINX, which forwards the response to the mobile application.

However, if no service is found that responds to a given URL, NGINX searches for a file in the **public storage** of the web application, that matches the path specified in the URL.

The **web services** implemented using *Lumen*, a *PHP* framework, are used as an interface for the mobile app to interact with the remote database. These services query the data on the geographical database and convert the result set to a format that can be processed by the mobile application. Additionally, for insertion operations, a validation of the data sent by the clients is done, and after that the data is inserted into the remote database.

The **geographical database** includes data about the tours and the associated points of interest and contents, as well as data about the visitors (birth year, gender, occupation, qualifications). Additionally, this database also offers geographical functions and types.

The presented architecture differs from the one from the initial iterations. At the start of the project the system did not have a server, only the mobile application. This architecture was changed, since we wanted to store the demographical and trajectory data from the users in a central repository, for it to be used in future statistical studies. Furthermore, the size of the app, including the database and the contents, was greater than the permitted by the Google Play store.

In the following subsections first an abstract architecture of the mobile application is presented. Following, is the data model used for structuring the data in the remote database as well as the mobile application's database. Next artefacts elaborated in order to design the user interface are presented. Finally, the Web services are listed.

3.4 Mobile application architecture

A diagram was elaborated to get a general understanding of how the application would be organized, what components would it have and how would these components interact with each other (see Figure 3.4). In this diagram abstract components and layers are presented as well as their interactions. Components from a layer don't interact with each other. Components from upper layers coordinate the work exerted by components from lower layers.

Starting from the top, the "**presentation manager**" component comprises all the classes with the responsibility of displaying information about tours and their points of interest and capturing visitor gesture and touch events. To receive data, and when user events are captured, objects belonging to this component send requests or report events down to the "**decision and logic**" component.

The "**decision and logic**" component dispatches the requests or user events to one the components of the layer bellow. In this layer, each component is responsible for handling requests that concern a specific element of applications environment. In this case there are three elements: the visitor; the tour; and the point of interest. The "**visitor manager**" includes all the updates made to the visitor data, and the storage or synchronization of visitor location data. The "**tour manager**" handles requests for data and contents describing tours. Finally, the "**points of interest manager**" handles requests for data and contents related to a specific point of interest.

Each of the previously mentioned managers interact with lower level managers that manage more concrete concepts. One of these managers is the "**communication manager**" which handles all the logic concerning the retrieval or sending of data through the network. This manager, hence, is responsible for the interaction between the client and the server. Another component is the "**location**

manager", which manages not only location updates, but also orientation updates, which are used to display the position of the visitor on the map, as well as to compute the displaying of content in geographically constrained *Augmented Reality* experiences. The "**content manager**" communicates with the local storage and retrieves cached images, icons, audio and video files and *Augmented Reality* experiences. Finally, the "**database manager**" handles operations that need to be done to the data stored in the *SQLite* local database.

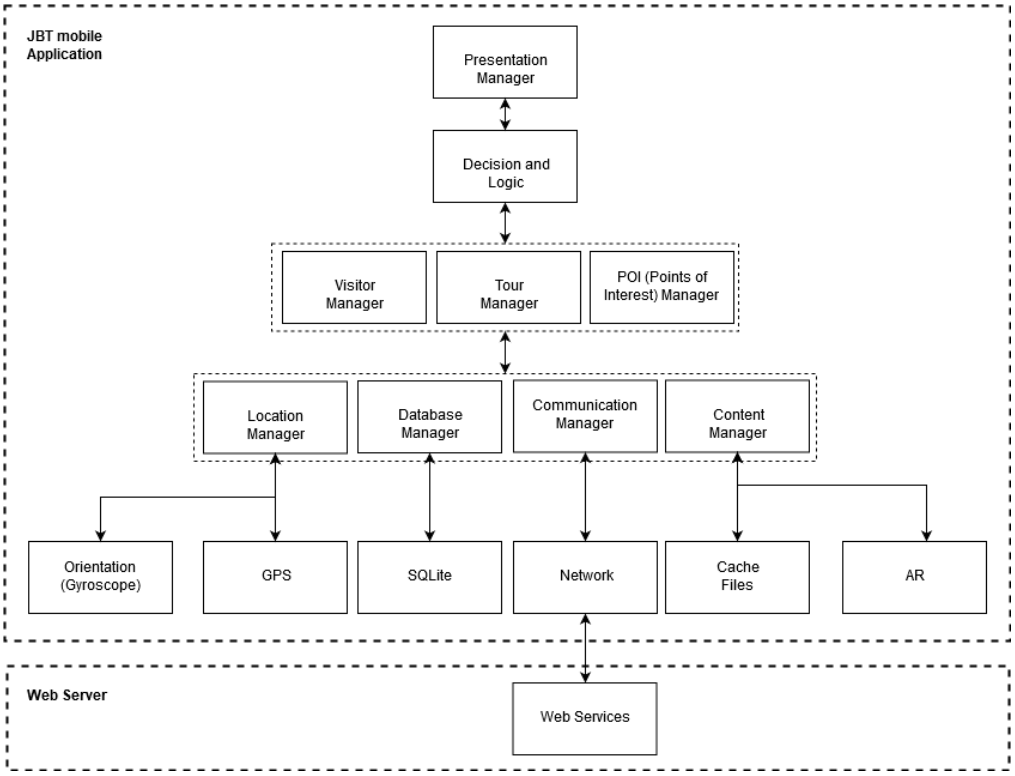

Figure 3.4. Mobile application software architecture.

3.5 Database

The data model is an abstract model which aims to organize the data elements of the system, and how these elements are related to each other. This model also represents the data required, its format, and also to prevent data redundancy.

In Figure 3.5 we can see all the entities included in the data model as well as the relationship between them. The crows' foot notation is used for representing entities and their relationships, and the SQLite types are used for the attribute types in the entities. Also, the foreign keys and primary keys of the tables are notated with **FK** and **PK** respectively. The diagram can be then separated into five areas: *Configuration*, *Visitor*, *Tour*, *PointOfInterest* and *Content*. Each of this area comprises one of the most important concepts, and other concepts tightly related to those concepts.

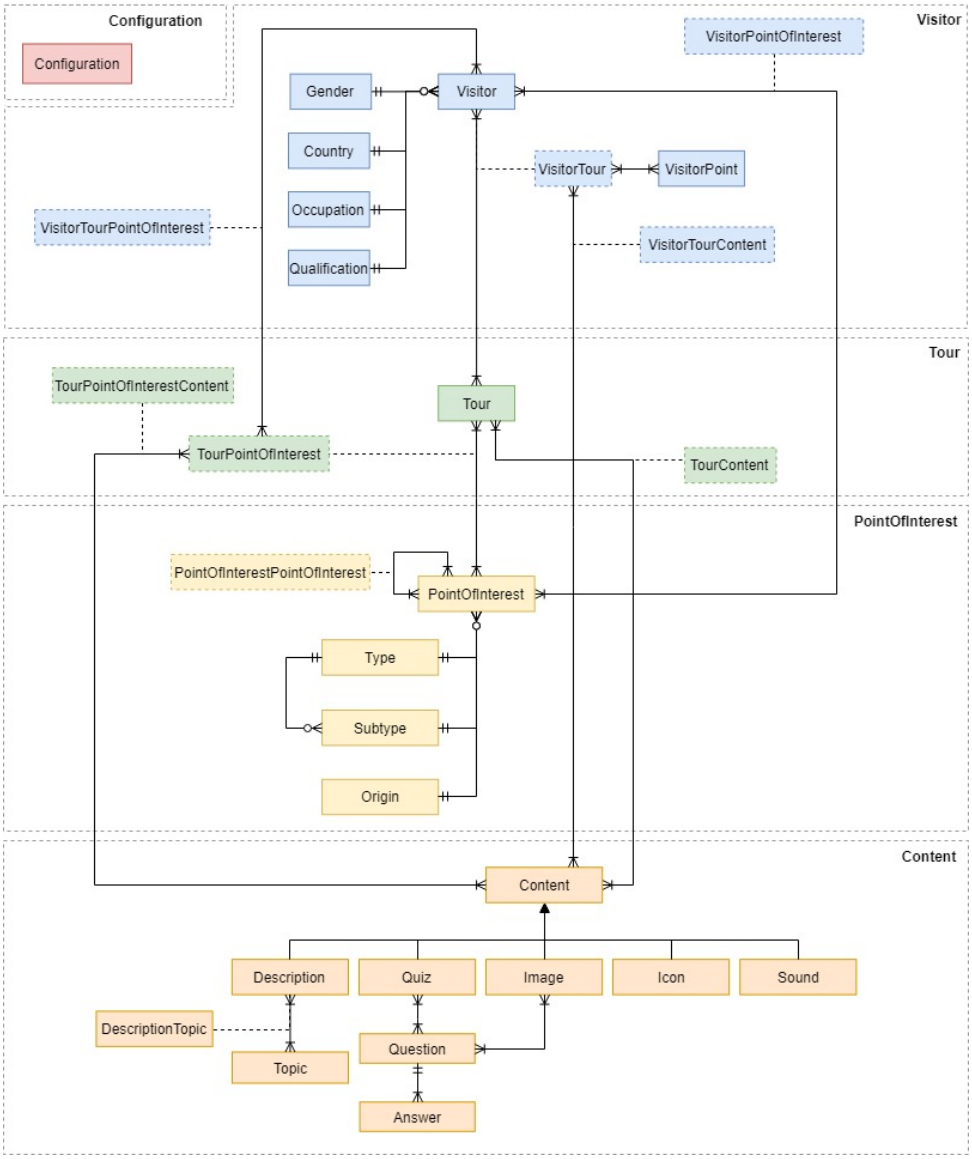

Figure 3.5. Data model with all entities and relationships split into concept areas.

The Configuration area (Figure 3.6) contains only one entity and is not related to any other entity in the model. The Configuration entity (table) is used to store all the data related to configurations in the app, such as the last synchronization time, which is the date in milliseconds of the last time the visitor's data has been synchronized with the server.

Configuration	
PK	id:INTEGER
	lastSynchronizationTime:INTEGER

Figure 3.6. Configuration conceptual area expanded containing only the configuration table.

Next in the Visitor area of the model (Figure 3.7), we have the Visitor entity as well as all the entities that store data related to actions the visitor does while using the mobile application. The Gender, County and Occupation entities help define columns (attributes) of the Visitor entity. VisitorTourPointOfInterest, VisitorTour, VisitorPointOfInterest, VisitorPoint and VisitorTourContent, relate the Visitor entity with other entities and store data related to visitors' actions, such as visiting a given point of interest in the context of a tour and favoriting a point of interest.

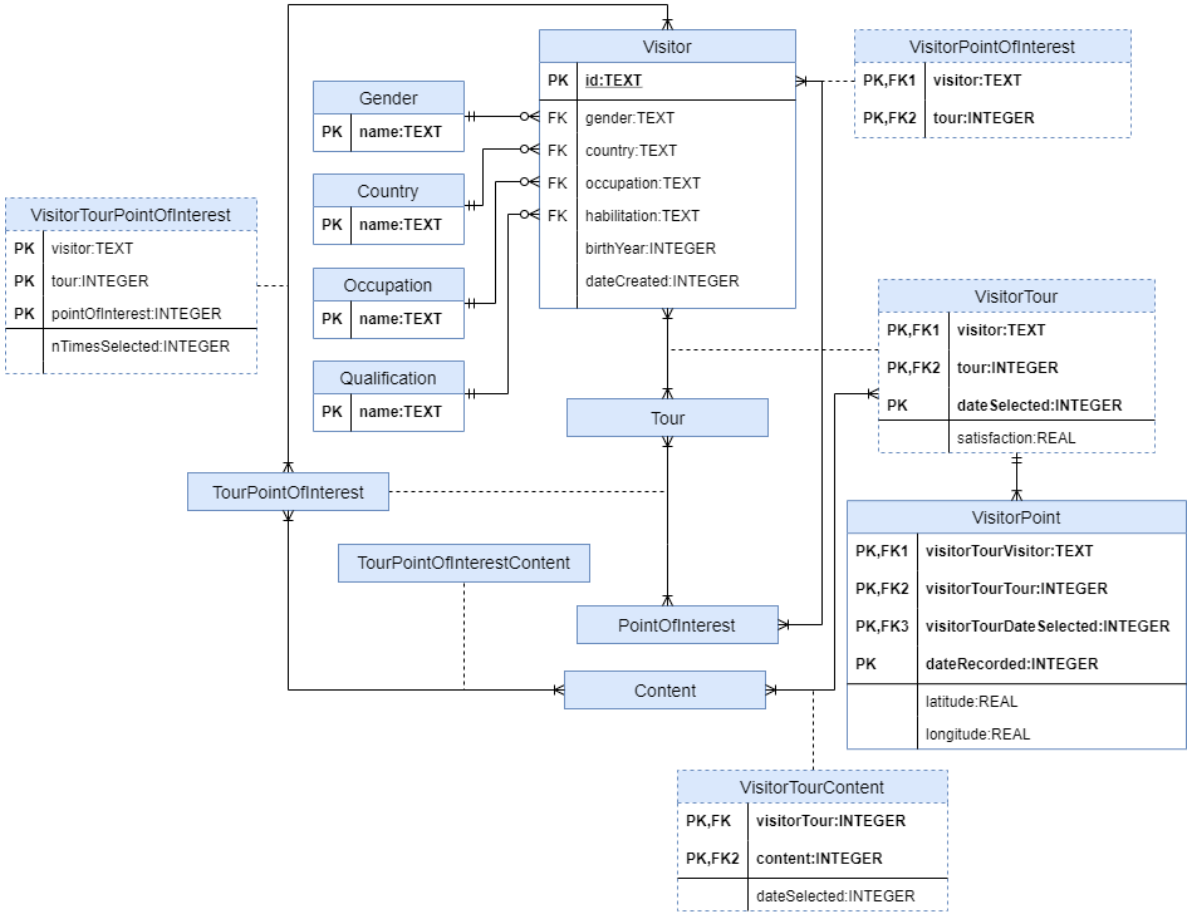

Figure 3.7. Visitor conceptual area expanded. The Gender, County and Occupation entities help define columns (attributes) of the Visitor entity. VisitorTourPointOfInterest, VisitorTour, VisitorPointOfInterest, VisitorPoint and VisitorTourContent, relate the Visitor entity with other entities and store data related to visitors' actions, such as visiting a given point of interest in the context of a tour and favoriting a point of interest.

When the user enters the app, a register is created with the visitor information namely: a unique identifier; the visitor's gender; the visitor's country; the visitor's occupation; the visitor's habilitation; the visitors birth year; and the visitors creation date. The unique identifier is a string made of seven random letters and the date and time of day in milliseconds. This format prevents conflicts when registering visitors. The gender, country, occupation and qualification have entities dedicated to storing values for these attributes averting possible insertion errors. There's no table for the *birthYear* as this can be dynamically generated and controlled. Finally, the Visitor has a creation date.

Relationship entities that contain data that the user creates while using the mobile application are also contained in this area. On the Right we have the *VisitorTourPointOfInterest*, which stores the number of times a visitor clicks a given point of interest in the context of a tour. When the visitor choses a tour the *VisitorTour* entity stores the number of times the user selected the tour. In addition, the user classification of the tour is stored in *satisfaction*. *VisitorPoint* stores all the visitor's locations (latitude, longitude) during the tour. The *VisitorPointOfInterest* relationship entity stores data about points of interest the visitor favorited, and the *VisitorTourContent* stores data about viewed contents. All time attributes in all these entities are stored as integer which represents the date and time of day in milliseconds when the data was recorded. In the remote database these values are converted to `timestamp` objects.

Following in the Tour area of the data model (Figure 3.8) we have the *Tour* entity and as well as all the entities that relate the tour with other entities: *TourPointOfInterest*, *TourPointOfInterestContent*

and *TourContent*. *TourPointOfInterest* and *TourPointOfInterestContent* are user to present the points of interest of the tour, while *Tour*Content is used to display contents when describing the tour.

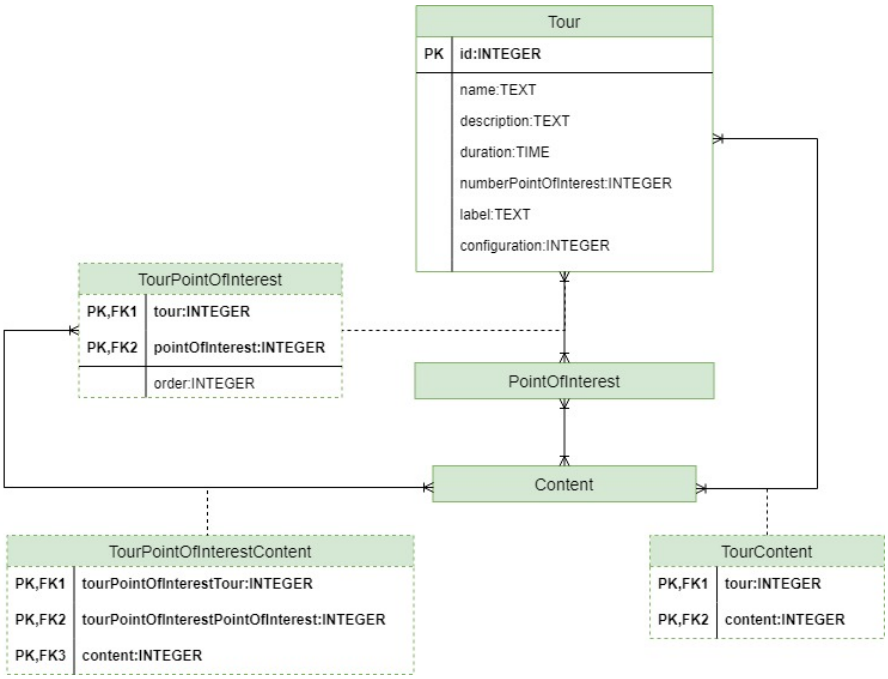

Figure 3.8. Tour conceptual area expanded. TourPointOfInterest and TourPointOfInterestContent are used to present the points of interest of the tour, while TourContent is used to display contents when describing the tour.

The attributes of *Tour* entity are: name; description; duration; number of points of interest; label; and configuration. The number of points of interests are now dynamically computed in the application. The label attribute is used to refer to contents that belong to it (e.g. for a tour with the label botanic, contents that start with botanic belong to it). Some tours imply specific behaviors while presenting the points of interest, such as the Historic tour that requires the filtering of points of interest. Therefore, the configuration attribute stores the behavior that the application must adopt while presenting the points of interest of the tour.

The *TourPointOfInterest* entity establishes what points of interest belong to the tour and the order these are to be visited. The *TourPointOfInterestContent* entity stores the contents that must be presented for a given point of interest. Finally, TourContent stores data regarding contents used to describe a tour This table is needed as icons and images are used to describe the tour in the app.

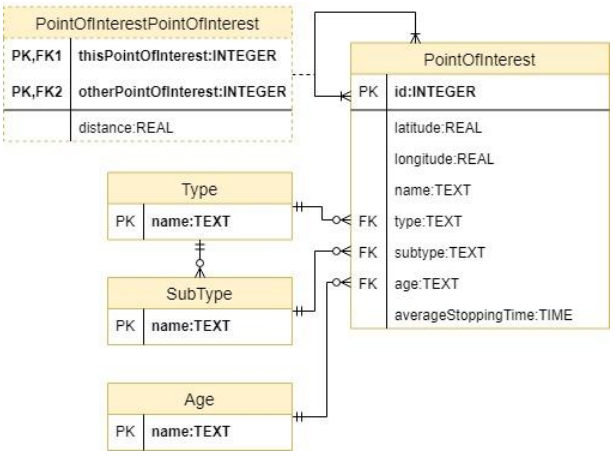

Figure 3.9. PointOfInterest conceptual area expanded. Type, subtype and age, are used to distinguish between points of interest, while PointOfInterest stores the distance between points of interest.

The Point of interest area contains the *PointOfInterest* entity, as well as entities that store values for its attributes or relationships between points of interests.

As we can see in Figure 3.9, the *PointOfInterest* entity is defined by a position (latitude, longitude), stored as floating point numbers, on the local database, and as a geography point on the remote database. The point of interest has also a name. The *type* attribute defines if a Point of Interest is a plant, building, statue, lake or bird, and for some types it is necessary to define a subtype. For example, if a point of interest is a statue the subtype could be a statue or bust. The Historic tour contains points of interests that belong to several subgroups, that the user can filter: XVIII-XIX centuries; XX century; Trees with history; and Portuguese World Expo (*Expo Mundo Português*). Each point of interest may belong only to one of these subgroups. The age attribute is used then to distinguish points of interest in the historical tour. The *PointOfInterestPointOfInterest* stores distance between points of interest.

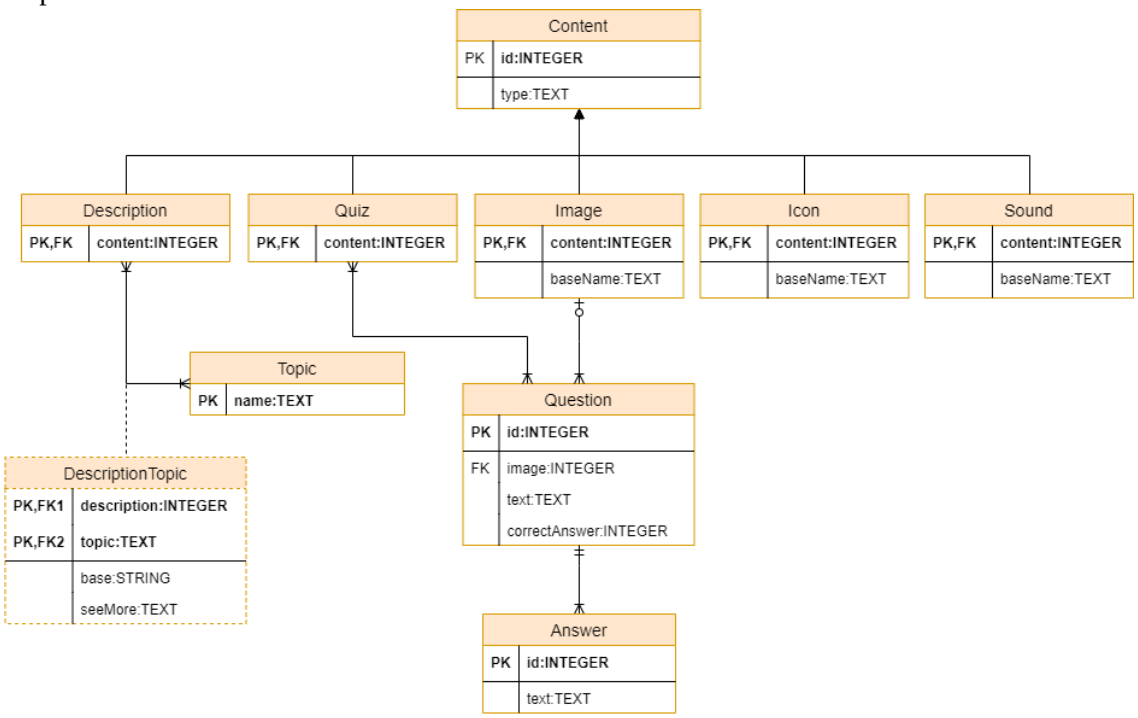

Figure 3.10. Content conceptual area expanded. Description, Quiz, Image, Icon and Sound are all the types of contents currently supported. Question and answer are entities that help structure the quiz and DescriptionTopic and Topic entities help structure the description, by separating it in different topics that are presented in a given order.

Figure 3.10 shows the Content area that contains all the entities related to the content of POIs.The Content entity is specialized in the entities Description, Quiz, Image, Icon and Sound.

The Description entity contains text describing POIs. In some tours (e.g. botanic) there are descriptions organized in Topics. When the number of characters for a topic exceeds certain threshold of 680 characters the text is collapsed, and only part of it is visible. *DescriptionTopic* materializes this structure by relating the various Topics to its respective Description, and also through the *base* and *seeMore*, the text that should always be visible and collapsed is specified.

Although the ludic tour was not implemented, a draft of the structure of the data describing a Quiz. The Quiz is then made of different questions that may or not be related to a picture and each question has only one correct answer.

Image, icon and sound file, are stored in a designated directory in the public storage of the web application and also the application's internal storage. For this reason, *Image*, *Icon*, and *Sound* store the trailing name component of the path (*basename*) is stored by these tables for each file.

3.6 User Environment Design diagram

Before designing the user interface, it is a good practice to draw a *User Environment Design* diagram. To define a coherent and structured product. In this diagram the application is organized into focus areas. A focus area is an abstract concept that consists of a designated working area where the user must be able to do a related set of activities. All focus areas included in the diagram are recognized by the user, i.e., the user must be aware that they exist.

Figure 3.11 shows the UED for the JBT mobile application with five focus areas: *Tour list*; *Tour details*; *Map*; *Point of interest list*; and *Point of interest description*. This area includes all features related with tours in the app namely the details of tours. From this point the user can choose one of the tours to read its description entering the *Tour Details* focus area. From *Tour Details* the user goes to Map area. In this area one of two scenarios can happen: the representation of the tour in the map is shown or the contents belonging to the tour are downloaded and after that the tour is shown. At any time, the user may cancel the download and go back to the *Tour Description* focus area, or it may go to the *Tour list* area directly. From the *Map* depending on the tour chosen, the visitor could go to *Point of interest list* (e.g. Birds tour), or to a *Point of Interest Details* focus area directly, by tapping on one of the positions marked on the map. The user can also filter the points of interest in some cases (e.g. Historical tour).

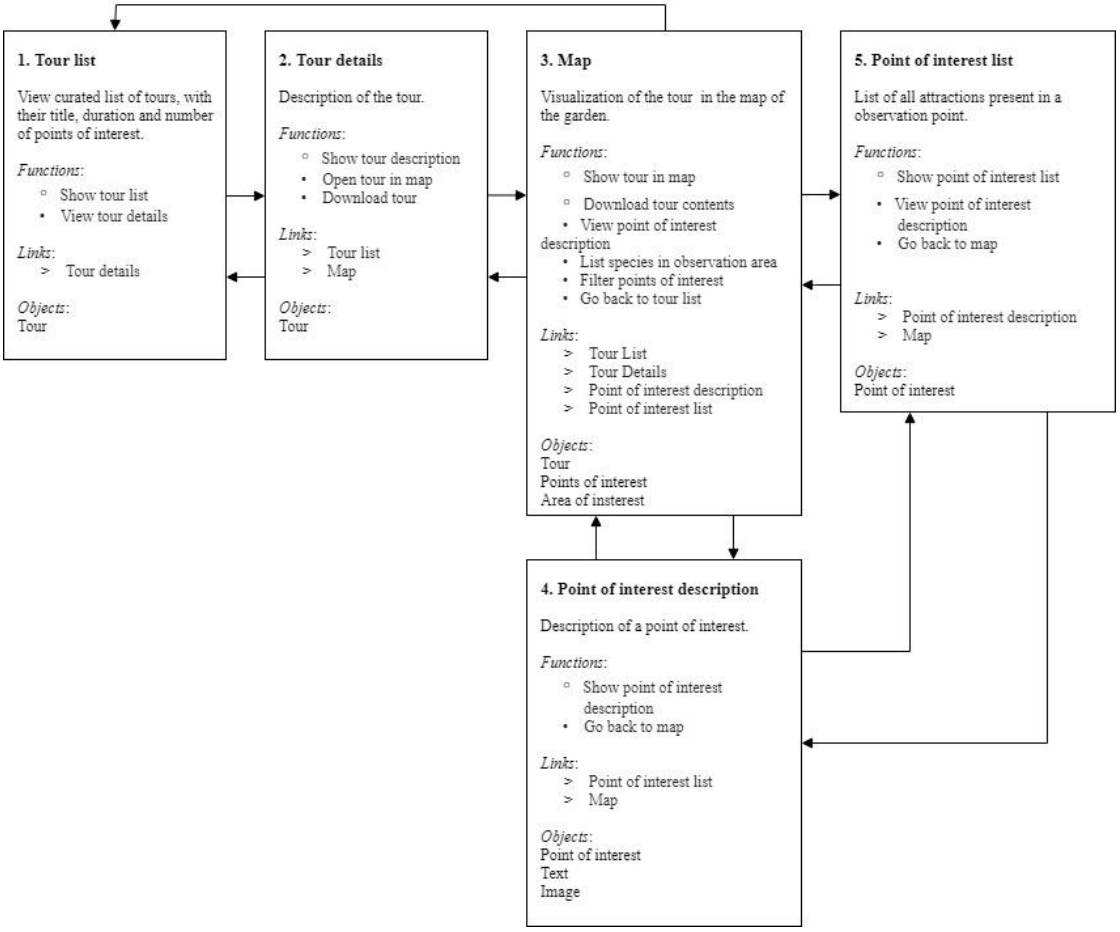

Figure 3.11. User Environment Design for the application. Functions marked with a solid black circle are functions that are triggered by user input while functions marked with a •.

In *Point of Interest List*, the user can choose one of the points of interest included and view its details (entering the *Point of Interest Details* area). The user can also return to the map. The *Point of*

interest list focus area is available only for specific tours namely the Historic tour. For others (e.g. Botanic tour) each point on the map leads directly to the *Point of interest details* focus area directly.

In a typical scenario the user would go from area 1 to 5, passing by area 2, in which he/she reads information about the tour, then goes to area 3 where the content is downloaded or if it already downloaded to view the tour on the map. Finally, he/she alternates between area 3 and 4, as he/she views the data related to each of the points of interest included in the tour and marked on the map.

3.7 User Interface

To design the user interface several artefacts were produced namely wireframes and prototypes. We start with the design of wireframes and early prototypes of the app user interface. Wireframes are early simple representations of the user interface. After conceiving the wireframes, and the use cases prototypes were developed in android. Based on the stakeholders' feedback and the interface design rules for android applications the interface was also improved.

3.7.1 Wireframes

Wireframes, also known as page blueprints, are visual guides that represent the skeletal framework of a UI application. The wireframes are the layout of a screen and describe how the elements will be arranged. In the case of this project the wireframes were used in order to create a terminology and structure in the interface that would meet the expectations of users.

To create these wireframes, we used the *Balsamiq Wireframes* [65] application which offers elements such as mobile phone frames, and components commonly used in wireframing such as buttons, images, and icon drawings. Additionally, links between different screens are permitted increasing the diversity of user interactions. The tool enabled quick and effective communication of the features included in a tour of the mobile application.

Two versions of the wireframes were created because in the initial stage of the application development the target users and the types of tours were unknown. The wireframes worked also as prototypes to support requirements elicitation.

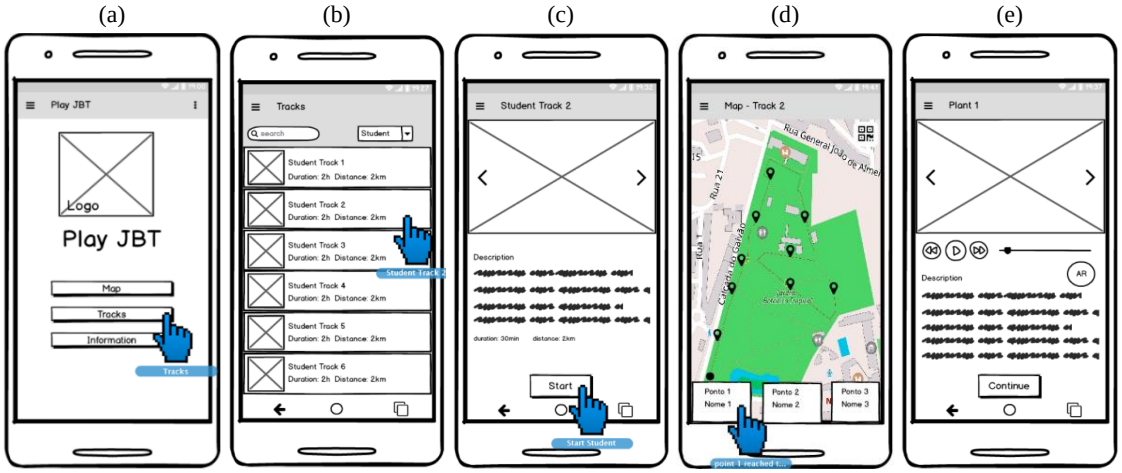

Figure 3.12. First version of the wireframes for the mobile app.

For the first wireframes, the target audience was divided in three groups: students and experts; tourists; families with children. The envisioned tours were:

- **structured** - a tour in which the visitor had to follow a certain order of points of interest, and the content were related mainly to details about the plants in the garden.

- **unstructured** - a tour in which the visitor wondered around and could get recommendations about plants.
- **playful** - in which challenges were presented at each point of interest included in the tour, and the visitors had to complete the challenges in order to be shown the next point of interest.

The students and experts were the target audience for the first two types of tours, and the last one was directed to families with children.

The first wireframes can be seen in Figure 3.12. In the initial screen (a), we had three modes of use: *Map*, *Tracks* (equivalent to tours), and *Information*. The *Map* mode was essentially the Free **unstructured** tour, where all points of interest were presented and the user could filter them, and also could get recommendations on what to visit next. The *Tracks* mode included all the structured tours. Finally, the *Information* mode, could offer visitors specific information about the garden such as, parking, schedule and garden regulations/norms. In Figure 3.12, a typical use of the *Tracks* mode is presented. The user can choose a track from the featured list of tracks and then click on the card corresponding to a certain point of interest, to know more about the point of interest. For the point of interest different types of contents are presented.

In Figure 2.13 the *Information* and *Map* modes are illustrated. At the top we have the filter bar which filters the visible points of interest by their type and subtype. In *(b)* we can observe that the *Trees* type has been expanded into its subtypes.

In addition, in (a) and (b) on the right side there are some additional icons. Starting from the top there's the QR icon which when click opens the QR reader that is used to obtain more information about a point of interest. Next, there's the search button which enables the user to search a point of interest by name. The compass icon rotates the map to match the orientation of the visitor in the map. The eye icon centres the viewport on the visitor's location. Once the visitor reaches a point of interest by clicking on the *AR* (Augmented Reality) button, he/she can view the augmented reality experience of that point of interest. Finally, the icon on the bottom right corner lists all the points of interest that are visible at the moment.

In "Info" mode *(c)* there's a list of cards containing practical information about the garden, such as schedule, pricing, and guided tours. In addition, there's a research button that leads to a screen presenting the research done at the garden. While in rules the garden's regulation is listed.

Figure 3.13. "Map" and "Info" mode wireframes.

In the second version of the wireframes several changes were made. Based on the meetinds with expert stakeholders it was identified the correct names of the tours and in the initial screen o the app the tours are the Botanic (*"Botânico"*); the Historic (*"Historico"*); and the Ludic (*"Lúdico"*). Later a

new tour was considered to be included in the featured set of tours, the Birds tour (*"Aves"*). For each tour it was also designed its interaction.

Figure 3.14. Second version of the wireframes.

In Figure 3.14 shows the second version of the wireframes. The initial screen now shows a list of the tours featured in the app, described by their title, total distance and duration (Figure 3.14 (a)).

All elements in the initial screen (Map, Tracks and Info) were moved to a sidebar (Figure 3.14 (b)). In addition, several new options were added. The *Favourites* allows users to see the points of interest they favourited. The *History* option includes a list of the last visited points of interests, for the user to review its contents. The *Notes* option would be added to enable visitors to take notes, and to review all their notes. The *Groups* option would enable the creation of groups in order to solve the group challenges of the tour, as part of the Ludic ("*Lúdico"*) tour. Finally, the *Share* option would enable the user to share notes or points that he or she included in the list of favourites, with friends on social media.

Next, the point of interest list at the bottom of the *Map*, in the Botanic, Historic and Birds tours, was replaced by a list icon on the top right corner of the navigation bar (Figure 3.14 (b)), which led to a point of interest list screen. The user could navigate back to the map by clicking the map icon of the navigation bar or select the point of interest he/she wanted to know more about.

At the time of Master thesis dissertation, the full implementation and tests were realized for the features related to *Tours* mode, i.e., the interaction with the different tours included in the application.

3.7.2 Prototypes

A prototype is an early version of a product, with a simulation or a partial implementation of the system's functionalities, to validate the design of the product. Along the development of this project many prototypes were conceived in order to test the design of the interface and interaction, of the application.

The first prototype (Figure 3.15) that was conceived was presented and tested at a Workshop that took place in the Rectory of the University of Lisbon, with all the stakeholders, and members of the team in charge of the project Jardim XXI. In this early version of the mobile application, the Botanic tour was simulated, with not real data, in order to show the main interaction in the app. With this prototype, feedback was received on the early interface design for the mobile application.

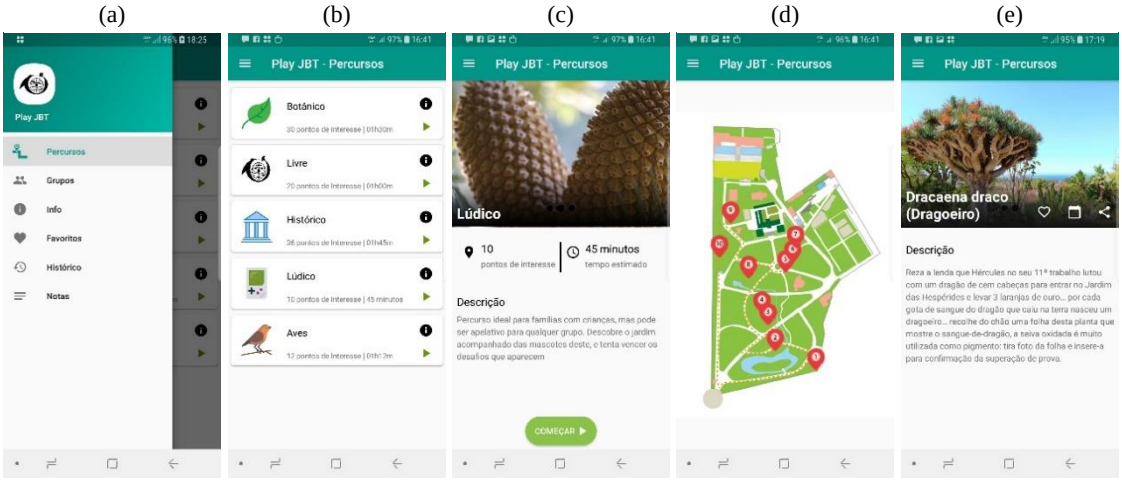

Figure 3.15. First prototype of the mobile app (high-fidelity).

In this version the main components of the application were already included:

- a side menu containing the various modes of use of this application.
- an initial screen, which contained the tours.
- a page describing the tour, from where users can start the tour.
- the map, where the user would navigate.
- and the description of the point of interest screen.

The icons used were not created by the same artist and followed different design languages (Figure 3.15), and a colour palette [72], with teal as the primary colour and lime as the accent colour, were used for the application in general. To create this colour palette, was followed the Google Material design colour theory, and suggested colours.

Each element in the list of tours had two buttons on the right (Figure 3.15 (b)), one to get a description of the tour, and another to go directly to the tour.

The map used in this version is also the map that the botanical garden provided to users, before rehabilitation works started. The map in all these prototypes enables the user to zoom and pan it. Lastly the description screen for the point of interest contained the favourite (heart), schedule (calendar) and share icons (Figure 3.15 (e)). The *Favourite* as the name suggests was to favourite points of interest. The *Schedule* was to schedule a point of interest to see later. The *Share* was to share the point of interest with friends on social media.

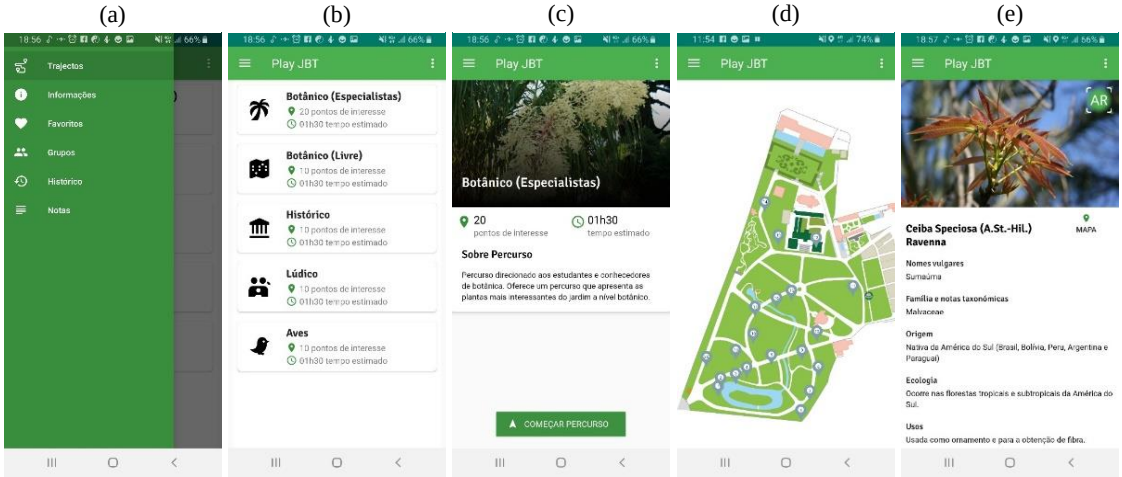

Figure 3.16. Mobile application second prototype.

Several changes were made mostly related to stakeholders' suggestions: the palette of colours, icons of the tours, the map and the POI markers. All these changes are illustrated in Figure 3.16.

Since *Augmented Reality* features were added to the mobile application an *AR* icon started to be added to the point of interest description screen (Figure 3.16 (e)). As could be seen the palette has changed to a monochromatic one, i.e., the colours used for the primary and accent colours are different shades of the same colour. It was decided to explore this type of palettes as it gives a more modern look to mobile applications. For the primary colour the teal colour was replaced with a green colour, and on for the accent a darker green was used.

On the side menu bar, was removed the header (Figure 3.16 (a)), which contained the logo and the name of the APP and kept only the menu options. The style of the icons used for each of the tours changed from a cartoon and thin style to a solid design, based on the Google's material design. The icons were also stripped of their colour and became black. The information about the tours on the home page (Figure 3.16 (b)) was also changed to a more vertical representation, with green coloured icons to capture and the visitor's attention. The shape and size of the icons in the tour description screen were kept the same, however the colour was changed to green (Figure 3.16 (c)) in order to keep consistency with the icons in the home screen screen.

In the tour description screen, the style of the start button was changed from a rounded, to a squared type of button, to give the a app a more modern look, and from a lime colour to a darker green colour, to keep consistency with the new adopted colour pallete.

The map also was changed to one based on geographical data about the garden that Professor Paula Redweik provided, which was converted into an *SVG*, using the *QGIS* geographical information system interface, painted. Additionally, this map was added some other elements, such as paths in the oriental garden, and other paths that lead to the green houses, that were not available, based on an *SVG* version of the map provided by Cesár Garcia.

Figure 3.17 compares the different versions of the markers. On the left there's the first version which did not change its appearance when clicked. Next, there's the intermediary version that already changed color when point of interest was visited however it had different colors (grey with white circle in the middle). Also, was adopted a smaller and thinner shape with a white circle for the number (Figure 3.17 (b)). At the right there's the new version with a teal color that changes to a darker shade when point of interest is clicked without a circle for the number. The colour of the number was also changed to white. The icon for the position of the visitor on the map was transparent green with a safari hat colour in the middle Figure 3.17 (b).

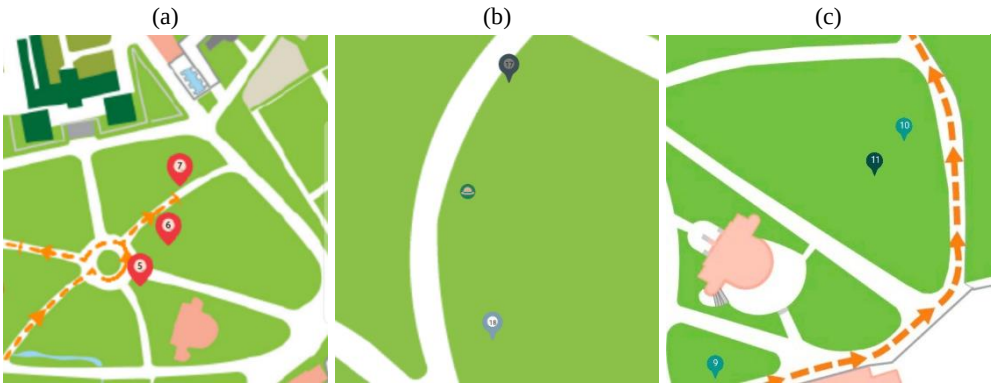

Figure 3.17. Comparison between versions of the location markers. **(a)** first version which did not change its appearance when clicked. **(b)** the second version that already changed color when point of interest was visited however it had different colors (grey with white circle in the middle). **(c)** the third version with a teal color that changes to a darker shade when point of interest is clicked.

The screen of the point of interest description has an *AR* icon, for points that allow *Augmented Reality* experiences, represented by a camera frame with *AR* written in the middle (Figure 3.16 (e)).

The description of the points of interest was split into sections. Additionally, a button to return to the map was added (Figure 3.16 (e)). This button enables the users to go back to the map, and the marker of the visited point of interest is highlighted, by increasing its size, and adopting a darker colour (Figure 3.17 (b)). The *Share* schedule and favourite were removed, as these functionalities were not identified as a priority. However, we plan to add these features in future versions of the mobile application.

In the third prototype the graphical resources for the tour icons, the map makers and the *AR* icon were provided by designer Tiago Ribeiro. These icons had to be adjusted in the mobile application as they were not colour blind friendly (Figure 3.18). As my colleague in charge of conceiving the *AR* experiences was colour blind, the colours for the icons were chosen by presenting him with some versions of the colour palettes, from which he chose the one that let him better see the colours. In

Figure 3.18. Comparison between original palette and new. Top original, bottom new.

The new markers have a colour that matches the dominant colour of the icon, of the tour that is being displayed on the map (Figure 3.19 (c)). Also, when the point of interest has not yet been visited, the marker has a white circle in the middle, and when it has not yet been visited it doesn't.

In Figure 3.19 the new prototype with the new icons for the tours, markers and *AR*, as well as the new navigation is presented. This prototype has changes at the navigational level. The access to the side menu became accessible only to top level screens such as the tour list screen. The rest of the screens have available only the back arrow on the top left corner, which enables users to go back one or two levels depending on the screen they're in. For instance, if the user is on the map screen, the back arrow takes the visitor, back to the tour list screen, jumping over the tour details screen. If the visitor is on the tour description screen it takes him back only one screen to the tour list screen. Also, as the user enters other screens the title on the action bar (navigation base) at the top, is changed to indicate to the user what is being displayed on the screen.

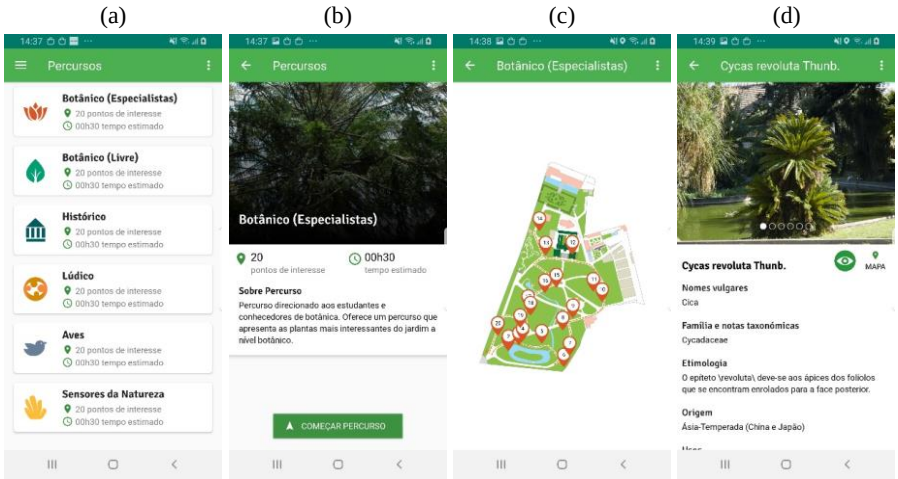

Figure 3.19. Prototype with the new icons provided for the tours, markers and AR.

At this time the architecture of the system was changed to a more complex one with a remote server and database, and new screens were designed. The first screen (Figure 3.20 (a)) represents the download of information from the server, and the second screen the situation when the network or server are not available.

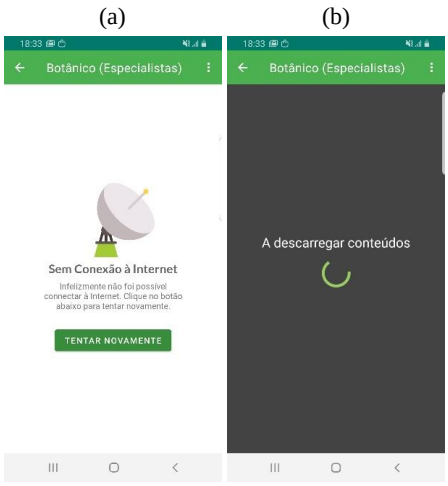

Figure 3.20. Download and retry connection screens respectively.

This prototype was presented at one congress - *Encontro Ciência* at the *Centro de Congressos de Lisboa*. Based on the feedback received from the participants of the congress and improvements made in the Birds and Historic tours, it was conceived the current version of the app. This version already contains three tours.

The Birds tour implied a shift from points of interest to areas of interests. As birds tend to move around and also as multiple birds can be seen in the same place, contrary to the flora, which stays always in the same place and only one specie occupies one place at a time, for each stopping location included in the tour we had to list all the birds species that could be seen.

Another improvement made was in the Historic tour where it was considered subtours, each covering a certain historical period, or type of object. To accommodate subtours the map it was considered the addition of a list of buttons used to filter the points by their subtour. The first version used a Tab Layout. This version was quickly discarded because only one option could be selected at a given time which was perceived as a limitation of interaction with the app.

In the following versions we started using a scrollable *chip* group. A *chip* is a type of button, typically used in the design of *Android* applications, when designing a filtering interaction. The advantage of this approach is that more than one *chip* can be selected. Moreover, it was tested if the icons and the chips should have the same colour, and if the markers should disappear when chip is deselected or became transparent. Figure 3.21 shows the prototypes created for this purpose, with (a) and (b) representing the prototype where markers and chips don’t match in colour and the markers become transparent when filtered, and (c) representing the prototype where the colours of the markers and chips match and the markers become invisible when filtered.

Figure 3.21. Prototypes of the historic tour UI (user interface). a and b, prototype with non-matching colors between chips and markers, with transparent markers when deselecting chip. c prototype with matching colors between chips and markers; markers become invisible when the filter is toggled.

As a result, it was decided to use the solution that matched the colour of the *chips* with the colour of the markers. The markers become invisible when the user deselects the chip associated with a subtour, instead of transparent. Also, it was decided to match the colours of the markers with the colours of the *chips* as it was easier, for the user to find what *chips* wants to deactivate in order to hide points of interest. The implementation of invisible markers instead of transparent, as were in previous version, prevents errors, as the visitor is not tempted to click on markers that shouldn't be there, and reduces the amount of information that the user needs to process in order to make her/his choice.

However, this solution was slightly changed by reducing the height of the *chips*, removing the distinction in *chip* colour between the selected and the deselected states (as the checked icon already provides the user with that information). It was also decided that the colour of the text in all *chips* should be white, independently from whether the colour is lighter or darker (Figure 3.22 (c)). In addition, a very light green was added as background an also a caption for the viewed and not viewed points of interest markers.

Since the download process usually has a long duration, and usually in other applications the circular progress bar is used for fast indeterminate operations, some users thought that the app just stopped responding. For the circular progress bar in indeterminate state, was replaced by a linear progress bar with progress updates (Figure 3.22 (b)). Also, white is used as background colour in the download screen.

Figure 3.23 shows the latest version of the Birds tour. In (a) it can be observed that an AR icon was added to the markers of points of interest that have AR experiences. Next in (b) and (c) the list of points of interest of an area of interest, with a label marking the elements already viewed. Finally in (d) it can be observed that *AR* icons were added in the elements that have individual *AR* experiences.

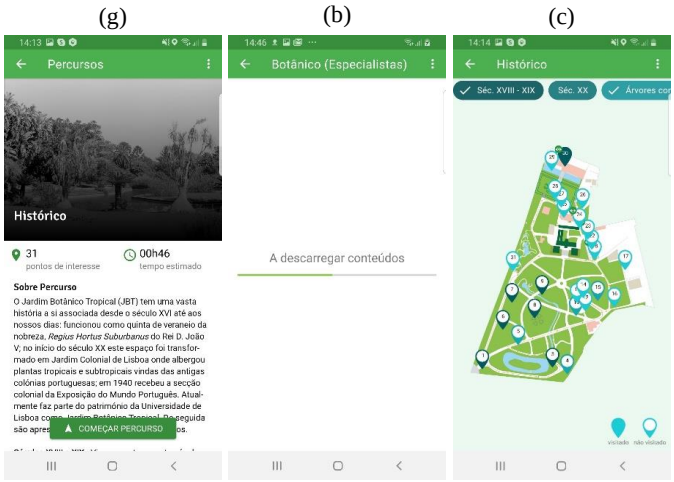

Figure 3.22. Latest version of the Historic tour. (a) description of the Historic tour, (b) content download screen, and (c) map screen with chip group for filtering and caption for visited and not visited markers.

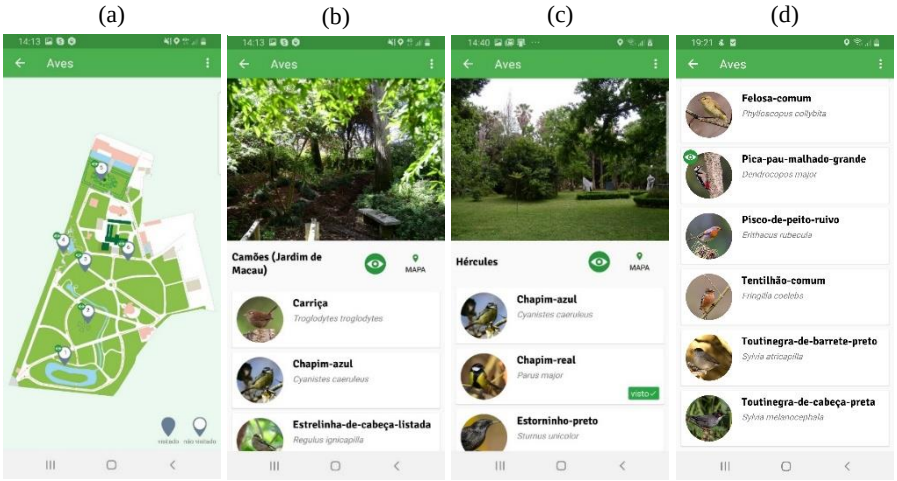

Figure 3.23. Latest version of the Birds tour. (a) map with markers for areas of interest marked with AR icon. (b) and (c), with AR button for experience of the area, also in (c) one of the points of interest is marked as viewed. (d) an example of a point of interest with individual AR experience.

3.8 Web services

In order to query and insert data into the remote database, several web services have been defined. These services are accessed by the mobile application through a REST API materialized through the use of the LUMEN PHP framework. However, before implementing these services first the API needs to be defined. Table 7. Web services for querying and inserting data into the database. presents the description of the web services considered as well as their URLs.

Table 7. Web services for querying and inserting data into the database.

Service Number	HTTP method	URL	Service description
1	GET	/tours	Returns all tours of all tours and contents for describing them
2	GET	/tours/{tourId}	Returns data about tour and its contents
3	GET	/tours/{tourId}/pois	Returns data on all points of interest and their contents
4	GET	/tours/{tourId}/pois/images	Returns images of all points of interest of a tour
5	GET	/tours/{tourId}/pois/descriptions	Returns descriptions of all points of interest of tour
6	GET	/tours/{tourId}/pois/{poiId}/images	Returns all images of a point of interest in of a tour
7	GET	/tours/{tourId}/pois/{poiId}/descriptions	Returns all descriptions of a point of interest of a tour

8	POST	/visitors	Inserts data of visitor
9	POST	/visitors/points	Inserts locations of visitor that make up its trajectory in the garden

Some of these services are directed to storing data related to visitors, others related to retrieving data related to tours, while other are used to retrieve data of points of interests. In Table 7 are first presented the services that retrieve data of tours and points of interest.

Services 1 and 2 retrieve data related to tours. Service 1 lists all the tours and their contents and 2 lists data regarding one tour with *tourId* as its identifier and the contents that describe it.

Services 3, 4, 5, 6 and 7 retrieve data related to points of interest. Service 3 lists all the points of interest of a tour with *tourId* as its identifier and their respective contents. Service 4 shows data regarding the images of all points of interest of a tour identified by *tourId*. Service 5 shows data regarding the descriptions of all the points of interest of a tour identified by *tourId*. Services 6 and 7 show data related to images and descriptions of a point of interest in the context of a tour respectively.

Service 8 and 9 are related to storing data related to visitors. Service 8 creates a new visitor, service 9 stores locations that make up the trajectory of a visitor in the garden. For each visitor is stored its identifier, gender, country, occupation, qualification, birth year and creation date.

3.9 Summary

In this chapter the resources and methods used during the development of this project are presented. Furthermore, are presented a list of requirements to the system and application. Next the architecture of the system and application is presented, followed by the data model for the local and remote databases. In addition, the User Environment Design diagram for the application interaction is also presented, which guided the design of the user interface. Wireframes and early low-fidelity prototypes of the application were also presented, as well as the evolution of the interface user interface until the current version. Finally, it was described the developed services and the REST API considered for the mobile application to communicate with the Application server.

In the next chapter we will present the implementation details for each component in the system, namely the backend, which includes the database and the web application, and the mobile application and its local database.

Chapter 4

Implementation

In this chapter are detailed the conceived features and components in the system. We start by exploring the backend components, namely the database, the web services and finally the public storage.

Following that it is presented the local database and how it is integrated in the mobile application. In subsection 4.2.2 *Content Management*, we present the various content download situations and how the download is processed. After that it is presented how the interactive map was created, how the points of interest and the tour are drawn in it, how the user can navigate it and finally how the points of interest are filtered (in the case of the Historic tour). Next, in subsection 4.2.4 *User Interaction*, other ways the user interacts with the map are explored, such as how its orientation is displayed and are the points of interest marked as viewed. Finally, in subsection 4.2.5 *Visitor Trajectory Management*, it is presented the user locations synchronization process.

4.1 Backend

The backend (server side) of the application was implemented using the LUMEN PHP framework. This framework comes with several tools that help structure and build web applications such as the artisan command line interface, which helps generate files that are part of the framework structure. Additionally, it helps executing scripts to manage components that communicate with the web app, such as databases. The implemented web app serves as an interface for the mobile app to obtain data from the remote database. The remote database consists of a PostgreSQL database extended by PostGIS to support geographical data objects.

This section presents a detailed description of the implemented components that constitute the backend layer of the system. First the processes for installing and managing the remote database are presented, followed by the description of the webservices how these enable the communication with the mobile application. Finally, the public storage component of the backend is presented and how are accessed the resources in it.

4.1.1 Database

To implement the remote database, first we had to install the DBMS (PostgreSQL) – `apt install postgresql-10` – and the PostGIS extension – `apt install postgis`. Next the database and a user must be created and given the user permissions to access that database:

```
CREATE DATABASE yourdbname;
CREATE USER youruser WITH ENCRYPTED PASSWORD 'yourpass';
GRANT ALL PRIVILEGES ON DATABASE yourdbname TO youruser;
```

Furthermore, the PostGIS extension must be created in the database, in order to add geographical capabilities with the command: `CREATE EXTENSION postgis`. With this final step the installation process for the postGIS is finished.

Before proceeding to the implementation of the database structure, the LUMEN application environment must be configured. LUMEN uses the DotEnv library for this purpose, which enables the setup of the environments variables, such as the database connection variables, by placing them in a .env file. As such in this file the database connection, host, port, database name, username must be set, in order to connect the web application to the database. Bellow this setup is illustrated:

```
DB_CONNECTION=pgsql
DB_HOST=192.168.10.10
DB_PORT=5432
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

The LUMEN framework was used. LUMEN to implement the structure of the database. offers a migration functionality in order to structure and keep track of modifications to the database. A migration is a PHP class which implements an up and down method. The up method is executed when a migration is run, and the down method is executed when a migration is rolled back. Additionally, in order to support geographical types the lumen-postgis by *shaozeming* [73] library was added to the framework, enabling us to access and insert geographical types in a safe way.

Each migration name starts with the date when it was created, as these are executed sequentially based on that. More than one table can be created or modified per migration; however, it is a good practice and also promoted by the Laravel team, which also developed the LUMEN framework, that only one table must be modified or created per migration. A migration was created for each table in the data model and the respective attributes and types were defined. An example of a migration is shown below:

```
class CreateAgesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('ages', function (Blueprint $table) {
            $table->string('name',100)->primary();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('ages');
    }
}
```

As the database modified in each iteration of the development, we used the seeding method provided by the Lumen framework, to provide an initial set of data to the database when it is being installed. The Lumen framework provides a simple way to manage this process. For each table on the database a *Seeder* class can be generated, which implements a run method, which is executed when the seeding command is invoked on the artisan command line interface (`php artisan db:seed`). As such on the run method we need to specify the data that we want to insert into the database. In the code bellow a seed class is exemplified.

```
class AgesTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('ages')->insert([
            ['name' => 'Séc. XVIII - XIX'],
            ['name' => 'Séc. XX'],
            ['name' => 'Expo Mundo Português'],
            ['name' => 'Árvores com história'],
        ]);
    }
}
```

In this example the run method the code is executed for inserting data into the *ages* table. By using seeds we have the advantage primarily of being able to execute all the insertions immediately after we create the database with a single command using the *artisan* interface – `php artisan migrate:refresh --seed`. This command reverses all migrations, runs them again and runs the seeding of data into the new schema. This procedure provides a more efficient workflow for modifying the database.

4.1.2 Web services

To implement the webservices it was used the routes and controller classes, of the Lumen structure. The route class defines the available URLs for requests and the actions executed. Routes also enable to specify the format for parts of the URL in order to avoid URL manipulation attacks. The controller class handles all the logic in the Web application and is responsible to query or insert data into the database based on the user request.

GET requests return JSON objects, and POST requests imply the validation of the Json objects attached to these requests. All the web services presented in chapter 3 section 3.8 were split through three controllers – ToursController, PointsOfInterestController, VisitorController – each handling requests that are related to a given concept. The implementation of the /tours, /tours/{tourId}/pois, /visitors and /visitors/points routes will be further detailed.

The GET /tours route, retrieves data about all the featured tours and their respective contents, is handled by the ToursController, and invokes the *getAllTours* method in it. As a response to this request, the corresponding web service returns a JSON array of objects containing data about the tours and the contents (images and icons) belonging to the tour. Each element in this array follows the format specified bellow:

```
{
    "id":1,
    "name":"Botânico (Especialistas)",
    "description":"Percurso direcionado aos estudantes e conhecedores (...)",
    "duration":"00:30:00",
    "number_points_of_interest":20,
    "label":"botanic_specialists",
    "configuration":0,
    "images": [
        {
            "content":9,
            "base_name":"araucaria_bidwillii_2.jpg"
        }
        ...
    ],
    "icons":[
        {
            "content":261,
            "base_name":"ic_botanic_specialists.png"
        }
        ...
    ]
}
```


The initial part consists on all the attributes of the tour table, while the last two elements are arrays of data related to images and icons respectively. Images and icons are represented as json objects with keys corresponding to the attributes of their respective tables, i.e., *content* and *basename*.

The GET tours/{id}/pois route, which retrieves data about the points of interest that constitute a given tour identified by *id* and their respective contents, is handled by the PointsOfInterestController, and invokes the getAllPoisOfTour method from it. The response is JSON array of data related to points of interest that belong to a given tour, as well as data concerning all the contents associated with the points of interest. Each entry in the array follows the following pattern:


```
{
    "id": 1,
    "longitude":"-9.20337576482106",
    "latitude":"38.6982848770672",
    "name":"<i>Beaucarnea recurvata<\/i> Lem.",
    "type":"Planta",
    "subtype":"árvore",
    "average_stopping_time":"01:30:00",
    "age": null,
    "order":1,
    "images": [
        {
            "content":12,
            "base_name":"beaucarnea_recurvata_1.jpg"
        },
        ...
    ],
    "description":88,
    "topics": [
```

```
            {
                "order":1,
                "topic":"Nomes vulgares",
                "base":"Baucárnea-de-folhas-recurvadas",
                "see_more":null
            },
            ...
        ]
    }
```

The JSON object starts with the information on the point of interest: *id*; *latitude*; *longitude*; *name*; *type*; *subtype*; *average_stopping_time* and *age*. The *order* represents the sequence number of the point of interest in the tour, and after that we have data related to contents associated with the point of interest. First, we have an array of image data, presented by two attributes: *content* and *base_name*. Following the array of images is the unique identifier of the point of interest's description. Finally, at the end of the object there's an array of sections of the description described by *order*, *topic*, *base*, *see_more*. The *order* is used to organize the sections, and *base* and *see_more* are used to indicate which part of the text should be visible and which should be collapsed when the number of characters in the section exceeds the threshold.

Lastly both the POST /visitors, which creates a new visitor, and the POST /visitors/points, which stores data related to user locations, routes are handled by the VisitorController. Contrary to the previous routes incoming requests contain JSON objects that need to be validated. To validate the data provided, the validator provided by the LUMEN framework was used, that receives as parameters the data that needs to be validated, and a relational array, with keys corresponding to attributes in the data (e.g. created_at), and strings containing rules these attributes must follow as values (e.g. required|date|regex:<format of the date>). If no problems with the validation are found, the data is inserted into the database, and a 200 OK is sent back, otherwise, the data is not inserted and there's no return.

```
{
    "id" : "PTIZKANSMON1569432491327",
    "gender": null,
    "country" : null,
    "occupation" : null,
    "qualification" : null,
    "birth_year" : 2019,
    "created_at" : "2019-09-28T10:31:11.477Z"
}
```

In the above example is presented the format of the JSON object attached to POST requests to the /visitors route. The object contains the keys: *id*, *gender*, *country*, *occupation*, *birth_year*, *created_at*. It can be observed that these keys are equivalent to the columns of the Visitor entity in the data model. At at the moment the screen for adding information about the gender, country, occupation and qualification of the visitor is not yet implemented, validation rules for these fields were not created, and only the id and created_at columns are set and validated, for each row in the *Visitor's* table on the remote server.

4.1.3 Public storage

The LUMEN framework contains in its structure a public directory and contains index.php, the entry point to the web application. This directory also includes all the files and directories that the users may access with no authentication or identification necessary. In the case of the implemented system this directory was used to hold the directories with the contents of the application: icons, images, audio and video. The files contained in this directory are not accessed through web services, instead it is used the relative path from the public directory. For instance, to access the image with the name motacilla_alba_1.png from the web application, which is accessible at https://jbt.ulisboa.pt/, a GET request is sent to https://jbt.ulisboa.pt/images/motacilla_alba_1.png.

4.2 Mobile application

In this section are detailed the various components and mechanisms that constitute the mobile application. First is presented the architecture adopted for the mobile application. Next, are detailed the components used to download contents in the app. After that the creation and features of the interactive map used in the application are described. Furthermore, the orientation and progress tracking components are presented. Finally, the visitor positions synchronization component is described.

4.2.1 Application architecture

Figure 4.1 shows the architecture of the application as a result of the materialization of the architecture presented in section 3.4. This architecture is based on the MVVM (Model-View-ViewModel) pattern which is recommended by Google.

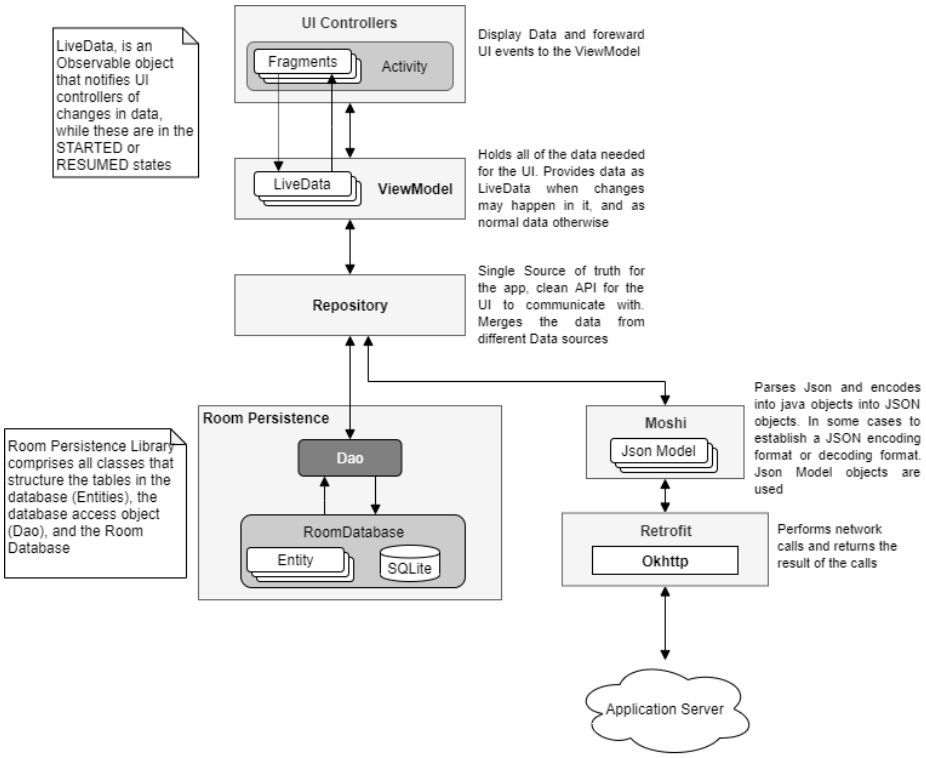

Figure 4.1. Implemented application architecture, using the Model-View-Viewmodel pattern.

The MVVM is an architectural pattern for applications that contain a data source and a user interface. This pattern splits the software into three components: the Model, which holds all

components responsible for handling the data on an app; the View, which comprises all components that handle the display of data, and listen to user events such as touch and gestures; the ViewModel, which interprets the user inputs and updates the model and provides data streams to the view, which subscribes to them. Additionally, a repository class is added as part of the architecture, which is used to further abstract the data access by combining data from different data sources: the local database and the remote database.

In the Android architecture, the role of the Views is interpreted either by an Activity or a Fragment, which are groups of views that constitute the layout of the screen. The Activity represents an entry point to a mobile application, and it is the class that creates the window in which the UI is drawn. The Fragment represents a behaviour, or a portion of the user interface and can be seen as part of an Activity. The Fragment enables the reutilization of UI components across multiple activities and enables the removal and addition of compartments in the Activity that work independently from each-other. When implementing an android application, the View layer can be structured as a set of Activities, each activity for a screen of the application; one activity with multiple fragments, each Fragment for a different screen; and finally, a combination of the two.

We used the approach of one activity with multiple fragments, as Fragments load faster than activities, and offer more control over their lifecycle. Fragments do not extend Context, which is the base class for accessing global information about the applications environment, such as where is the storage folder dedicated to the app, and the different API's provided by the operating system, such as the location API. All this code is then delegated to the Activity that holds the fragments. The fragments execute then most of the work related to displaying data on the screen.

The *ViewModel* layer is materialized by the *ViewModel* implementation provided in the Android Jetpack, which is a set of libraries and components provided by the Android development team to simplify the development process [74]. This implementation is lifecycle aware, and retains all data referenced in it and cancels all operations on configuration changes. Lifecycle aware is an important feature of Android for components that depend on Activities and Fragments, as these have lifecycles. A lifecycle is a set of states that a component takes from its initialization until its resources are freed by the system. The part where the *ViewModel* is important is when the user rotates the screen. At this point, if the activity is not locked in a given orientation, then it is destroyed and reconstructed again. The *ViewModel* saves its state, and when the Activity is recreated again, the state is restored automatically. Additionally, the *ViewModel* handles the communication with the model. It interprets events captured by the View and transmitted to it and gets the data or updates the data source needed in order to update the View with the necessary changes.

Next the repository component abstracts the data access, by combining data from the different data sources. To access data from the remote database the *Retrofit* library is used to send requests to the application server and receive data. *Retrofit* is built on top of *Okhttp* a HTTP client library, and the *Moshi* library is used to parse received JSON objects. To parse the JSON objects *Moshi* receives a set of *JsonModels* which it uses to convert JSON into Java objects.

To access the local database, the repository uses a Dao (Data Access Object). Dao's are components that are part of the Room Persistence Library, which provides an abstraction layer over the SQLite to allow for more robust database access. The Dao object is annotated with *@Dao* and holds all methods for accessing data from the database (*insert*, *update*, *delete*, *select*). The Dao can be an interface or an abstract class. In both cases abstract methods should be annotated with the operation that the method should do (*@Insert*, *@Update*, *@Delete* or *@Query*). The *Insert*, *Update* and *Delete* operations are handled directly by Room, and the methods should receive an object as parameter. For queries and *Insert*, *Update* or *Delete* operations that require more details, it is used the *@Query* annotation. The return type for the methods can be a single object, a list of objects or LiveData, which consists on a stream of data that Views can subscribe to.

Another component of the *Room* library is the *RoomDatabase* which holds the database connection. In this class it is defined the name of the database, described what the database should do once it is first created or each time the connection is established, and registered what tables should be created in the database.

Finally, the database is structured using entities. To define entities, classes from the domain model (e.g. Visitor) are annotated with *@Entity* to mark the class as an entity, and the primary keys are annotated with *@Primary*. Foreign keys and complex primary keys are passed as values for the arguments *primaryKeys* and *foreignKeys* in the *@Entity* annotation.

4.2.2 Content management

One of requirements of the developed application is that it should work offline. In order to fulfil this requirement, the download of contents from the remote servers had to be implemented.

There are two scenarios when the contents are downloaded: when the user starts the app and when the user chooses a specific tour for the first time. In the first case contents such as text, icons and images used to display information about the tours are downloaded as the splash screen of the app is visible. In the second case contents used to display points of interest on the map as well as to describe them are downloaded.

The first thing that was needed to decide was where to place the downloaded contents. In *Android* the storage can be divided into two areas: *Internal* storage and *External* storage. This nomenclature is misleading, as it does not match the context. These names are normally used to differentiate between the storage that is soldered onto the motherboard, and the one that can be plugged in and out such as micro-SD-cards. The *Internal* storage is the storage that is accessible exclusively by one application, and which is erased when the application that uses it is uninstalled. On the other-hand the *External* storage, is world-available (i.e., all applications can manage it). The *External* storage can then be a folder inside the storage that is soldered onto the motherboard or in an *SDK* card. When the application is removed however, only the files that belong to the folder designated for the application are removed. If contents are stored in folders outside the designated directory, such as in images, the contents will not be erased. Hence, an *External* storage should be used when files do not require access restrictions, we want to share them with other applications, or if we want the images to be accessible from the computer. the *Internal* storage should be used when we want to be sure that neither the user nor other applications use the files of application.

It was decided to use the *Internal* storage as it gave us a greater control over the storage. Also, as all images and audio contents are protected under copyright law, it was necessary to keep these contents in a storage where it is possible to implement a control those images.

Components provided by the Android API for downloading contents such as the *DownloadManager*, which dispatches the download operation to the operating system cannot access the *Internal* storage of the application, thus it was necessary to implement our own download framework.

Download at the beginning of the application

At the start of the application a splash screen, i.e., a screen with the application’s name and logo, is shown to the user. While this screen is visible a download process occurs, which aims to download all contents, such as images, icons and descriptions used to present the featured tours in the application. This download process is illustrated in Figure 4.2.

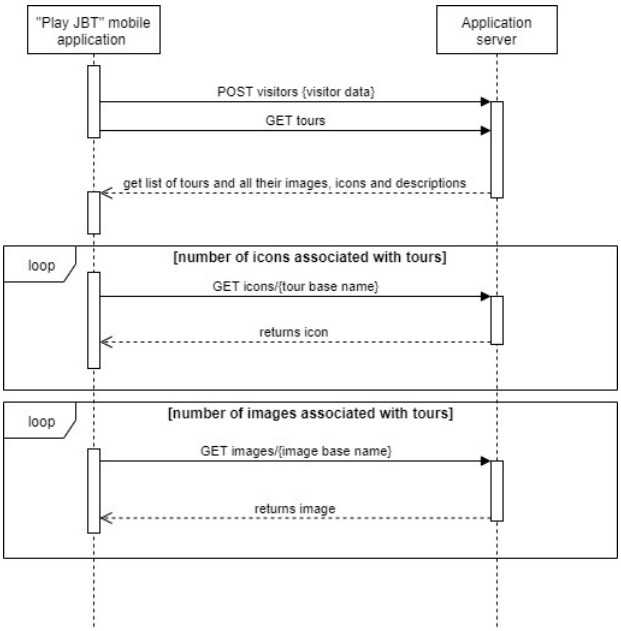

Figure 4.2. First application kickoff content download. At this stage only contents (icons, images and text descriptions) used to present the tours.

First the visitor is created in the remote database, and data related to the tours and contents used to describe them such as text, images and icons are requested. Next all the icons are downloaded followed by all the images. The download of these contents is executed in this exact order; however, both the download of the icons and images is done through concurrent requests to the application server.

Tour download

When the user chooses a tour for the first time, data related to points of interest included in it and contents used to describe them are downloaded. This downloading process occurs while the *DownloadFragment* is still visible to the user (i.e., until it enters the *onStop* event in its lifecycle). This means that the visitor can cancel the download at any time simply by pressing back on the *Android* system controls.

At the beginning we implemented the download process as part of the *ViewModel*. However, this blocked the *ViewModel* while writing operations that needed to finish were still executing. Additionally, this added one more concern to the *ViewModel* that did not belong to it. Considering these, all the code was moved from the *ViewModel* to the *DownloadFragment*.

However, this was still a bad design as a *ViewController*, such as a Fragment or Activity, should not be concerned with the download of contents. Thus, a separate component was implemented, the *FileDownloadManager*, that contains all the download logic, and sends progress updates to the *DownloadFragment* which displays them. The *FileDownloadManager* is aware of the local database, the *HTTP* client and the local storage, as it needs these components to do his work and the DownloadFragment knows the *FileDownloadManager*.

Figure 4.3 shows the first implementation of the download process which was sequential. First data describing the points of interests and contents associated with them was retrieved, and after that the contents were retrieved from the database one by one. This approach was very slow, taking on average 1 minute and 30 seconds to download all the contents in the Botanic tour, which was not acceptable. This implementation was also very memory consuming, as contents were first loaded into memory until they were fully downloaded and then written into memory, which is related to the *Retrofit* libraries implementation of a synchronous call (*execute*). However, this approach poses the

advantage of being easier to cancel, as to do so, we only need to check if interrupt has been issued on the thread and cancel the current download operation and loop.

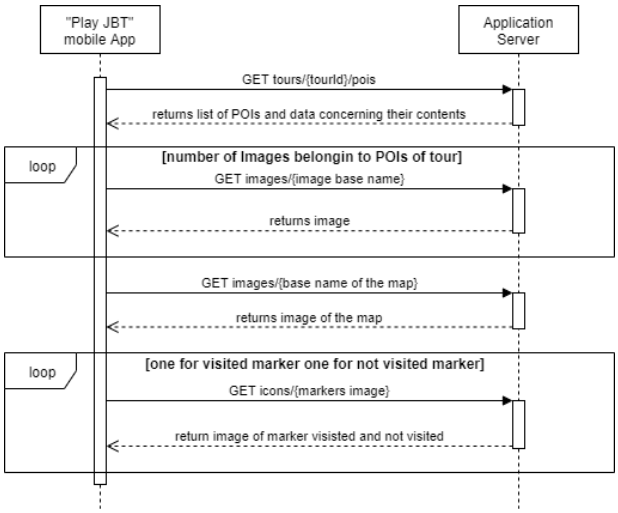

Figure 4.3. Sequential implementation of tour content download. In this implementation, contents used to present the tour's points of interests, are downloaded one after another with the next content being downloaded only when the previous content has been downloaded.

Figure 4.4 shows the second and current implementation of the download process which uses asynchronous calls and the streaming behaviour provided by the *Retrofit* library. This new approach significantly speeds up the download process, taking an average of about 10 seconds for the same tour previously mentioned. The new approach is also less memory consuming as parts of the content are written to storage directly as they come. This approach however uses more processing power, as it uses multiple threads. In order to limit the impact on the battery, it was imposed a limit of five concurrent calls. The cancelling of the work is also more complex, as references to all the calls must be stored and iterated through, while calling cancel on them.

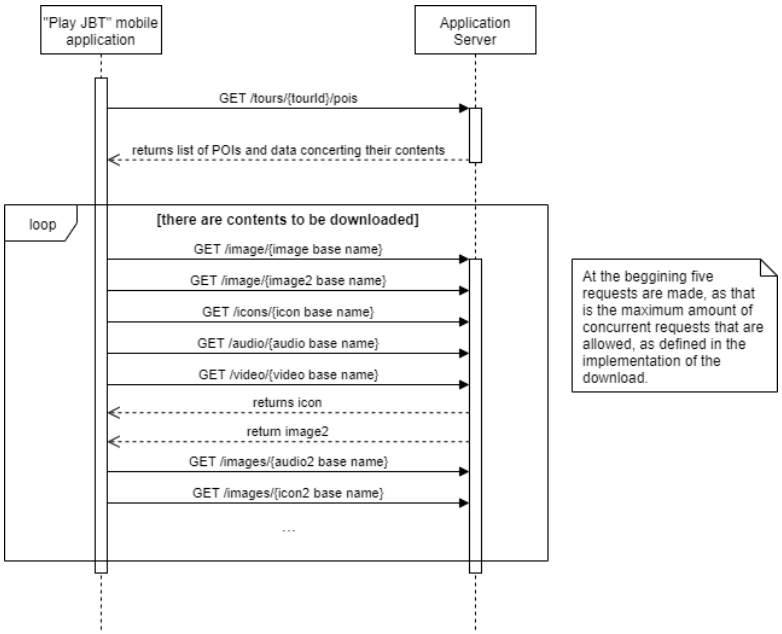

Figure 4.4. Concurrent implementation of tour content download. In this implementation of the download of contents used to present the tour's points of interest, the contents are downloaded independently from each other, with only five simultaneous download operations being allowed.

4.2.3 Map interaction

In this subsection we start to explain the process of the creation of the JBT map and explore different features of it such as how the user navigates it, how geographical positions are drawn in it and how the points of interest are filtered in the case of the Historic tour.

Building the map

The map that is used in the mobile app is based on high accuracy data on the different shapes that make up the garden, provided by Professor Paula Redweik as a shapefile. This data can be seen on the top left corner of Figure 4.5 and it can be observed that aside from the shape and walk paths in the garden, are additionally displayed trees, elevation, and the buildings outside of the garden which aren't relevant to the representation of the garden and would overwhelm the user. As such this representation was exported to SVG, without changing the orientation in the geographical space, so it could be edited using graphical editors such as GIMP, and all the irrelevant details of the map were removed.

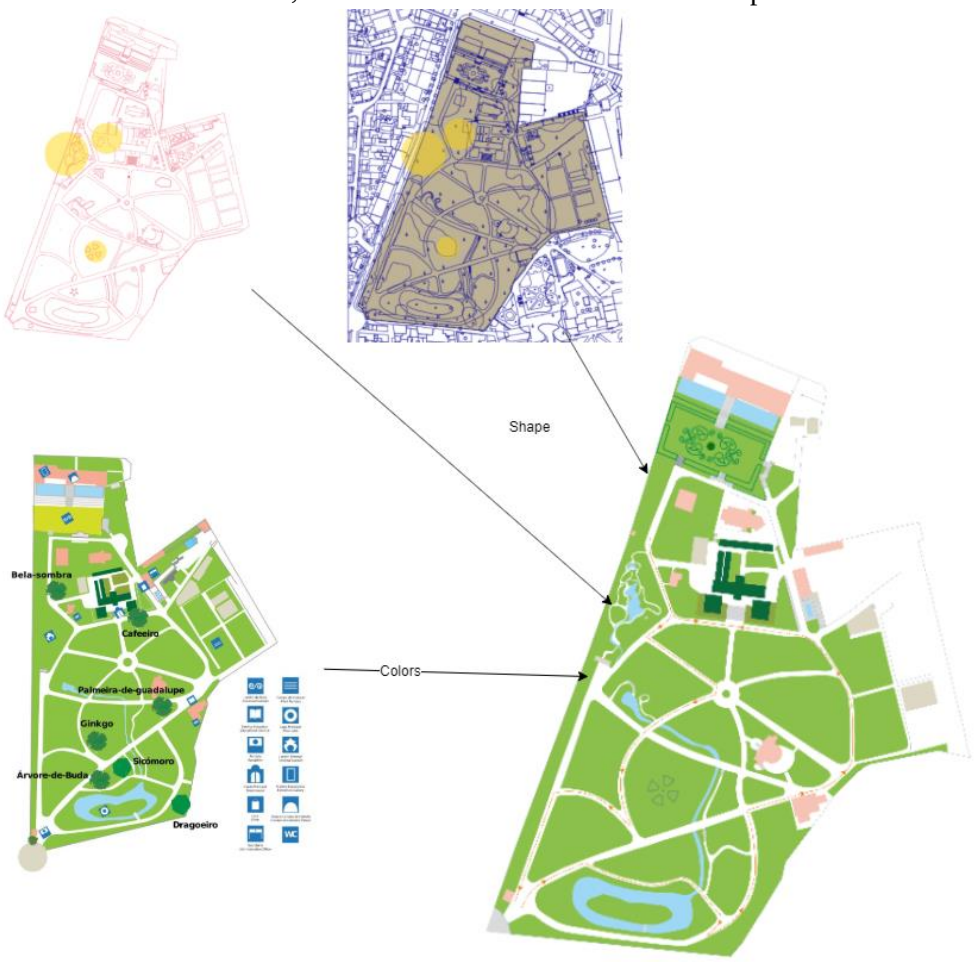

Figure 4.5. Summary of the creation of the map. On the Bottom left we have the original map of the garden, used to extract the colors. On the top we have the map with data offered by Professor César Garcia, with the areas that differed from the map on the right (with data provided by Paula Redweik) in yellow. These areas were merged with the data shape of the map available on the right, as a representation of the data provided by Professor Paula Redweik.

The resulting simplified version of the data was then colored using the color palette from the map available at the entrance of the garden (Figure 4.5 bottom left). As some areas of the original map suffered changes during the rehabilitation of the garden, some areas, paths and buildings were not included in the current version of the map. In order to incorporate these missing data, it was used a map provided by Professor César Garcia (Figure 4.5 top left corner).

All these changes resulted in the map used currently for displaying tours in the map (Figure 4.5 bottom right corner). To be better integrated on android the resulting image was exported to PNG. Although SVG is supported on Android starting from API level 21, which is supported by our app, there are not any libraries that support zooming, panning and overlaying of objects onto an SVG. For this reason, the PNG format was used. Also, the PNG instead of the JBT format was used as JPG when zoomed, starts to be show compression artifacts (noticeable distortion of the media).

Navigating in the map

The implemented solution for map navigation, enables the user to view his/her position in the map, as well as the positions of the points of interest included in the tour. Also, the map has drawn in it a recommended path to be followed in the context of a specific tour. Additionally, the positions marked on the garden map, provide a recommended viewing order. When the user taps some position marked on the map, this position is marked as visited. This change is transmitted to the user by changing the marker's (icon marking the position on the map) from having a white circle around the order number, and black text to a marker without the white circle and white text for its order number.

In the case of the Botanic ("Botânico") and Historic ("Histórico") tours, there's a one-to-one correspondence between the position marked on the garden and a specific point of interest. However in the Birds tour ("Aves"), the positions represent areas and have a one-to-many relationship with the points of interests, i.e., in a single marked position we may see various attractions, instead of only one. Another special case is the Historical tour, where in order to enable the visitor to view points of interest that belong to certain thematics or time periods, we provide a filtering mechanism where the visitor can filter points of interest by the time period they belong to.

To display the map and to mark positions on it a subsampling-scale-imageView is used. To track the position of the visitor the phone's GPS is used, and the gyroscope, accelerometer and magnetic field sensors are used to calculate the user orientation. Each time there's an update from the sensors or the GPS, or the user zooms and pans the map, the view holding is refreshed (invalidated) in order to apply these changes.

Drawing geographical positions on the map

To provide an interactive navigation system the map was integrated into the application and for each tour featured in the application the map has drawn the recommended track to be followed by the user. The map is loaded by the library *subsampling-scale-image-view* typically used in image galleries or to display large images that can't be loaded directly into memory as this would result in out of memory errors. The library also supports zoom, pan and rotation operation, and is very extendable, enabling overlays and tour detection listeners. For images that are large (with pixel dimensions larger than 2000px per 2000px), this library uses the tiling process by which a lower resolution image is loaded into memory, and as we zoom higher resolution parts of the image are loaded into memory.

Nonetheless, to display the points of interest in the map and to display the position of the visitor some calculations had to be made. The library provides a method (sourceToView) that converts any given coordinates of the full-sized image of the map, into coordinates of the viewport. This method is very convenient as we only need to calculate the formula for converting geographical coordinates into coordinates of the full-sized image. An additional convenience is that since the area of the garden is relatively small, we don't have to consider the curvature of the earth, as such we can apply a simple 2D transformation to convert geographical coordinates into coordinates of the map.

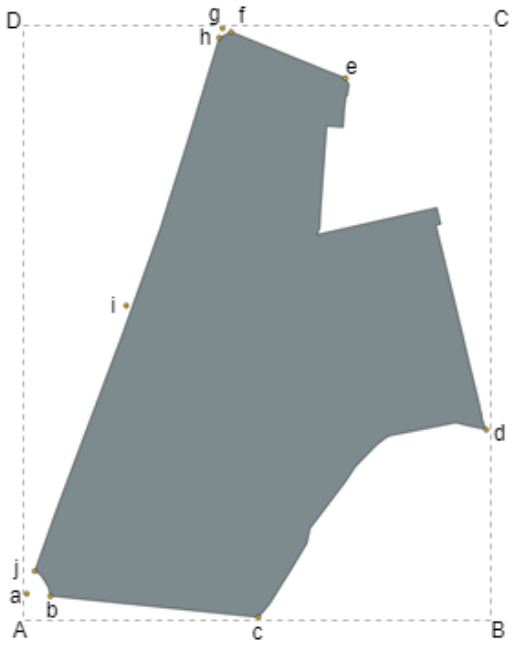

Figure 4.6. Shape of the garden and other geometric elements used to calculate the 2D transformation that converts geographical coordinates into map image coordinates. Points *a*, *c*, *d* and *g* are used to calculate the coordinates of the vertices of square ABCD used to simplify the calculation. Points *b*, *c*, *h* and *j* were used to calculate point a. Points *e*, *f*, *h* and *j* were used to calculate the coordinates of point g.,

Figure 4.6 shows the shape of the garden and other geometric elements that were used to calculate the 2D transformation that converts any geographical coordinate in coordinates of the map image. Points *a*, *c*, *d* and *g* were used to calculate the coordinates of the vertices of square [ABCD] used to simplify the calculation of the transformation. The QGIS geographical information system was used to calculate the coordinates of some of the vertices in the shape of the garden (*b*, *c*, *d*, *e*, *f*, *h* and *j*). Points *a* and *g* were derived from calculating the intersection of vectors made of the points whose coordinates were calculated by the QGIS system. Point *a* was obtained by calculating the vectors $\overrightarrow{bc}$ and $\overrightarrow{hj}$, then use the resulting vectors to calculate equations of the lines $\overline{bc}$ and $\overline{hj}$ respectively. Finally, the coordinates of *a* were found by calculating the intersection point between lines $\overline{bc}$ and $\overline{hj}$. The same process was used to calculate the coordinates of *g*, however the intersection of $\overline{fe}$ and $\overline{hj}$ was calculated instead.

Vertex *a* has the minimum longitude, *c* has the minimum latitude, *d* has the maximum longitude and *g* has the maximum latitude. The reference square is made of vertex *A* with the coordinates ($longitude_a$, $latitude_c$), another *B* with coordinates ($longitude_d$, $latitude_c$), one C with coordinates ($longitude_d$, $latitude_g$) and finally one D with ($longitude_a$, $latitude_g$). In the calculations we use the *A* point as reference in the transformation. The length of the square is equal to the length of the vector $\overrightarrow{AB}$, and the height is the length of vector $\overrightarrow{BC}$.

$$P' = P * translation\ (-longitude_A, latitude_A) * Scaling\ (1, -1) * translation\left(0, length\ of\ vector\ \overrightarrow{BC}\right) * scaling\left(\frac{length\ of\ \overrightarrow{AB}}{length\ in\ image\ \overrightarrow{AB}\ in\ pixels}, \frac{length\ of\ \overrightarrow{BC}}{height\ in\ image\ \overrightarrow{BC}\ in\ pixels}\right) * translation(padding\ horizontal, padding\ vertical) \quad \textbf{(1)}$$

Equation **(1)** consists on the complete transformation that converts any point P with geographical coordinates into a point P' in the image of the map. This transformation can be split into five steps The first step is to place point A in the origin, i.e., $translation(-longitude_A, -latitude_A)$ **(2)**. Then as the height in values for the ordinates grow in an inverse order, as compared with the real world, i.e., as we

go up in the viewport, the value for the ordinate becomes smaller, a symmetry needs to be applied in order to invert the map. As such the second step is $scaling(1,-1)$ **(4)** and after that $translation\ (0, length\ of\ vector\ \overrightarrow{BC})$ **(5)** to put it back over the x axis. Next the rectangle [ABCD] is scaled in order to match its real height and length with the length and height in the image:

$$scaling\left(\frac{length\ of\ \overrightarrow{AB}}{length\ in\ image\ of\ \ \overrightarrow{AB}\ in\ pixels}, \frac{length\ of\ \overrightarrow{BC}}{height\ in\ the\ image\ of\ \overrightarrow{BC}\ in\ pixels}\right) \quad \textbf{(6)}$$

The scaling factor for the length and height of square [ABCD] are defined in relation to the area it occupies in the source map image instead of the size of the map, since the map image contains padding which is used to improve the presentation of the map, and to enable zooming and panning close to the extremities of the map.

Finally, a translation is applied to position square [ABCD] correctly in the image $translation(horizontal\ padding, vertical\ padding)$ **(7)**. The zooming and panning the image, and the necessary operations in order to transform a position in the source image of the map to a position in the viewport are all handled by the *subsampling-Scale-ImageView* library.

Point of interest Filtering

In the Historic tour, the user has the option to filter the points of interest based on their historical period or sub-tours: the XVIII-XIX century tour (*"Séculos XVIII-XIX"*); the XX century tour (*"Século XX"*); the Portuguese World Exposition of 1940 tour ("Secção colonial da Exposição do Mundo Português"); and finally the Trees with history tour ("Árvores com História").

The filters are represented as a *chip* group that is scrollable (see Figure 3.22). A *chip* is an oblong shaped button, typically used in material design (user interface design language used for *Android* applications) when designing a filtering interaction, that when pressed is added a check symbol on the left. On the background of this interaction the ages that are to be filtered are added to a list of strings representing the age. When the map refreshes, the *age* attribute of the points of interest marked on the map are checked against the list of filters. If the *age* is in the list of filters, then the point of interest is not drawn onto the map.

4.2.4 User Interaction

Obtaining visitor orientation

When navigating in the map the visitor can see its position and the position of points of interest of the tour in the map. Additionally, he/she can also see its orientation through the rotation of the icon that marks its position. In Figure 4.7, the blue circle with white outline marks the visitor's position, and the arrow centered in it points the way the user is looking.

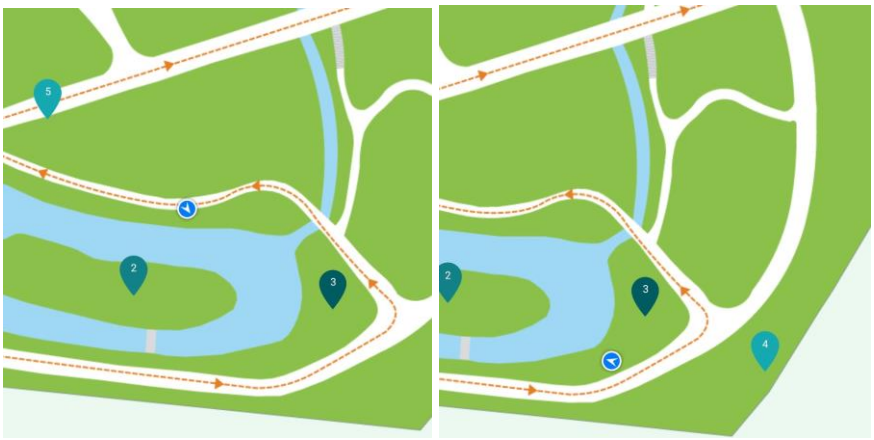

Figure 4.7. Map with the visitor position and orientation marker. The blue circle with white outline marks the visitor position, and the white arrow centered in it points to the way the user is looking to.

Normally the strategy used to get the orientation in Android is to use invoke the method `SensorManager.getOrientation()` that combines data from the accelerometer and magnetic field sensor. This data could have many variations and be inaccurate, due to electromagnetic interference from other magnetic fields. The gyroscope data is more accurate; however, it suffers from what is called the *drift*. The drift comes as a result of the integration of the angular velocity with the aim of obtaining the actual orientation. To calculate the orientation the angular velocity in all axis is multiplied by the difference between the time of the last and current sensor output. The sum of all rotation increments results in the actual rotation of the device. During this process small errors are introduced that add up overtime resulting in a constant slowdown in rotation.

In Figure 4.8 how data from the accelerometer, magnetic field and gyroscope sensors is combined in order to overcome the shortcomings of using these sensors separately, based on a tutorial by Paul Lawitski [75]. In this implementation the gyroscope data is used for changes in short time periods while the accelerometer/magnetic field data is used for changes over long periods of time. This is accomplished by applying a low pass filter to the orientation obtained from the accelerometer and combining it with the high pass filtered data from the gyroscope. The process of combining the low-pass filtered data from the accelerometer and magnetic field sensors with the high-pass filtered data from the gyroscope is what is called a complementary filter.

In signal processing a signal is a function that conveys information about the phenomenon, which in this case is the orientation. A low-pass filter is a process which removes high frequency components from a signal and the high-pass filter removes the lower frequency components. The low pass filtering of the accelerometer and magnetic field orientation and the high pass filtering of the gyroscope are accomplished by combining their orientation values every 30 milliseconds: $FusedOrientation = 0.98 * GyroscopeOrientation + 0.02 * AccelerometerMagneticFieldOrientation$. The factors 0.98 and 0.02 which are multiplied by the *GyroscopeOrientation* and *AccelerometerMagneticFieldOrientation* respectively, are time-constants that define how much these orientation values should influence the absolute orientation (*FusedOrientation*). The values chosen are based on the ones used in Paul Lawitski's tutorial, as these were tested to work well for 30 milliseconds sampling rate.

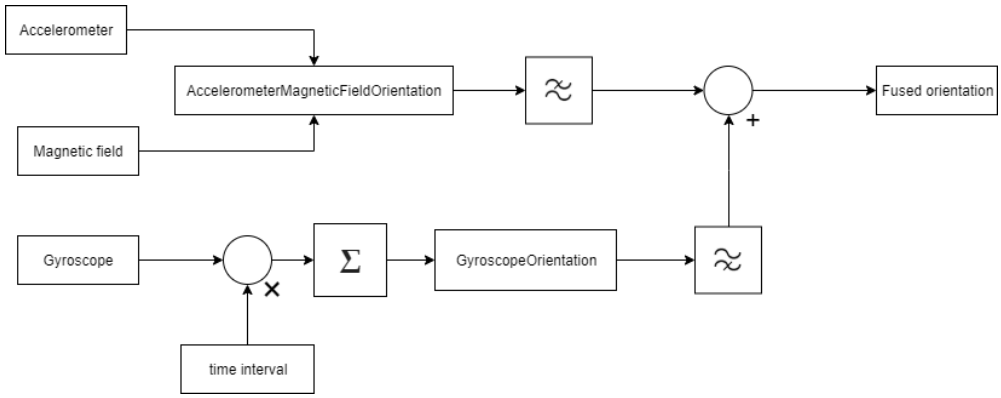

Figure 4.8. Combination of data from accelerometer, magnetometer and gyroscope, inspired in diagram presented by Paul Lawitski [75]. The combined data from the accelerometer and magnetic field sensors, is applied a low pass filter as indicated by the symbol in the square, while the integrated rotation data from the gyroscope is applied a high pass filter. The filtered data from theses sensors is integrated together overtime through the formula at the top of the leftmost circle.

With the filtered data from the sensors, the values attributed to the azimuth (x axis value) are used to update the orientation of the icon that marks the location of the visitor. This is accomplished by rotating the bitmap of the icon around itself the amount of degrees obtained from the fused orientation. The orientation component was tested in the garden and the results were considered satisfactory.

Marking viewed points

In order to mark points of interest as viewed the VisitorTourPointOfInterest table is checked in the local database. If there is a row with a given tour id and tour point of interest and the viewCount is greater than 0 then we mark the point as viewed. Every time the visitor taps on a marked position on the map, it is incremented the view count of the point of interest that is associated with that position. As to visually transmit to the user what points of interest are already visited, the icons that mark the position of a point of interest switch from having a white circle around the number of the position, to an icon without a circle and with white text for the number.

In addition to marking the viewed points of interest, the last viewed point of interest is highlighted. This is achieved by keeping a reference to identifier of the last viewed point of interest. When the map is refreshed, which in this case happens at the same time the visitor taps on the icon of a given position, the icon that was most recently tapped increases its size.

Special case: Areas of observation

In the Birds tour each position marked on the map represents and observation area of several birds. As such instead of having each position correspond to a single point of interest, each position corresponds to a list of points of interest

To implement this behavior a generic approach was used to represent both a one to one correspondence and a one to multiple correspondence between positions on the map and points of interest. In this approach while iterating through the points of interest to mark them on the map, only the first occurrence a given position is drawn, and all immediate consecutive points of interest with the same position are jumped. In order to be able to do this an additional point of interest, which serves as a reference to an observation area, is added to the database that has a lower order value than all the points of interest that should be listed for that area, guaranteeing that it is iterated first. Additionally, it is stored the index of the first and last point of interests that are part of the observation area.

When the visitor taps on a marker it is checked if there's a one-to-one correspondence between the positions and points of interest, by checking if the different between the indexes of the first and last point of interest that constitute equal to one. If the difference between the indexes of the first and

last points of interest is greater than one, the points of interest that belong to the area are listed, and the user can choose which point to view (see Figure 3.23 (b)), otherwise the visitor is taken directly to the details of the point of interest that is represented in that position.

Another particularity of this tour is that markers for position are marked as visited by setting the incrementing the viewCount of the VisitorTourPointOfInterest entry for the point of interest that serves as a reference for the observation area. The points of interest included in the observation area are also marked, however this is done in the list view where they are displayed. All points of interests that have already been viewed are added a green square with a check icon and the text "viewed" ("visto"), at their bottom right corner (see Figure 3.23 (c)).

4.2.5 Visitor trajectory management

When the user starts visiting a specific tour all the GPS positions are recorded and synchronized with the server. This synchronization happens in two scenarios: when the map Fragment (where the visitor position is presented, and which is responsible for triggering the storing of data) enters the *Stop* state or after five minutes of continuous interaction with the map. A Fragment enters the *Stop* state when the phone's screen is shutdown, or when it or the Activity it is attached to is replaced by another (e.g. the point of interest details Fragment).

To synchronize the data, it is used the SyncAdapter framework, provided in the Android API. The sync adapter framework offers many advantages such network verification, synchronizing the data only when network is available, and improved battery performance, as all the synchronization is done in one place at the same time. Additionally, this component offers the structure for easily implementing time scheduled synchronization with the server, as well as server issued synchronization. If the phone has low battery the synchronization, the time scheduled synchronization is not executed after the specified time period, instead it is delayed a few seconds. Finally, the synchronization is executed in a background service, which means that it is executed even if the app is closed, and also is done sequentially, i.e., no more than one synchronization operation can be executed at a given time. If a synchronization is triggered while another is still taking place, then it is added to the work queue of the background service and is executed when the currently executing synchronization finishes.

The SyncAdapter framework is used in conjunction with the Retrofit library, which is used to implement a REST client, for executing network calls. As such when a synchronization is scheduled the sync adapter checks for the network, while retrofit executes all the HTTP operations needed in order to synchronize the data.

Figure 4.9 shows a diagram representing the two possible states for the app: collecting data and synchronizing data. These states are processed simultaneously. In the collecting data state, every two seconds the app stores the visitor's positions until the map Fragment enters the *Stop* state, at which point additionally a synchronization task is added to the SyncAdapter's queue. If the map Fragment does not enter the Stop state 5 minutes after the last synchronization task was triggered, a new one is triggered.

During a synchronization operation, the network connection is checked; if there's not network connection the synchronization process is aborted. If, however there's network connection the local database is queried for all visitor positions data that are to be synchronized. Next, the data is sent in buffers of six elements, or less if there are not many elements left to send to the server. Also, if some error happens during the synchronization of one buffer of data then all the synchronization process is cancelled.

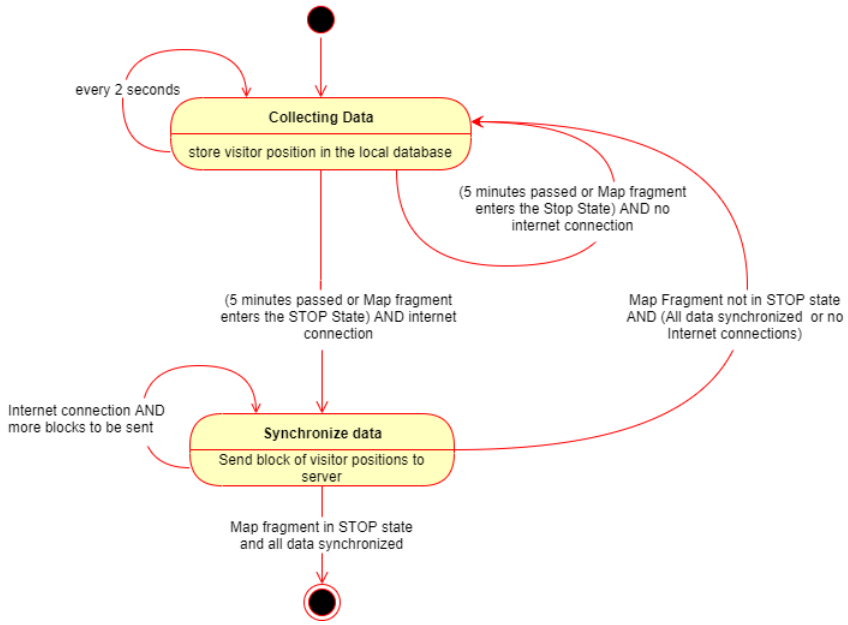

Figure 4.9. State diagram for the data synchronization. The collecion of data occurs in parallel to the synchronization of data.

Figure 4.10 shows in greater detail a successful synchronization scenario. First the local database is queried for all visitor positions (*getAllUnsynchronizedVisitorPoints()*). If the query returns some positions (*points*) then these are sent in buffers of six or less, attached as JSON objects to POST requests to the /visitors/points route, used to store visitor positions. Once the data reaches the server it is parsed, and it is validated if its format is correct and the visitor exists. If no errors are found, then the data is inserted into the remote database, and a 200 OK response is sent to the SyncAdapter. The synchronization of the visitor positions is tracked through the *lastSynchronizationTime* of the *Configuration* table in the local database (see Figure 3.6). When a synchronization is successfully completed, this field is replaced by the time the last synchronized point of interest was created. When querying the local database for points that still need to be synchronized, it is checked if the date they were recorded is more recent then the last successful synchronization.

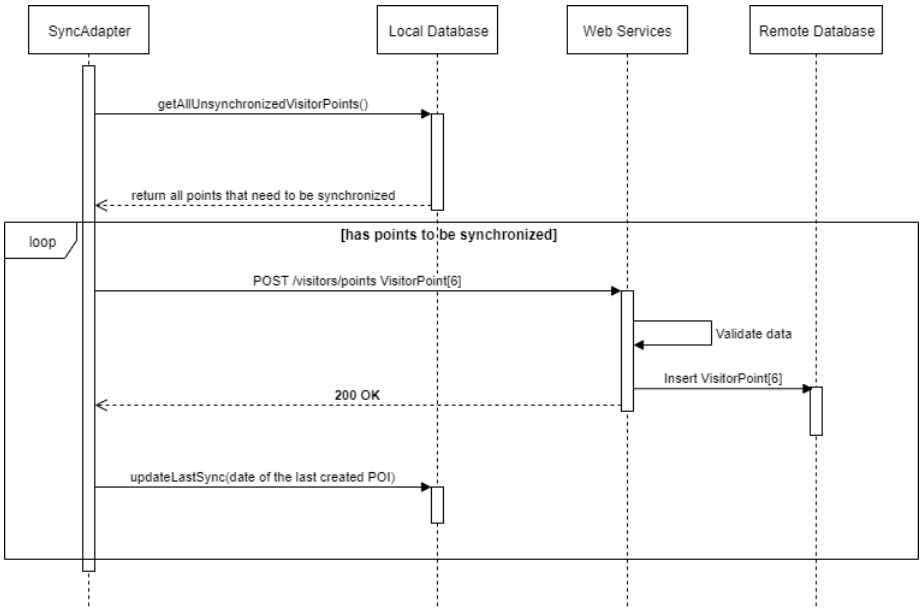

Figure 4.10. Sequence diagram of a successful synchronization process. *VisitorPoint[6]* indicates that a group of six VisitorPoint objects are sent with the request, in the context of the *POST* and in the inserted in the *Insert* operation.

4.3 Summary

In this chapter the implementation details of the features and components of the system were presented. We talked about the remote database installation and management, the web services and the public storage on the backend. We also talked about the architecture of the mobile application and explored features of the application such as, the download component, the interactive map, the visitor orientation and progress components and finally the visitor positions synchronization component. In the next chapter the activities executed in order to evaluate the developed system are detailed.

Chapter 5

System evaluation

In this chapter are described activities executed in order to evaluate the prototype described in chapters 3 and 4. We start with the expert testing and collection of user feedback carried out in order to assess the usability of the system, followed by the server performance tests. At the end of this chapter we discuss the results of these activities and are presented suggestions and improvements derived from this discussion.

5.1 Expert testing

The first type of evaluation of the system was based upon tests with experts, who installed the mobile app in their own smart phones and carried out the tours featured in the app on site. The experts at hand provided the contents for the various tours in the application.

5.1.1 Objectives

These tests were with experts designed to meet the following objectives:

- Understand if the implemented tours match the vision of the experts for the Birds (*Aves*) and Historical (*Histórico*) tour
- Identify issues with the data and contents presented in the app (e.g. if the points of interest were correctly positioned).
- Receive feedback on what could be improved or added to the interaction.
- Get a general understanding of the usability of the system.

5.1.2 Participants

Three experts participated in the evaluation of the system:

- **Participant E1** – Professor César Garcia, from the birds (*Aves*) tour team.
- **Participant E2** – Professor Ana Leal, from the birds (*Aves*) tour team.
- **Participant E3** – Professor Ana Godinho Coelho Dotti de Carvalho, from the Historical (*Histórico*) tour team.

The participants used smartphones running the Android operating system, and all of them None had never used a mobile application in the context of a museum or botanical garden. However they were familiar with map services (e.g. Google Maps [16]), to find the location of a certain object (e.g., a building, monument, or street).

5.1.3 Tasks

The participants were presented with one of two tasks: **T1** – explore the birds tour; **T2** – explore the historical tour. Each participant executed one of these tasks according to the tour they contributed to. Both tasks begin with the following steps:

1. Select the respective tour from the tour list.
2. Start the tour. This step includes downloading the tour's contents.
3. With the map open walk towards any position marked on the map.
4. Tap the marked position.

Starting from this point, tasks T1 and T2 start differing from each other. On Task T1, because the birds tour uses areas of observation containing multiple points of interest, a list of the included points of interest in the area is presented. Additionally, augmented reality experiences are available in the list of points of interest, as well as on the details of some of the points of interest. As such, **T1** comprised additionally the following steps:

5. View the augmented reality experience of the observation area.
6. Select a point of interest from the list to view its contents.
7. View the augmented reality experience of the point of interest.
8. Go back to the list of points of interest of the area of observation and repeat steps 6 to 7 until all the points of interest in it have been visited.
9. Go back to the map.
10. Repeat steps 3 to 9 until all points of interest have been visited.

Task T2, is oriented towards exploring the Historical tour, so it has the filtering functionality. In this tour, there is a one-to-one correspondence between a marked position on the map and a point of interest, which is different from the one-to-many relation present in the Birds tour, resulting from the observation areas. As such, task T2 comprised the following additional steps:

5. View the augmented reality experience of the point of interest.
6. Go back to the map.
7. Use the filters to hide the points of interests that belong to the XX century period.
8. Show the points of interest that belong to the XX century period.
9. Repeat steps 3 to 6 until all points of interest have been visited.

5.1.4 Apparatus

To execute these tests, the developed application was installed on the mobile phones of participants E1 and E2. E1 used a Samsung Galaxy S7 Edge running Android Oreo (8.0), with a 1440 x 2560 pixels screen, 4GB or RAM, and an octacore Exynos 8890 chipset. E2 used a Huawei P Smart 2019 running Android Pie (9.0) with a 1080 x 2340 pixels screen, with 3GB of RAM, and an octa-core Hisilicon Kirin 710 chipset. Participant E3 did not previously install the application and since there was no Wi-Fi network in the garden my phone was used, a Samsung Galaxy S9 running Android Pie (9.0), with a 1440 x 2960 pixels screen, 4GB of RAM and an octa-core Exyos 9810 chipset. All phones featured a gyroscope, a magnetic field sensor and an accelerometer.

5.1.5 Procedure

Two test sessions were conducted. One session was focused on the exploration of the Birds tour, which had the participation of E1 and E2 and the other session focused on the exploration of the Historic (*Histórico*) tour, which had the participation of E3, member of the historic tour team.

At the start of each session the participants were asked to explore a certain tour. As participants followed the tour their suggestions were registered either with text or, if alterations were to be made in

the map, an annotated print screen of the map in the position where errors were detected. Finally, at the end of the tour an inquiry was filled by the participants, containing some questions focused on features of the tour that is being explored, and others on aspects common to all tours.

5.1.6 Results

The inquiries at the end of the test sessions, revealed concerns about the general structure of the tours featured in the app, as well as about tour specificities.

Regarding the general structure of tours, the experts had no trouble selecting the tour they were tasked with. The download durations for the tour contents was considered adequate, however E2 noted that, it would be necessary to warn users that they should download the contents in a place where there is Wi-Fi network, otherwise the download would cost a considerable amount of mobile data.

In map view all experts were able to find out the path that they needed to follow, and also to distinguish between the visited points of interest and the not yet visited. All experts were able to distinguish between the points with and without augmented reality experiences, however E2 suggested that the augmented reality icon should be increased, as to be easier to view. Furthermore, in the context of the point of interest detail screen, regarding to adequacy of a map button to lead visitors back to the map, even though there already is an up button (back arrow) that does the same job, all participants found this feature adequate. Finally, the overall structure of the information presented to the users in the application was considered adequate.

Regarding the concerns specific to the Birds tour, although E1 was able to identify all augmented reality experiences present in the tour, both at the area of observation level, and at the point of interest level, E2 had some trouble identifying the points of interest that had augmented reality experiences, as the icons in the list of POI of the observation area were too small, increasing the difficulty of the task.

Regarding the concerns specific to the Historic tour, E3 suggested that the filters should not completely hide the filtered points of interest and instead, it would be better they were transparent in order to transmit to the visitor information about all the points of interest that are included in the tour at any given time, even on smaller mobile devices.

When asked what the participants would change in the current version of the application, E1 suggested that it should be shown the orientation in addition to the location of the visitor in the map (similar to the navigation mode with *Google Maps* [16]). Also, E3 suggested an alternative approach to the exploration of the tours, where the user should be allowed to wonder around the garden and be notified when near any point of interest.

Lastly in all the test sessions, errors in the position of some points of interest included in both the birds tour and the historic tour were noted and afterwards corrected.

5.2 User feedback

A version of the mobile application with the Botanic tour was presented at the *Encontro Ciência* event at *Centro de Congressos de Lisboa*, on the 10th of Jully 2019. This presentation was also used to get user feedback on the implementation of the tour.

With this feedback we intended to:

- Evaluate visitor acceptance and excitement with the implemented system.
- Get diverse feedback about features that could be added to the application.
- Understand if the interface was comprehensive and what could be done to improve it.

Feedback was collected from 20 participants. Each of the participants in this study were presented with only one task: **T** – explore the botanical (*Botânico*) tour. To complete this task, users had to follow a few steps:

1. Select the tour from the tour list.
2. Start the tour.
3. Select one of the marked points of interest in the map and view its contents.
4. Go back to map.
5. Repeat 3 and 4 until all the points of interest have been viewed.

To obtain user feedback, the mobile application was installed on my phone (see specification on section 5.1.4), and my colleagues' phone, which is a Huawei P10 running Android Oreo (8.0), with a 1080 x 1920 pixels display, 4GB of RAM, and an octa-core HiSilicon Kirin 960 chipset.

The participants were first asked if they were interested in testing the application. If the participants accepted, a brief presentation was made to contextualize the mobile application and explain what the application offered. After the explanation, the users were left to explore the tour. As users explored the tour, notes were taken of issues raised while using the application, as well as suggestions for improving the mobile application.

Most of the collected feedback contained suggestions in order to improve the augmented reality experiences, included in the tour. However, some suggestions helped shape future versions of the user interface.

One of the suggestions was that a caption should be added on the map, in order to distinguish between the seen and not yet seen points of interest. Another suggestion that was considered was to change the title of the topics marked with JBT notes (*Notas JBT*), to observations (*observações*). This topic contained additional information about a given specimen that distinguishes it from other plants of the same species. Next, it was suggested that the expand button in topics with large amounts of text should be modified in order to be more visible to the users. Another suggestion was to present, not only the common names in Portugal, but also in other Portuguese speaking countries. Finally, it was suggested that the tours list should be accessible from all the views screens of the application.

5.3 Performance tests

The last type of evaluation consisted on testing the performance of the server by sending different

5.3.1 Objectives

As an important part of the developed system is the server, tests were made in order to study the performance of it. These tests were made with the following objectives:

- Understand how the duration per request, evolves as the number of concurrent requests is increased.
- Observe how the duration per request increases as the data that is sent as a response for the request increases in size.

5.3.2 Apparatus

All tests were made in the same place in an office at LASIGE and were executed on a PC with an Intel Core ™ i7-8750H processor and 16GB of RAM. In all tests the computer was connected to an ethernet cable, which enabled for a very stable internet connection. The server had a two core Intel Xeon CPU E5-2660 v2 @ 2.20GHz CPU with 3GB of RAM.

A script was implemented in Python using the urlib, and multithreading libraries, that repeatedly made requests to the /tours/{tour id}/pois, simulating the users fetching data about the points of interest and contents that belong to a given tour in parallel. The script can be executed through the

command line, and has two parameters, one optional argument -t in which the tour that is being requested is specified, and a positional argument number_threads which specifies the number of threads that are used do simulate parallel clients. As such, when a client sends a request to /tours/{tour id}/pois, in a successful scenario, the body of a response has the following structure:

```
[
    {
        "id":<id of the point of interest>,
        "longitude":"<longitude of point of interest>",
        "latitude":"<latitude of the location of point of interest>",
        "name":"<Name of the point of interest>",
        "type":"<type of the point of interest>",
    "subtype":"<subtype of the point of interest>" // can be null,
        "average_stopping_time":"HH:mm:ss",
        "age":"<type of the point of interest>" // can be null,
        "order":<order of the point of interest>,
        "images":
            [
                {
                    "content":<id of the content>,
                    "base_name":"<base name of the image file>"
                }
                ...
            ],
        "description":<id of the description>,
        "topics":
            [
                {
                    "order":<order of the topic>,
                    "topic":"<name of the topic>",
                    "base":"<the part of the text that is always visible>",
                    "see_more":"<the part of the text that may be collapsed>"// can be
null
                }
                ...
            ]
    }
    ...
]
```

For each JSON object contained in the response we first get information about the point of interest (*id*, *longitude*, *latitude*, *name*, *type*, *subtype*, *average_stopping_time*, and *age*). Note that these key-value pairs are based on the attributes from the PointOfInterest table of the database (see Figure 3.9). The *order* attribute, which is part of the TourPointOfInterest table, indicates the order in which this point of interest should appear on the tour.

Next comes data related to the contents of the points of interest included in the tour: *images*; *description;* and *topics*. The *images* are an array of JSON objects containing information about the images used in the details page of the points of interest. Each JSON object in this array has two attributes: the content, which is the identificatory of the content that is the image; and the base_name

which is the base name of the file (ex: motacilla_alba_1.jpg). Next the *description* contains the identifier of the textual description of the point of interest. Finally, *topics* contains an array of JSON objects representing the topics that make up the textual description. Each JSON object contains: an *order* which indicates the order in which the topic must be presented; the *topic*, which contains the name of the topic or section of the description; the base and see_more, contain the text that should always be visible and the text that can be collapsed respectively, when the text exceeds the established threshold of 650 characters.

Now that we understand how the response is structured, the response for each of the tours can be further detailed. As such in terms of the number of points of interest included in the tour: the botanic tour (id = 1) contains 20 points of interest; the historic tour (id = 3) contains 31 points of interest; the birds tour (id = 5) contains 55 points of interest. In terms of the size in bytes of the response: the Botanic tour's response has 48335 bytes; the Historic tour's has 30902 bytes; and the Birds tour's has 70330 bytes.

5.3.3 Procedure

The implemented simulation script was run three times for each tour, with an increasing number of clients (threads). For every task the simulated clients had the objective of finishing a fixed load of 2000 requests. The round trip for each request would be timed and this as well as the id of the tour, the number of parallel clients in the test, the date time with day, month, year, hour, minutes, seconds and milliseconds, and finally, also if there was an error, for each request would be written to a csv file. The tests were executed for 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 parallel clients (threads).

Before executing each test, the server was pinged, and the result noted. Additionally, as the tests ran, the memory and CPU usage in the PC and in the server was recorded.

5.3.4 Results

Network speed and memory and CPU usage

After analysing the noted results from the pings, it was concluded that for all tests the duration for sending and receiving a packet would take from 2 to 3 ms. As for the CPU and memory usage in both the client and the server, it was also concluded that memory swap was never used, and for both the client and the server, no more than 55% of the memory was used. As for the CPU, in the client only 20 to 40 % of the CPU was used at any given time, and in the server only from 16 to 55 % of the CPU was needed.

Correlating the number of threads with the duration

The data produced by the simulator script for the response times was analysed, and it was found that once in a while, for less than 5% of the requests, the duration would be too short or too fast. As such an inter quantile outlier removal algorithm was applied to the data and the result was plotted in a box plot (see Figure 5.1).

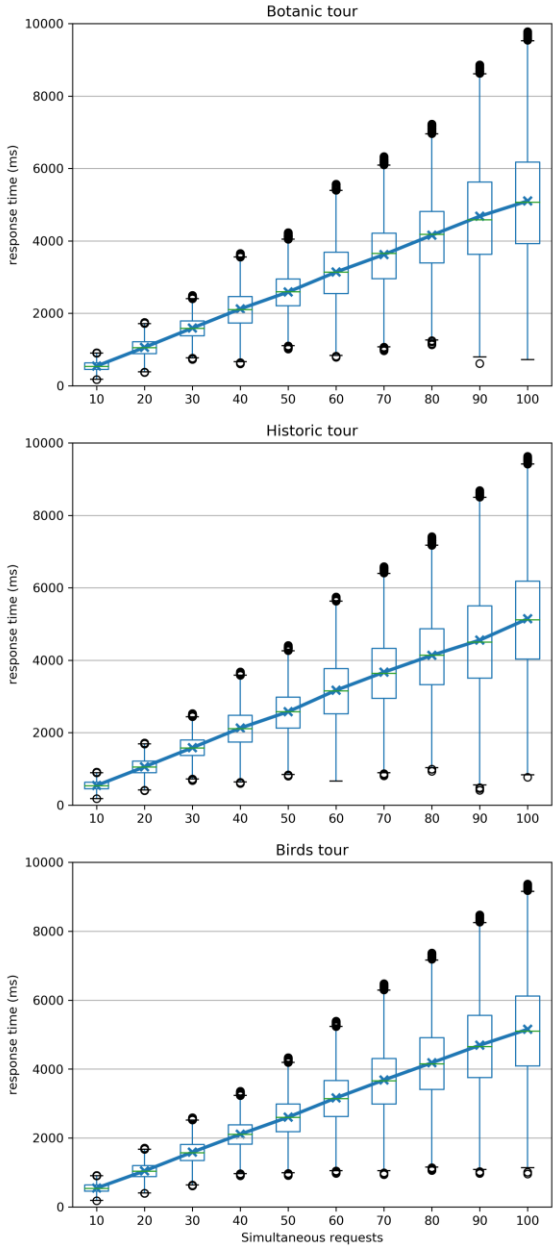

Figure 5.1. Response time related to the number of simultaneous requests. In this diagram we can visualize the distribution of the values for the response time in the form of box plots, as well as the evolution of the average duration time per request marked by the blue thick line. The circles in the extremities of the bar plots are outliers.

Figure 5.1 shows when the number of simultaneous requests is increased, the duration per request increases as well as the variation of the duration. We can also observe that the average duration time progresses in a linear manner, as illustrated by the blue line connecting the means of the response times for each number of simultaneous requests. Since in visitors of the JBT will use Wi-Fi networks it is expected for these values to be greater. Also, 100 was chosen as the maximum number of simultaneous requests, as it was considered a reasonable value for the number of users that would be using the mobile application simultaneously, at any given time.

Additionally, for each tour, an equation that relates the number of simultaneous requests and the average duration (mean) per request, was found using linear regression. As such for the Botanic tour the line that defines this relationship is described by the following function $y = 51.06x + 51.8$; for the Historic tour the line is described by the following function $y = 50.84x + 59.90$; finally for the

Birds tour the line is described by the following function $y = 51.65x + 37.19$. The result of all these functions is in milliseconds.

5.4 Discussion

The Encontro Ciência event, where user feedback was collected, took place before the expert study I will first discuss the changes deriving from the collected suggestions. Next the results from the expert's study will be analyzed. And finally, the results obtained from the performance tests are analyzed.

From all the user feedback, only one suggestion was integrated in the newer prototypes. This change consists on the addition of a caption on a map that differentiates the visited and the not yet visited points of interest. As for the other suggestions, since in most cases they were only presented by one subject, and there was limited time, it was decided to discard them. One of the suggestions was not accepted, namely changing the JBT notes to observations, based upon the opinion of the person in charge of providing the description for the Botanic tour. Figure 5.2 shows the before and after versions of the map in the app resulting from the feedback with users.

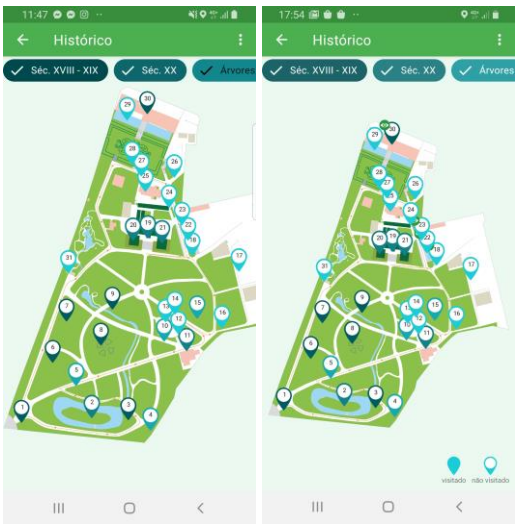

Figure 5.2. Visited and not visited points of interest markers caption – comparison between application interface before and after the addition of the caption distinguishing between the visited and not visited points of interest.

The expert study revealed that for the most part, the participants were interested and were positive about the value of the implemented application, but they also presented some suggestions on how to improve the mobile application. Most of the suggestions were related to corrections in the locations of points of interest, and also the map, which were carried out after the tests using QGIS. Other suggestions, such as those provided by Professor Ana Leal (participant P2), for example resizing of the augmented reality icons, were also applied and integrated in newer versions of the application. The suggestion by Professor César Garcia, concerning the display of the visitor's orientation in the map view, was also addressed and featured in the newer versions of the application. The suggestion provided by Professor Ana Godinho Dotti of letting the visitor walk while it is notified of nearby points of interest (see sub-section 5.1.6), was not applied to the current version of the application, as it would take too much effort, and considerably alters the way the application works. However, as this is an interesting feature, it was considered for future work on the application.

Regarding the performance tests, they show that the server behaves as expected: as the number of simultaneous requests increases, the average duration per request increases and also the standard

deviation in a linear manner. This behavior is explained by the fact that since multiple requests arrive at the same time, some may need to wait to be processed, while others are processed immediately.

However, once in a while, errors were detected. These errors were investigated, and it was verified that the CPU usage on both the client and the server did not surpass 50 % for these tests. It was also found that the memory usage has never reached 100% of usage. As such it was concluded that these errors did not happen because of hardware limitations.

It was also found that the duration of the requests that had errors was considerably longer than others, taking normally around 21 seconds to be answered. By testing for a greater load of simultaneous requests (150) it was found that this error was thrown when the client times out, or the server does not respond correctly.

Next, the error log of the server application was checked and it was found that all the errors logged were related to Nginx not being able to connect to php-fpm.sock. Php-fpm stands for *PHP fast CGI* process manager, and it is a service that executes separately from the web server process, in charge of managing a set of processes that process PHP scripts. PHP-fpm is then used to execute the PHP script(s) used to connect an URL to some method, execute it, and provide the result to *Nginx* which sends the response back to the client. In order for the Nginx server to communicate with PHP-fpm, a socket is opened between these processes, for Nginx to send the requests to PHP-fpm. As no hardware limitations were observed, we can assume that the origin of these errors is related to the number of processes managed by PHP-FPM not being sufficient to answer the incoming load.

As such, a possible solution to the identified problem could be to increase the number of processes that are made available to PHP-FPM by default (five processes), to six or more. Since the tests must be executed in the same place, the duration of a complete testing session for a given number of processes managed by the PHP-fpm takes a long time, this experiment was considered for future work. Also, as these errors occurred with very low frequency (2 in 6000 requests) they were ignored and removed from the samples used to draw the graphs.

If we consider that while the number of simultaneous requests is increased, the average duration per request grows in a linear manner, we can then predict the average duration per request, if we further increase the number of simultaneous requests of a given tour, by the functions presented in the subsection. For instance, if we increase the number of simultaneous requests to 110, and apply the equations presented in subsection 5.3.4, then the average duration per request for the Birds would be of 5717 milliseconds, 5652 for the Historic tour and 5668 for the Botanic tour. It would also be interesting to correlate these results with the contents and dimension in bytes of the JSON response however due to time constraints this was also considered for future work.

5.5 Summary

In these chapter it was described how the expert tests and performance tests were conducted, and how we obtained additional user feedback. It was also presented the results from all the tests and feedback collection, and the steps taken to address the suggestions made by the experts and users. Furthermore, the results from the expert testing were analysed, and it was explained the origin of errors that occurred within theses tests. It was also given an estimate of the possible outcome of the tests with higher load of simultaneous requests, considering that the growth is linear, and that it follows the function obtained for fitting the data.

In the next chapter the contributions made with the completion of this project, the acquired skills, the challenges and suggestions of future work are presented.

Chapter 6

Conclusions

This chapter presents the main contributions of the project, acquired skills, challenges that raised during the development of the project, and suggestions for future work.

6.1 Contributions

Considering the first objective of the project – the development of a mobile application with a curated list of tours, for the Tropical Botanical Garden of Lisbon (*Jardim Botânico Tropical de Lisboa*) – it was accomplished with the creation of the first version of the app with three tours the Botanical tour ("*Botânico (Especialistas)*"), the Historical tour ("*Histórico*") and the Birds tour (*"Aves"*), each with a different thematic, and directed to different types of visitors. The mobile application offers an interactive navigation system built from scratch on top of an image of the map created based on collected high accuracy GPS data about the garden. In this map the tour is displayed with positions for attractions marked on the map, as well as a recommended path to follow through the garden. Additionally, the visitor position and orientation are displayed on the map, using the GPS receiver and phone sensors, such as the gyroscope, accelerometer and magnetic field, respectively. For each position marked a description is presented about the point of interest.

The second objective – creation of an application server layer, to store contents and data that are used in the mobile application – was fulfilled by implementing a web application, which exposes a REST API to the clients that enables them to download data and files, to describe the different tours provided and points of interest.

The third, and last objective – system evaluation – was accomplished by collecting user feedback, realizing tests with experts and performance tests with the server. By collecting user feedback and doing expert testing we were able to get a general understanding of the usability and user acceptance of the system. With the performance tests we were able to understand how the response time will grow as the payload is increased and also predict the duration for increased payload sizes. We were also able to identify problems that may appear for greater loads of work, and solutions were presented to these problems.

6.2 Acquired skills

At the software development level, it was learned how to design and implement a software system, for real world application. During the different phases of this project it was acquired knowledge on how to analyse a real-world problem, and correctly model a solution by defining the systems requirements as well as designing its architecture, and structure of all components that make the system. A more in-depth knowledge on android development was acquired. It was learned how to

apply different patterns to the structure of an android mobile app, as well as how to use various types of libraries provided booth by Android and third parties. Additionally, it was learned how to implement Web services using the LUMEN PHP framework. Finally, I it was the intricacies of the PostgreSQL syntax and how to use the different aggregation functionalities in order to make the returned data set follow a certain format (e.g. JSON).

With the development of this project it was learned how to use a various set of tools that may be useful in future projects. It was also learned how to create vector assets, and how to edit images using both the Adobe Illustrator and Photoshop as well as the Affinity Photo and Design. I've learned how to use geographical information systems such as QGIS and ArcGIS. It was also learned how to use the Postman Tool in order to test API's. Lastly, It was learned how to use the different functions provided in the Android Studio IDE such as the profiler to study the performance of the app, diagnose memory leaks and how to improve the code based on these diagnoses.

Nevertheless, I've also acquired some soft skills such as: critical thinking, emotional intelligence, coordination, and decision making. All these skills are important when dealing with tight deadlines for a product and when dealing with large teams, with elements from different areas that have different perspectives.

6.3 Challenges

The biggest challenge in this project was to work is in an interdisciplinary team in which member have different perspectives on the app and consolidating their views in a single quality software product. As this project was developed in collaboration with a big multidisciplinary team, various interviews and demonstrations of the app had to be prepared in order to establish a fixed and robust set of requirements for the system. In these interviews alternatives to implementations of features were discussed, and in some cases the suggested alternatives were not considered, such as when they involved fundamental changes to the system.

Another challenge was the implementation of the map. An extended analysis of the available technologies that could be used to implement the application was made, by visualizing garden geographical data on all these map services. The result of this analysis was to implement a map from scratch by overlaying geographical data on an image view, that provided the zoom and pan actions to the user. To implement this solution first a new map was created based on geographical data provided by the garden staff and it was necessary to transform any geographical position into a position in the map. The created map had to be exhaustively tested in order to guarantee that the drawing is accurate.

Additionally, in order to show the orientation of the visitor, a fusion of data from different sensors available on the phone was made, which was not a trivial task since it was necessary to learn how these sensors work and how the data from them can be filtered to give accurate results for the dimensions they are measuring.

Finally, the implementation of the web services was straight forward however implementing the communication of the app with the services some complex process. Several approaches were explored to implement the download functionalities to provide the current download performance. Also, to implement the synchronization of visitor positions data with the server, various adjustments had to be made in order to achieve the solution used currently. The implementation of both these features required to learn how to use various libraries and components, such as SyncAdapter, Retrofit, OkHttp and Moshi.

6.4 Future work

Several features, improvements and components that could be added to the system. These elements were collected as biproducts from the analysis, design and implementation phases, in the development of the application, as well as from the workshops, user feedback and tests done in order to evaluate the quality of the system.

First some features need to be implemented before making the app available to the public: a form to collect demographical data of the user; a Ludic tour and a Free (*Livre*) tour; an "About" page, where the artists as well as the team are properly credited and presented, as some contents featured in the app require licenses and were provided by artists outside the team in charge of project *Jardim XXI*.

The Free tour would enable the visitors to search for the points of interest and create their own tours. The Ludic tour would be similar to a structured tour (for example the Botanic (*Botânico*) tour), with a gamification component. As such at the beginning only the starting point of interest would be presented and the users would have to defeat the challenges presented to progress to the next points of interest, this process being repeated for following points.

Next, some features could be added to the functionality of the application to make the interaction with it more modern and fluent, such as favouriting and sharing features, as well as recommendations based on user choices. Also, it should be added a dashboard where the user could edit its demographical information as well as data collected from usage, such as favourited points of interest and history of the visited points of interest. A system for creating groups based on code sharing could be implemented, to enable the user to share his/her experience with friends or group. This is interesting especially for the Ludic tour.

Furthermore, as observed in most applications, information about the garden such as entry price, parking hours, and geographical location should be added in the application in order to enable users to better prepare for the visit of the garden, and also to help bring visitors to the garden. Additionally, information about the regulation within the garden should be made available in the application.

Since the application should be made available to tourists more languages such as English and French should be added in order to make the app inclusive to tourists from outside Portugal and other Portuguese speaking countries.

As visitors not only have different languages but also devices with different operating systems, a new version of the application could be implemented using *Flutter*, *React Native* or *NativeScript*, all of which are cross platform frameworks that enable the deployment of applications for *Android* and *iOS* using the same language.

The current version of the application does not synchronize modifications of contents on the server with the application. Thus, a synchronization should also be implemented that updates the contents on the mobile applications when changes occur on the server.

Finally, to improve the user interaction a brand a new colour palette, and user interface could be adopted in the application in order.

On the server side, a Web application could be implemented that would enable the garden staff to easily and independently from the development team, to add or modify tours, points of interests and contents. Another future task is an application to analyse the collected data from the visitors. With the visitor data, we can extract information about, the most visited and most liked points of interest. We can also identify what tours need to be improved based on the classifications and feedback attributed by the users. Finally, with the visitors' locations, heatmaps can be created to understand the behaviour of the visitors' inside the garden.

Finally a more complex representation of the data about the garden could be implemented, by storing not only positions of points of interest but also a representation of the garden area, the streets inside it and the green areas inside it this way more interesting applications could be implemented

such air quality and temperature studies in different areas of the garden. This change would also help to dynamically draw a suggested path on the map and improve the estimation of the duration of a tour.

From the user feedback and the expert test it was marked for future work that the interaction with the map could be improved by adding a scrollable horizontal list of points of interest, where the user could select the points of interest that he/she wants to view by name, in the context of a structured tour. Also, a background service could be added in order to notify the visitor when it has reached a position close to a point of interest, to make the interaction less intrusive and enable the visitor to better contemplate its surroundings.

Bibliography

[1] D. Rakow, “What is a public garden?,” in *Public Garden Management: A Complete Guide to the Planning and Administration of Botanical Gardens and Arboreta*, 2 edition., Hoboken, N.J: Wiley, 2011, pp. 3–14.

[2] S. Lacerte, “Public Gardens and Their Communities: The Value of Outreach,” in *Public Garden Management: A Complete Guide to the Planning and Administration of Botanical Gardens and Arboreta*, 2 edition., Hoboken, N.J: Wiley, 2011, pp. 175–189.

[3] M. Z. Gough and J. Accordino, “Public Gardens as Sustainable Community Development Partners: Motivations, Perceived Benefits, and Challenges,” *Urban Affairs Review*, vol. 49, no. 6, pp. 851–887, Nov. 2013, doi: 10.1177/1078087413477634.

[4] “Retrofit.” [Online]. Available: https://square.github.io/retrofit/. [Accessed: 22-Aug-2019].

[5] c.sousa, “Jardins,” 21-May-2014. [Online]. Available: https://www.ulisboa.pt/info/jardins. [Accessed: 11-Dec-2019].

[6] “Lisbon Tropical Botanical Garden | European Network of Historic Gardens.” [Online]. Available: http://europeanhistoricgardens.eu/en/portfolio-item/lisbon-tropical-botanical-garden/. [Accessed: 11-Dec-2019].

[7] L. Velho and F. Groetaers, “Jobim botanic,” in *SIGGRAPH Asia 2014 Mobile Graphics and Interactive Applications on - SA ’14*, Shenzhen, China, 2014, pp. 1–6, doi: 10.1145/2669062.2669065.

[8] R. T. Azuma and I. Malibu, “Malibu, CA 90265 azuma@isl.hrl.hac.com,” p. 31.

[9] A. Tomiuc, “Navigating Culture. Enhancing Visitor Museum Experience through Mobile Technologies. From Smartphone to Google Glass.,” *Journal of Media Research, vol 7, issue 3 (20) / 2014*.

[10] Jardim Botânico do Rio de Janeiro, “Jardim Botânico RJ - Apps on Google Play,” *Jardim Botânico RJ - Apps on Google Play*. [Online]. Available: https://play.google.com/store/apps/details?id=br.gov.jbrj.visitante&hl=en_US. [Accessed: 14-Aug-2019].

[11] The Royal Botanic Gardens & Domain Trust, “Royal Botanic Garden Sydney,” *App Store*. [Online]. Available: https://apps.apple.com/au/app/royal-botanic-garden-sydney/id1332941220. [Accessed: 14-Aug-2019].

[12] Mobile 72, “RJB, museo vivo,” *App Store*. [Online]. Available: https://apps.apple.com/es/app/rjb-museo-vivo/id1136045846. [Accessed: 14-Aug-2019].

[13] C. Mann and H. Walk, “A study of the iPhone app at Kew Gardens: Improving the visitor experience,” p. 7.

[14] “Jardim Botanico da Madeira – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.Madeira_BG&hl=pt_PT. [Accessed: 21-Nov-2019].

[15] “Plants with Bite - Royal Botanic Garden Sydney - Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.specialistapps.PlantsWithBite&hl=en_US. [Accessed: 04-Dec-2019].

[16] “Overview | Maps SDK for Android,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro. [Accessed: 12-Aug-2019].

[17] M. Economou and E. Meintani, “Promising Beginnings? Evaluating Museum Mobile Phone Apps,” *Proceedings of the Re-Thinking Technology In Museums: Emerging Experience Conference, Limerick, Ireland*, pp. 87–101, 2011.
[18] “Mobile App Store Revenue: Major Gains Expected,” *App Annie Content*. [Online]. Available: https://www.appannie.com/en/insights/market-data/app-annie-forecast-2017-mobile-app-store-revenue-exceed-139-billion-2021/. [Accessed: 11-Aug-2019].
[19] “App KHM Stories.” [Online]. Available: https://www.khm.at/en/learn/kunstvermittlung/app-khm-stories/. [Accessed: 11-Aug-2019].
[20] “Rijksmuseum - Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=nl.rijksmuseum.mmt&hl=en_US. [Accessed: 09-Dec-2019].
[21] A. S. Author, “Magic Tate Ball by Tate Gallery,” *AppAdvice*. [Online]. Available: /app/magic-tate-ball/451772129. [Accessed: 21-Nov-2019].
[22] “Tate Modern : How It Is App,” 16-Feb-2010. [Online]. Available: https://adage.com/creativity/work/how-it-app/18951. [Accessed: 06-Sep-2019].
[23] A. S. Author, “Magritte Your World by Tate Gallery,” *AppAdvice*. [Online]. Available: /app/magritte-your-world/457005636. [Accessed: 04-Sep-2019].
[24] C. C. Abt, *Serious Games*. University Press of America, 1987.
[25] B. Bergeron, *Developing Serious Games (Game Development Series)*. Rockland, MA, USA: Charles River Media, Inc., 2005.
[26] D. R. Michael and S. L. Chen, *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade, 2005.
[27] D. Thompson *et al.*, “Serious Video Games for Health: How Behavioral Science Guided the Development of a Serious Video Game,” *Simul. Gaming*, vol. 41, no. 4, pp. 587–606, Aug. 2010, doi: 10.1177/1046878108328087.
[28] A. S. Author, “Tate Trumps by Tate Gallery,” *AppAdvice*. [Online]. Available: /app/tate-trumps/371670940. [Accessed: 24-Nov-2019].
[29] C. S.L, “CloudGuide,” *cloudguide*. [Online]. Available: http://www.cloudguide.me. [Accessed: 12-Aug-2019].
[30] “KeyARt - The museum app in Augmented Reality,” *KeyARt*. [Online]. Available: https://keyartapp.com/. [Accessed: 09-Dec-2019].
[31] “Lisboa Cool: guia de viagem – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.bloomidea.lisboacool&hl=pt_PT. [Accessed: 09-Dec-2019].
[32] London & Partners Ltd, “Visit London Official City Guide - Apps on Google Play,” *Visit London Official City Guide - Apps on Google Play*. [Online]. Available: https://play.google.com/store/apps/details?id=com.londonandpartners.londonguide&hl=en_US. [Accessed: 11-Aug-2019].
[33] Lonely Planet, “Guides by Lonely Planet - Apps on Google Play,” *Guides by Lonely Planet - Apps on Google Play*. [Online]. Available: https://play.google.com/store/apps/details?id=com.lonelyplanet.guides&hl=en. [Accessed: 13-Aug-2019].
[34] “Visit Korea : Official Guide – Aplicações no Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.visitkorea.eng&hl=pt_PT. [Accessed: 09-Dec-2019].
[35] TripAdvisor, “TripAdvisor Hotels Flights Restaurants Attractions - Apps on Google Play,” *TripAdvisor Hotels Flights Restaurants Attractions - Apps on Google Play*. [Online]. Available: https://play.google.com/store/apps/details?id=com.tripadvisor.tripadvisor&hl=en. [Accessed: 13-Aug-2019].
[36] P. Vergo, *New Museology*. Reaktion Books, 1989.
[37] “About SQLite.” [Online]. Available: https://sqlite.org/about.html. [Accessed: 23-Aug-2019].
[38] “Room Persistence Library,” *Android Developers*. [Online]. Available: https://developer.android.com/topic/libraries/architecture/room. [Accessed: 22-Aug-2019].

[39] “Oracle Berkeley DB.” [Online]. Available: https://www.oracle.com/database/technologies/related/berkeleydb.html. [Accessed: 22-Aug-2019].
[40] *A lightweight, document-oriented (NoSQL), syncable database engine for .NET: couchbase/couchbase-lite-net*. couchbase, 2019.
[41] *LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values.: google/leveldb*. Google, 2019.
[42] Bitrise, “Realm VS SQLite: Which database is better for Android apps?,” *Medium*, 03-Aug-2018. [Online]. Available: https://medium.com/bitrise/realm-vs-sqlite-which-database-is-better-for-android-apps-c751dc8b150c. [Accessed: 23-Aug-2019].
[43] “Overview | Maps SDK for Android,” *Google Developers*. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/intro. [Accessed: 23-Aug-2019].
[44] “Introduction,” *Mapbox*. [Online]. Available: https://www.mapbox.com/android/maps/overview/. [Accessed: 12-Aug-2019].
[45] “7.2. NextGIS Android SDK — NextGIS 1.11 documentation.” [Online]. Available: https://docs.nextgis.com/ngmobile_dev/ngmav3.html. [Accessed: 23-Aug-2019].
[46] *OpenStreetMap-Tools for Android. Contribute to osmdroid/osmdroid development by creating an account on GitHub*. osmdroid, 2019.
[47] *Offline Maps API for Android. Contribute to mapsme/api-android development by creating an account on GitHub*. MAPS.ME, 2019.
[48] “OpenStreetMap Wiki.” [Online]. Available: https://wiki.openstreetmap.org/wiki/Main_Page. [Accessed: 23-Aug-2019].
[49] “OsmAnd - Offline Mobile Maps and Navigation.” [Online]. Available: https://osmand.net/. [Accessed: 15-Dec-2019].
[50] “OkHttp.” [Online]. Available: https://square.github.io/okhttp/. [Accessed: 22-Aug-2019].
[51] *A modern JSON library for Kotlin and Java. Contribute to square/moshi development by creating an account on GitHub*. Square, 2019.
[52] “NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy,” *NGINX*. [Online]. Available: https://www.nginx.com/. [Accessed: 23-Aug-2019].
[53] “Welcome! - The Apache HTTP Server Project.” [Online]. Available: https://httpd.apache.org/. [Accessed: 23-Aug-2019].
[54] “NGINX vs. Apache: Our View of a Decade-Old Question,” *NGINX*, 09-Oct-2015. [Online]. Available: https://www.nginx.com/blog/nginx-vs-apache-our-view/. [Accessed: 24-Aug-2019].
[55] “Lumen - PHP Micro-Framework By Laravel.” [Online]. Available: https://lumen.laravel.com/. [Accessed: 24-Aug-2019].
[56] “Installation - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: https://laravel.com/docs/5.8. [Accessed: 24-Aug-2019].
[57] “PostgreSQL: The world’s most advanced open source database.” [Online]. Available: https://www.postgresql.org/. [Accessed: 24-Aug-2019].
[58] R. O. Obe and L. S. Hsu, “PostGIS in Action,” p. 28.
[59] “PostGIS — PostGIS Feature List.” [Online]. Available: https://postgis.net/features/. [Accessed: 15-Dec-2019].
[60] “Spatial Developer’s Guide.” [Online]. Available: https://docs.oracle.com/cd/E11882_01/appdev.112/e11830/sdo_locator.htm#SPATL340. [Accessed: 15-Dec-2019].
[61] “Downloads do SQL Server | Microsoft,” *Microsoft SQL Server - PT (Português)*. [Online]. Available: https://www.microsoft.com/pt-pt/sql-server/sql-server-downloads. [Accessed: 15-Dec-2019].
[62] “Postman | Products,” *Postman*. [Online]. Available: https://www.getpostman.com. [Accessed: 24-Aug-2019].
[63] “Database: Migrations - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: https://laravel.com/docs/5.8/migrations. [Accessed: 24-Aug-2019].
[64] “Database: Seeding - Laravel - The PHP Framework For Web Artisans.” [Online]. Available: https://laravel.com/docs/5.8/seeding. [Accessed: 24-Aug-2019].

[65] “Balsamiq. Rapid, effective and fun wireframing software. | Balsamiq.” [Online]. Available: https://balsamiq.com/. [Accessed: 28-Aug-2019].
[66] “Welcome to the QGIS project!” [Online]. Available: https://www.qgis.org/en/site/. [Accessed: 03-Sep-2019].
[67] “About ArcGIS | Mapping & Analytics Platform.” [Online]. Available: https://www.esri.com/en-us/arcgis/about-arcgis/overview. [Accessed: 03-Sep-2019].
[68] “Git.” [Online]. Available: https://git-scm.com/. [Accessed: 15-Dec-2019].
[69] “Download Android Studio and SDK tools,” *Android Developers*. [Online]. Available: https://developer.android.com/studio. [Accessed: 15-Dec-2019].
[70] P. Kruchten, *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 2004.
[71] “Home - PHP-FPM.” [Online]. Available: https://php-fpm.org/. [Accessed: 16-Dec-2019].
[72] “The color system,” *Material Design*. [Online]. Available: https://material.io/design/color/. [Accessed: 18-Dec-2019].
[73] Ming, *ShaoZeMing/lumen-postgis*. 2017.
[74] “Android Jetpack,” *Android Developers*. [Online]. Available: https://developer.android.com/jetpack. [Accessed: 22-Aug-2019].
[75] P. Lawitzki, “Paul Lawitzki | software developer and game designer.” [Online]. Available: http://plaw.info. [Accessed: 28-Oct-2019].

Appendix A – Expert testing Quiz – Botanic tour

Questionário da Visita: Percurso Botânico

1. Teve facilidade em escolher o percurso?

2. O tempo estimado correspondeu ao real?

3. Considera o tempo de descarregamento dos conteúdos adequado?

4. O posicionamento que a aplicação faz do visitante no mapa ajudou a realizar o percurso?

5. Conseguiu distinguir bem a sugestão do trajeto a percorrer?

6. Conseguiu fazer o percursos sem problemas?

7. Ficou claro quais os pontos que já foram visitados?
    a. Acha que visitou todos os pontos?

    b. Acha que não visitou algum ponto?

8. Considerou pertinente a existência do botão de voltar para o mapa, existindo a seta para voltar para trás?

9. Acha que a organização e apresentação da informação na aplicação é adequada, ou tem alguma sugestão?

10. Percebeu quais os pontos de interesse que têm realidade aumentada no mapa?

11. Percebeu as instruções para executar as experiências de realidade aumentada?

12. Tem comentários/sugestões sobre as experiências *time lapse*?

13. Que funcionalidade acrescentaria à aplicação?

Percurso histórico e livre

14. Achou pertinente a funcionalidade de filtrar os pontos que estão presentes?

Percurso das aves

15. Na experiência de realidade aumentada para procurar as aves:

    a. Percebeu as instruções?

    b. Deve-se parar o som que se está a ouvir quando se seleciona outra ave? Ou manter o som da(s) ave(s) anterior(es) até terminar o som do respectivo ficheiro de som?

c. As imagens das aves estão em posições adequadas?