Departamento de Engenharia Informática do
Instituto Superior de Engenharia do Porto

PSiS Mobile

Ricardo Manuel Soares Anacleto

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Área de Especialização em
Arquitectura, Sistemas e Redes

Orientadores
Professora Doutora Ana Maria Neves de Almeida Baptista Figueiredo
Professor Doutor Lino Manuel Baptista Figueiredo

Júri

Presidente: Prof. Doutora Maria de Fátima Coutinho Rodrigues, Professora Coordenadora, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Informática

Vogais: Prof. Doutor Paulo Jorge Novais, Professor Auxiliar, Universidade do Minho, Departamento de Engenharia Informática;
Prof. Doutora Ana Maria Neves de Almeida Baptista Figueiredo, Professora Coordenadora, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Informática;
Prof. Doutor Lino Manuel Baptista Figueiredo, Professor Adjunto, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Electrotécnica

Porto, Setembro de 2010

Acknowledgements

I would like to take this opportunity to thank all people that, more or less directly, helped me throughout all these years of study, especially on the end of this work, to obtain a masters' degree.

I'm very pleased for the given chance to participate in this nation-wide research project. It have bring to me responsibility, a substantial amount of pressure and stress to put all the pieces of the project, working together.

I would like to thank my advisors, Ana Almeida and Lino Figueiredo, which have accompanied my work since day one. Although they have a busy schedule, they always had time for helping me and to dedicate time to follow my work progression. All their knowledge have supported and improved my work.

I would also like to thank my family that played a decisive role in the conclusion of one more step in my academic life. They always have been there when I needed and have always support my decisions throughout my academic journey. But a very particular person must be enhanced, my girlfriend, although she doesn't have a deep knowledge in this area, helped me, bringing to me happiness and companion whenever I was developing this thesis.

Resumo Alargado

Os sistemas de recomendação têm vindo a ser cada vez mais utilizados nos últimos anos. Por isso, é imprescindível que estes sistemas se adaptem à evolução da sociedade incluindo cada vez mais novas funcionalidades, tais como a adaptação do sistema ao contexto da pessoa. Esta adaptação pode ser feita através de, por exemplo, dispositivos móveis, que têm vindo a apresentar uma taxa de crescimento de vendas muito grande.

Dada a crescente integração dos sistemas de recomendação com os sistemas móveis, foi elaborado um estudo sobre o estado da arte dos sistemas de auxílio ao turista que utilizam dispositivos móveis, sendo apresentadas as suas vantagens e desvantagens. Estes sistemas móveis de auxílio a turistas foram divididos em dois grupos: os que apresentam apenas a informação sobre pontos de interesse e os sistemas que são capazes de efectuar recomendações, com base no perfil do turista.

Um breve estudo sobre os sistemas operativos para dispositivos móveis é apresentado, sendo especialmente focado o sistema operativo Android que foi o escolhido para esta implementação.

Como os dispositivos móveis, actualmente, ainda possuem várias limitações, estas foram descritas e apresentadas as boas práticas no desenvolvimento de aplicações para este tipo de sistemas. É também apresentado um estudo que visa descobrir qual é o método mais leve e mais rápido para trocar dados entre a parte servidora e a parte móvel.

Com a parte introdutória apresentada, é exposto o projecto desenvolvido nesta tese, o PSiS Mobile. Este sistema é um módulo que faz parte do projecto PSiS e pretende trazer todas as vantagens dos sistemas móveis para o sistema base já implementado.

O projecto PSiS foca-se no estabelecimento de planos de visita personalizados com indicação de percursos para turistas com tempo limitado. Apoiando a definição de planos de visitas de acordo com o perfil do turista (interesses, valores pessoais, desejos, restrições, deficiências, etc.) combinando os

produtos de turismo mais adequados (locais de interesse, eventos, restaurantes, etc.) em itinerários eficientes.

A utilização de dispositivos móveis para acompanhamento da visita permite uma rápida interacção entre o turista e o sistema. Assim, o PSiS poderá recolher informação contextual do utilizador para que o perfil do mesmo seja enriquecido.

O sistema apresentado é composto por duas partes: a parte cliente e a parte servidora. Toda a informação, como por exemplo o perfil do turista, histórico de viagens e valores de similaridade entre utilizadores está presente na parte servidora. O processo de recomendação também é efectuado pela aplicação servidora, sendo esta a responsável pela atribuição de uma classificação aos pontos de interesse tendo em conta o perfil do utilizador em causa. A base de dados do PSiS possui toda a informação relativa aos pontos de interesse numa determinada cidade ou região e o portfólio completo do histórico de visitas de cada utilizador.

A componente móvel é uma parte muito importante para o sistema, pois interage com o utilizador no terreno. Um dispositivo móvel como o PDA, não só permite a apresentação de informação relevante ao utilizador, como também permite a recolha automática de informação contextual (por exemplo, a localização). Toda esta informação contribui para a definição de um perfil completo e para uma melhor adaptação do sistema às necessidades do utilizador.

De forma a nem sempre estar dependente do servidor, a aplicação móvel possui rotinas para a realização de recomendações básicas. Ou seja, a aplicação móvel não realiza a classificação dos pontos de interesse, mas apenas mostra os principais resultados já formados pela parte servidora. Por exemplo, se um utilizador gostar de comida Chinesa, um restaurante Chinês nas imediações irá ter uma boa classificação e, por isso, ser recomendado.

A aplicação móvel mostra ao turista o percurso definido para o dia em que o mesmo se encontra, sendo feito o rastreio do trajecto que o mesmo efectua. Assim, o sistema consegue saber se o horário do planeamento está a ser cumprido ou não. Caso não esteja, é invocado um algoritmo de planeamento que irá tentar corrigir o atraso ou o adiantamento perante o horário inicial. Depois de visitar um ponto de interesse, é pedido ao utilizador para fornecer feedback sobre o mesmo.

Se desejado também é possível mostrar os pontos de interesse existentes perto do turista (usando as coordenadas GPS obtidas pelo dispositivo móvel) organizados por categorias, raio de distância, etc.

Apesar dos dispositivos móveis possuírem várias restrições, pretendeu-se proporcionar ao utilizador uma boa experiência, através de uma aplicação rápida, de fácil utilização e adaptável, incluindo funcionalidades de planeamento, realidade aumentada e integração com a rede social do sistema. Todos estes factores contribuem para a disponibilização de informação detalhada ao turista.

Palavras-chave: Sistemas de recomendação móveis, Aplicações Cliente Servidor, Android, Sistemas de planeamento

Abstract

Recommendation systems have been growing in a relative number over the last years. But with the actual society evolution and expectations, these systems need to be improved to include new features, such as adapting the system to the context of the user. This adaptation can be performed using mobile devices, which nowadays are under an incredible growth rate in every business area.

Since recommendation and mobile systems might be integrated, this work presents the current state of the art in tourism recommendation systems using mobile devices, and states their advantages/disadvantages.

A brief study of mobile devices operating system is made and Android Operating System is described, presenting his functionalities and demonstrate how it works, in a software engineering way.

Since mobile devices have several limitations a study will be presented to discover the lightest and fastest way to exchange information between a server and a mobile client.

PSiS Mobile, that is the proposal of this thesis, is a mobile recommendation system and planning support, designed to provide an effective support during the visit of a tourist, providing context-aware information and recommendations about places of interest to visit based on tourist preferences and his current context.

Keywords: Mobile Recommendation System, Sight Information Provider, Context-Aware, Client-Server Application, Android, Planning System

Index

Figure Index

Table Index

Notation

AAPT	Android Asset Packaging Tool
API	Application Programming Interface
APK	Android Package
CMS	Content Management System
CPU	Central Processing Unit
DAL	Data Access Layer
DOM	Document Object Model
ENS	Event Notification Service
GECAD	Knowledge Engineering and Decision Support Group
GIS	Geographic Information System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HSDPA	High-Speed Downlink Packet Access

HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
ISEP/IPP	School of Engineering - Polytechnic of Porto
IST	Information Society Technologies
JSON	JavaScript Object Notation
kB	kiloByte
KML	Keyhole Markup Language
LAN	Local Area Network
LBS	Location Based System
MB	Megabyte
MMS	Multimedia Messaging Service
ms	Millisecond
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistance
POI	Point Of Interest
PSiS	Personalized Sightseeing Tours Recommendation System
QoS	Quality of Service
RAM	Random Access Memory
REST	REpresentation State Transfer
RIM	Research In Motion
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAX	Simple API for XML
SDK	Software Development Kit

SMS	Short Message Service
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TIP	Tourism Information Provider
UDDI	Universal Description, Discovery and Integration
UMTS	Universal Mobile Telecommunication System
URI	Uniform Resource Identifier
USB	Universal Serial Bus
VM	Virtual Machine
WAP	Wireless Application Protocol
WLAN	Wireless Area Network
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1
Introduction

When a tourist goes to a new location (country, city or region), he would certainly like to have available a user-friendly tool that would help planning his stay according to his objectives, preferences, knowledge, budget and period of stay. The task of planning where to go and what to do, in the limited amount of time available, are common problems encountered by tourists when visiting a city for the first time. In effect, cities are large information spaces and in order to navigate these spaces visitors often require numerous guide books and maps that provide large amounts of information. Häubl and Dellaert, 2004, state that this can be both a blessing and a curse. Although the amount of information allows tourists to select more appropriate points of interest (POIs), it also turns the process so complex that the tourist might not be able to assimilate all this information adequately. A recommendation system helps the tourist narrow the universe of choice, giving results according to the tourist preferences. Also, the system is able to process much more information and points of interest than the tourist could possibly do.

The recommendation system interaction with the tourist is also of utmost importance. When the tourist interacts with the system, every taken action has a meaning and can be used to complete the tourist profile. Nowadays, most systems build the tourist profile implicitly and/or use forms to request feedback. This is mostly possible through the use of mobile devices that travel along with the tourist providing context-aware information.

This type of system is very useful to use on a tourism situation, due its pocket size and computational capabilities. It can help tourist on planning his stay, show detailed information of a specific point of interest (POI) and recommend nearby points of interest to visit, according with his profile. With this, mobile devices have gained an even more significant role regarding the support of the tourist activity, being used as electronic guides, interactive or not.

Mobile terminals are embedded systems with very limited capabilities compared to a traditional computer, which needs to be considered because of possible technical, ergonomic or economic implications for the user. So, mobile applications performance is a crucial aspect that depends on many factors, especially when they consist of client-server applications. Although in recent years mobile technology has evolved significantly, in particular on storage, processing and communication settings, with the same time scale and lower prices, they have yet low performance, specially on time of battery available, that is the biggest obstacle to mobile performance growing.

However, for the mobile services provide effective support to the tourist in a trip, it should be considered the integration of a set of factors including actual available technology, such as bandwidth, connectivity, location capability, support for the paradigms of interaction, user interface and security issues. It should be also considered other issues, such as the balance between the use of reliable and secure real-time information with the ability to store that information on device.

The set of information that is necessary to interact with the client devices is vast. To a client computer the volume of information is not very important, depending only on Internet access speed, but for a mobile device that question is very different.

This is different in mobile devices, because their wireless capability can be slower compared with fixed or wired data connections. Besides, they often have a measurably higher latency. Many connection failures occur because of transmission interference (weather and terrain blockage) and noise. Thus, mobile data transfer often costs a lot of money compared to fixed data connections. Batteries still don't provide the desired amounts of energy without the need to be constantly recharged and wireless communication grows up the device power consumption.

To develop a client-server application for a mobile environment, all of this problems and limitations must be considered to provide an effective and reliable application.

1.1. Context and Objectives

This research work was developed in the context of the PSiS (Personalized Sightseeing Tours Recommendation System) project, developed in GECAD (Knowledge Engineering and Decision Support Group).

GECAD is a worldwide known research unit settled in the School of Engineering – Polytechnic of Porto (ISEP/IPP) with the mission of promoting and developing scientific research in the knowledge and decision support domains. The group slogan is "Intelligence for a Sustainable, Safe, and Inclusive World".

PSiS (Almeida, 2001) is a tour planning support, that aims to define and adapt a visit plan combining, in a tour, the most adequate tourism products, namely interesting places to visit, attractions, restaurants and accommodation, according to tourists' specific profile (which includes interests, personal values, wishes, constraints and disabilities) and available transportation modes between the

selected products. Functioning schedules also are considered as well as transportation schedules, to generate tour planning's when a tourist wants to visit a certain region.

The system gathers knowledge about the tourists' profile, creating groups and stereotypes with specific interests and features, allowing characteristics inheritance. A "tour box" stores tourists travel history, where all the places he visited are stored, which leads to accumulated knowledge about personal profiles. This knowledge, together with tourist stereotypes offer a mean of learning about general and specific tourists' interests, so this information can serve as a basis for studying new forms of tourist products.

It's also possible, for the tourist, to return information on accomplished tours. So, the system gathers knowledge about tourists' opinions and preferences. Based on this knowledge and on profile groups, categorized information can be delivered according to tourist specific interests, namely events, factual information, useful tips, promotional offers, recommended places to visit and more. PSiS intends to interact with the tourist through a web application and a mobile device (PSiS Mobile), providing both planning support and real-time assistance.

PSiS Mobile consists on developing a mobile tool to assist a tourist, on the "field", and it is integrated with the PSiS project. The main objective of PSiS Mobile is to help a tourist after he planning a trip for his vacations on PSiS Web portal, but it can also show a list of recommended sights to see on a specific location. The information taking into account are tourist preferences, past trips and users similarity (Personal Profile), his current context and nearby sights context.

In a preliminary analysis, the PSiS system is composed by two pieces: the server-side and the mobile client. The server-side is where all the main information is compiled. The mobile client is a very important piece in the system, because it is the "face" of the system. It is the platform that tourist will interact when he is on vacations.

PSiS Mobile objectives are:

- Transfer the original planning for a trip, from the Web Site to the mobile application;
- Do rescheduling of that planning, according to users' and point of interest context (open/close hour, weather, location and time);
- Recommend nearby sights to see, according to tourist profile;
- Retrieve user feedback from recommended sights, *e.g.*, user can rate a sight and give a comment to it; This is useful for future recommendations not only for this tourist, but for similar tourists;
- A tracking module, must be developed to track users' interactions with the system, and with the place, *e.g.*, how much time he spend at a point of interest;
- Develop a middleware, to exchange information between server and the mobile client, since it is used different platforms, on server-side is one and on mobile client is another. Also, this must be as fast as can be and imperceptible to the tourist, to enable a better user experience.

PSiS system in a preliminary phase is limited to data from the metropolitan area of Porto, Portugal. But it is designed to be integrated with all the places in the world, to avoid limitations on the amount of users that can use this system.

The work that must be developed to complete this thesis and how much time it can takes, is presented in this general task plan – table 1.

Table 1 - Task Plan

Task	Duration (months)
State of the Art	2
Research and Analysis of methods to exchange information between a client and a server	1
Implementation and developing 1. Study of the Android Platform 2. Recommendation module 3. Replanning module 4. Tracking module 5. Middleware 6. Reality View	5
Papers Writing (three papers on conferences and a paper on a magazine)	2
Dissertation Writing	2

1.2. Concepts and Evolution

Mobile devices are pocket-sized computing devices which popularity is growing day by day. Since they are small, simple and becoming relatively cheap, it's easy to carry them around and use them in all kinds of environments.

These devices have wireless capability that allows the device to connect to the internet even with low speed connections that networks on the go usually permit. They have high latency that can lead to long retrieval times, especially for lengthy content, but sometimes can be preferred to download one big file, than a lot of small files. Even so, this can lead to another problem, which is in the case that the connection is very unstable and always going down, so the large file can never be downloaded. The system must be effective at this point and be adapted to the context.

As the World Wide Web (WWW) evolved into an incredible huge mass of distributed information, recommendation systems emerged as an option to minimize the time consuming task of searching the Web. Although the concept is not new, the used techniques have been subject to research and evolved into different techniques.

The way the recommendation systems interact with the user is very important. Retrieving feedback from the user is time consuming, thus leading user to avoid submitting feedback forms. Transparency

might also be important to some users, as they want to know why they are receiving certain recommendations. The right transparency helps the recommendation system getting higher trust levels from its users.

The initial amount of data when the recommendation system first runs is also a big issue. Since recommendations are usually based on already existing data (*e.g.*, user profiles and choice history), systems need to tackle this issue so they don't suffer from the cold start problem (Luz *et al*., 2010).

So, because of mobile devices portability they are ideal to create an integrated system to assist the tourist in planning effectively a trip to an unknown city, and also to help the tourist when he arrives to the destination (country, city or region).

Tourists certainly appreciate to have a quite simple tool to assist them in planning their staying, according to their objectives, preferences, knowledge, budget and staying period, instead of having to look for guide prospects/bulletins which sometimes can be quite confuse (Almeida, 2009). This happens, since the number of tourism-related services in the Web grows every day offering hotels, flights, tickets and information of all sorts. So, where to go and what to do, with limited amount of time for choosing all the sights to visit, are common problems encountered by tourists when they decide to go on vacations.

A major issue in offering mobile services to nomadic users is the limited display size, resolution, power consumption, processing capabilities, low memory and networking capacity of mobile devices to communicate with a central service. When, it is necessary to plan a route or to show information about sights, the system could cover intelligent integration of information from different data sources and services, including geographical information systems, multimedia databases, and interactive internet data sources such as reservation systems. With a good networking capacity user experience can be improved.

A tourist that uses a system like this expects location-aware information about the destination domain including: history, culture, folk, art, economics, environment and nature. Advanced tourists also expect individualized information and services taking into account their own interests and their travel history (Poslad *et al*., 2001).

To provide a good mobile tourism service there are a lot of technologies available to use. A tourist with his mobile device (PDA, Personal Digital Assistance) or with a specific tourism guide mobile system can retrieve information via SMS (Short Message Service), MMS (Multimedia Messaging Service) or WAP (Wireless Application Protocol)/Web. A mobile device is the most reliable and user friendly option (because tourist usually already have one, so they don´t need to buy another device), but it needs a communication channel to retrieve data for the user.

Over the last years, these technologies are more advanced and the data transfer rates are rapidly increasing. GPRS (General Packet Radio Services) was the first to appear and has a low data transfer rate of approximately 40 kbps. In the year of 2000, UMTS (Universal Mobile Telecommunications

System) appears with a data rate of approximately 220 kbps, finally providing good enough speed to support a system like this and to confirm the capabilities of this technology. The UMTS is a worldwide service that makes it a very important factor when implementing mobile systems.

After UMTS, the HSDPA (High-Speed Downlink Packet Access) appears with a typical performance of 750 kbps. Although the data transfer rates are higher, the coverage area is smaller. These technologies are used in a global scope, but there are also indoor specific technologies (*e.g.,* Wi-Fi and Bluetooth) that usually outperform them. Although they have better performance, they aren't covering a large area, but nowadays many mobile carriers are quickly adding high-speed network capacity, in the cities, in the form of Wi-Fi. With this it is easier to connect to wireless hot spots with mobile phones, in an effort to deliver fast data and clear calls in areas where neither might be possible. Apart from mobile carriers, there are also some city councils implementing Wi-Fi system throughout their territory, it permits to mobile device users, mainly tourists, to access to free internet. With this, one of the constraints of mobile devices is partially solved. Only partially, since Wi-Fi is only implemented in a few big cities, and not on more rural environments where the HSDPA or UMTS are normally unavailable. Another problem is the data plans from carriers, that for example in Portugal, the coverage of these networks is around 99% of national territory but, the price per MB (Megabyte) is very expensive and there isn't any unlimited data plan from mobile carriers. Besides being expensive and not available in certain areas they have poor reliability that inhibits (sometimes, to the point of infeasibility) supporting applications that involve efficient access to WWW and Web Services.

Mobile client solutions, on client-server applications, can be built in many different ways, used on many different devices that operate over many different networks and integrate with many different back-end systems. The task of building a mobile solution can often be daunting given the many technology choices and implementation approaches. Unlike centralized systems, client/server systems that use a mixed bag of LAN-connected (Local Area Network) hardware need software optimization. These optimizations lead to less network traffic consumption in communications between clients and servers, and also to less monetary costs, less battery consumption and better performance.

In order to suppress all the constraints that mobile devices have, there must be paid attention to certain details. So, to create a mobile application there are several specific challenges like (Schuler, 2007):

- Interfacing Disparate Technologies - Mobile system application development often involves using different technologies due to platform restrictions. If there are two different technologies like Microsoft and Java, they must communicate seamlessly with each other. Web services, HTTP (Hypertext Transfer Protocol), and TCP (Transmission Control Protocol) sockets are used to bridge these gaps;
- User Interface Design - Designing the Graphical User Interface (GUI) on a mobile device can be challenging because of the small screen and the difficult data entry interface. If the application or data is complex, the user will need to interact with many screen objects such as entry fields, lists and radio buttons. Complex screens will need to be divided into separate

screens or tabbed interfaces. A wizard-like interface may be appropriate for some applications. Some applications, on pen-based devices, may require the use of the device's physical keys instead of the stylus. If a lot of free-form data entry is required then a tablet or notebook PC (Personal Computer) should be considered;

- Performance - Servers and desktop computers have progressed significantly and performance is typically not an issue anymore. However handheld computing devices are another story. Many are very slow by comparison with servers and desktop computers. Complex user interfaces, CPU (Central Processing Unit) intensive algorithms and data processing can easily make an application user-hostile;
- Memory Management - Although memories are cheaper than ever, most mobile devices come with a few amount of memory and cannot be upgraded. If systems analysis shows that data requirements include having large amounts of data on the handheld device, this may limit hardware choices. Efficient data storage is necessary and low-level interfaces may be required to make the most of the memory available. Because cold boots typically erase all non-volatile memory in the device, design must ensure that critical data is stored in non-volatile memory;
- Security - Security is a concern in many systems and mobile systems are no different. Mobile systems introduce a few new issues. What if the mobile device is lost? A stranger cannot be allowed access to sensitive business data. Some hardware includes thumbprint scanners to authenticate users. A user login may also be implemented so that users must present a set of credentials before application use.

Besides all the previously enumerated challenges, when selecting the platform for the device, three different types of applications can exist (Hariharan, 2008):

- Online Applications (also known as a thin client). This is client software, normally a browser, used when connectivity can be guaranteed. Without a connection, the mobile application does not work;
- Offline Applications (also known as a thick client). This is client software installed locally to the device that holds all required data for the duration of most operations and synchronizes at the end of each day or a preconfigured period of time;
- Occasionally Connected Applications (also known as a smart client). This is client software installed locally, similar to the offline model, but where the application can update and refresh data at any point in time. The frequency of the data refresh depends on the criticality of the application.

Summarizing, the technological choice for programming a distributed application has different steps: First, there must be chosen the Application framework/OS (Operating System); Second, what computational resources, from the mobile device, are needed; On third place, what distributed programming paradigm and environment will be used; And finally, what security mechanism will be used.

1.3. Thesis Overview

This thesis is organized in the following manner:

Chapter 1 presents the main objectives and motivations behind this research work. This chapter also gives a brief description of this work development context and organization.

Chapter 2 introduces a state of the art on actual mobile tourism guides. These systems are divided into two groups: Mobile Information Guides and Mobile Recommendation Systems. The first group includes all the systems that have only capabilities to show information about points of interest and the second one includes all the systems that have capability to show and recommend sights to visit. Also a comparison between all these systems is realized.

Chapter 3 presents a succinct study of mobile systems and their operating systems. Android, which is a mobile operating system, architecture is described.

Chapter 4 provides a brief introduction to PSiS system and how PSiS Mobile integrates with it, presenting the complete system architecture. Will be discussed the reasons and the importance of a mobile system to support the central one. Current mobile limitations will be presented as well.

Chapter 5 presents a case study which assesses the fastest and most reliable way of exchanging information between the server-side and the mobile system.

Chapter 6 presents the implementation details of PSiS Mobile, where the product of this work is put into practice.

Finally, chapter 7 contains conclusions, future work based on current limitations and new features that can be developed to integrate with actual system.

Chapter 2

State of the Art

In this chapter will be introduced some of the existing systems that supports a tourist on the go. First of all, will be presented four mobile information guides. These systems provide important services to guide the tourist along its travel. Secondly, four recommendation systems are approached. This type of systems allows the tourist to plan, select an appropriate route with base on a set of points of interest. Although these systems can be (and should be) integrated, very few approaches integrate both systems.

To improve user interaction, most of this systems support context-awareness, this can help by knowing a-priori the users' situation, personal preferences, information interests and environment conditions, so that the user doesn't have to specify these constraints and information delivery is automatically adapted to his circumstances. For example, if a context-aware system knows that an user is driving a car, it can adapt alerts to audio messages without requiring the user to input details about his current situation.

It is argued that location-based systems must allow users to participate as content providers in order to achieve a social and dynamic information space. Moreover, as these systems allow commercial and private users to annotate space with information on a mass-scale, information filtering techniques will become essential in order to prevent information overload and user disturbance.

This investigation aims to examine the employed techniques and their functions provided to the users, as well how it works, *e.g.*, if it is a server-client application or a standalone application.

The chapter will then end by presenting a conclusion/comparison between all the systems talked before, taking in account six requirements which are key factors to build a mobile tourist support system.

2.1. Mobile Information Guides

More and more people combine several purposes with travelling, such as business, leisure, entertainment and education. Such people may not have time to pre-plan a travel schedule in detail. They need location-aware information about the destination domain and expect individualized information and services. Mobile tour guides are the result of years of research in the areas of recommenders, ambient intelligence and pervasive computing.

The basic idea is to connect pieces of digital information to a specific latitude-longitude coordinate via some mobile device, thereby "attaching" them to a specific place in space. Later, users, again via some mobile client, can access that information. In this way, users will get the impression that the digital information is actually attached to a place in a way similar to post-its, graffiti, public signs and posters. There are systems that only display information about sights and other more complexes, which can share information between users.

2.1.1. MultiMundus

MultiMundus (Kropfberger *et al*., 2007) is a Web-based guidance system which supports optimized presentations of sights or exhibited objects on different types of available stationary and mobile consumer devices, possibly running different operating systems. This is accomplished by adapting both the objects' content (Video, Audio, Text and Images), as well as their presentation to the current usage context (Location, User Profile and Time). User profiles are divided into some defined categories: adults, children, historical interest and technical interest. So, the content is displayed according to the association between the content and user profile. This association is fed into the system via a Web-based and hereby platform-independent Content Management System (CMS).

To know user position the system uses two different types of technologies: GPS (Global Positioning System) and Bluetooth. In the second case, it is necessary to have a structured environment. To download data is used UMTS or Wi-Fi network.

Besides CMS, the system is composed by a content repository, a module to do live-streaming of video or audio and a statistics module. This last module logs complete user sessions and allows daily, weekly, or monthly statistics, *e.g.*, favorite/average content consumption, mainly used languages and profiles, average sessions duration and stopover times at certain points of interest. This knowledge can easily be feedback into the adaptation system to offer improved content and presentation optimizations, and to give suggestions to the users.

MultiMundus is designed to have easy customization to different application areas like museums, fairs, theme parks, touristy regions or even shopping centers. It has already a first practical evaluation, at an

outdoor theme park in Klagenfurt (Austria), in 2005. The park was covered by a number of WLAN (Wireless Local Area Network) antennas, so rentable or private mobile devices can reach the main server. Ten rentable devices were available to use for a small fee.

Summarizing the usage evaluation results, such a location-aware and multimedia-enhanced guidance system is rather a high-end solution for some interested visitors, who want to consume more information than provided by commonly available catalogues and brochures. But, there are two more crucial problems related with mobile devices usage. First, the battery life cycles were partially really short (had to be recharged three times per day). Many of the used batteries were dead at the end of the season. And second, the tourists who handed out the devices have occasionally broken the devices.

2.1.2. GeoNotes

GeoNotes (Espinoza *et al*., 2001) system tries to blur the boundary between physical and digital space (ubiquitous computing and augmented reality) and at the same time strives to socially enhance digital space (collaborative filtering, social navigation, etc.) by allowing users to participate in the creation of the information space. It is a location-based information system that allows user to access information in relation to his position on geographical space. The main goal of this project is to provide location-based information free to all users.

With this mobile application, tourists can retrieve notes regarding their current location. These notes are introduced by other tourists that have visited the same place and have created their own notes. To create a note, the sender needs to specify four components: a title, the recipient to whom the note is directed (to all people or only to a friend, to keep privacy), the author and a placement label (this 'tag' christens the place or object in which the note is intended to be posted).

These notes are retrieved according to user current location and using content-based filters, that filter notes based on the user's interest and the content of the notes. In such filtering technique, the user expresses his interest with a set of keywords of his own choice. Here can be used more or less sophisticated combinatorial and Boolean search methods. As a user moves through space, the system constantly scans available notes at his location and rates higher notes that match the users' keywords. This makes it more likely (or even certain, depending on the users' preferences) that some notes will be pushed to the user. As the interests of the user changes, he should be able to add or delete keywords. In addition to this, the system also collect usage data about how users read, click, save, choose, comment etc. notes in the system. Such actions can in different degrees be seen as voting for any given piece of information, which can then be used to rank notes. For instance, once the user has read a note, this note should probably be ranked lower next time the user enters the same location.

Clients, which have been developed in Java language, connect to the server using RMI (Remote Method Invocation). The server uses a socket connection to store content data about notes in the

database and SQL (Structured Query Language) commands to store metadata about the usage of notes in the MySQL database (figure 1).

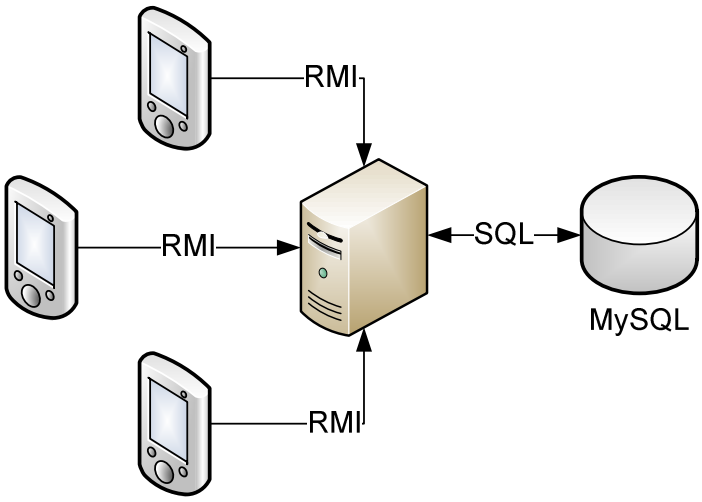


Figure 1 - GeoNote System Architecture

2.1.3. MobiDENK

The personal mobile assistant mobiDENK (Krosche *et al*., 2004) has been developed to assist tourists on a tour to the Herrenhausen Gardens, in Hanover (Germany), and includes points of interest on which historical information and images of the most significant features are presented on a PDA. It focuses users' attention to historic sites, provides location-based multimedia information at the different sightseeing spots and displays the person current location on a map.

AccessSights (Klante *et al*., 2004) is a subproject of mobiDENK and is intended to provide tourist information to both normally sighted users and visually impaired people traveling in the Gardens. The user receives visual and audio information. Normally, sighted users will make use of both senses to obtain information and may simply follow a guide map, while blind people listen to information. The system uses loudness in order to point out the distance between the users' current location and the points of interest, by simply making the voice signal get louder as the user comes closer to the point of interest. A mobile chase game is included and realizes a location-aware game that allows its players to find a set of geo-referenced checkpoints and solves the associated hypermedia riddles. The checkpoints are proximity-aware, exploiting the players' location. After the riddle is solved, the player physically moves on to another checkpoint indicated on the map. The paper chase game has also been enhanced by auditory to support the players with weakly intrusive navigation and orientation support.

2.1.4. MacauMap

MacauMap (Biuk-aghai, 2004) is a tourism-oriented map application for the Macau territory. This isn't a client-server application, because all the data is stored on the downloaded application. Within the

first year of its release, it has been downloaded nearly 100,000 times. Thus the most important feature of MacauMap is the display of a map containing only essential features of the territory. In the case of Macau, most of which is a densely built-up urban environment, this consists of: streets, represented as centerlines and including their street name; green areas (forested areas, parks); water areas (lakes, sea); hotels (including all 3, 4, 5 star hotels in Macau); restaurants (including all those registered with the Macau Government Tourist Office); tourist spots (museums, churches, temples, gardens, parks, scenic spots and tourist information offices); and public bus stops.

Besides basic map information, MacauMap includes a searchable database of such places, including hotels, restaurants and other tourist spots. These are categorized to easy retrieval of the relevant information. For each tourist place, detailed information such as address, telephone number, description, opening hours and admission fees (when applicable) are included.

MacauMap has a bus route calculator to help tourists' find out how to travel from one bus stop to another. User selects a start and destination bus stop, and MacauMap calculates the shortest bus route between this pair of bus stops and displays the route in the map.

MacauMap includes a function that allows the user to bookmark their favorite places in Macau which may not already be in MacauMap database. Users select the location in the map where the spot is located, add a name and a short description, also is given the opportunity to take a digital photo of the place using mobile phones' built-in digital camera (if any).

2.2. Mobile Recommendation Systems

As the World Wide Web evolved into an incredible huge mass of distributed information, recommendation systems emerged as an option to minimize the time consuming task of searching the Web. Although the concept is not new, the classical pure techniques have been subject to great research efforts and evolved into different hybrid techniques.

The initial amount of data when the recommendation system first runs is also a big issue. Since recommendations are usually based on already existing data (*e.g.*, user profiles and choice history), systems need to tackle this issue so they don't suffer from the cold start problem. Latest implementations began to use ontology's, thus empowering the recommendation system with rich semantics.

Although many derived approaches are emerging, recommendation systems are mostly based in three different paradigms: content-based, collaborative and knowledge-based. The content-based paradigm applies to systems that rely on item information to retrieve recommendations. This means that item attributes and ratings are used to see what best fits the user needs. On the other hand, collaborative systems compare similar users to provide recommendations. The knowledge-based paradigm tends to tackle the content-based and collaborative systems weaknesses and problems. Through the use of ontology's (or case-based rules) a reasoning process is performed, allowing the user to incrementally specify his needs, thus improving the recommendation results. Since pure

recommendation systems contain multiple weaknesses that can usually be tackled by merging different paradigms, hybrid systems have become the current popular choice that shines especially when the system needs to deal with highly heterogeneous information. A hybrid approach can involve all the three recommendation paradigms (Luz *et al.*, 2010).

The described systems implement profile learning techniques which in mobile tourist systems enrich substantially the tourist experiences.

2.2.1. Tourism Information Provider

The Tourist Information Provider (TIP) system (Hinze and Buchanan, 2005) is a combination of an Event Notification Service (ENS), a Location-Based Service (LBS) and a context-aware information delivery service. Delivering different types of information based on user interests, their travel routes and sight-related information is the main focus of the system. The users dynamically get information from the system via their handheld devices such as mobile phones or PDA. At the beginning, the users are required to define their preferences to the system as user profiles. In the user profile, the system keeps the user information, for instance, the type of sights they are interested (*e.g.*, cathedral and churches) as well as the type of information the users are interested (*e.g.*, history of the sights or a sights' architecture). This information can be changed whenever the users want to revise their interests. The information about users' current location, their already visited sights and sights' locations are considered before any information is given to the users.

Apart from giving the users information about the place they are visiting, the system also supports recommendations about places to go to or interesting activities to do for a particular user.

The current recommendation component in the TIP system considers the following five factors:

- User Profile - System learns users' preferences from the given information and provides recommendations based on the acquired knowledge about the user;
- User Context - It uses users' location and time to determinate the suitability to visit a sight, because its distance and opening and/or closing time;
- Sight Context - Context of sights also covers their location, operating hours and weather conditions. Recommendations might be given on the assumption that users who have visited several sights of a group might be interested in seeing more sights of this group;
- Users' Travel History – Users' travel history includes places, time and location that the user has been to. This information is a track of the user movements therefore the system will not recommend places that user has already visited. The system learn user preferences from what users did in the past and predict what they would like to visit or do in the future;
- Users' Similarity - Sights which other similar users liked may be recommended to the user. The user receives recommendations based on the similarity of sights to other sights, that this user gave positive feedback.

TIP system implements hybrid algorithms to give recommendation to the tourist. It uses four approaches:

- Collaborative Filtering - In this approach, users' feedbacks, which are known as ratings for the sights they have visited, are collected. Other users, who share the opinions or have the same taste, can use this user feedback to better decide which sights to go. The approach uses two steps: identify similar users and then recommend sights that similar users liked. Recommendation component filters sight information using the user id and his current location;
- Compound Approach - Recommend nearby sights which match a user profile. The user is required to define his interests when registers' on the system. Nearby sights which are of the same types as defined in the current users' profile and haven't been visited by the user are recommended;
- Extended Content-based Approach - This approach take all requirements (as described above) into account for giving recommendation. Other nearby sights, which have the same type as the sights that the current user has visited and given high feedback scores, are recommended;
- Pure Approaches - Recommend sights that are semantically-related to the sights that the user has visited in the past. Recommend nearby sights based on user context, sight context and user past visits information. Sights which are located near the current users' position and the user hasn't yet visited are recommended.

In summary, though the four recommendation methods use all the identified system requirements, they still have some drawbacks which are challenging to manage. This is because the practicality of each recommendation approach strongly depends on the availability of the system requirements. For instance, option one of the implemented approaches - the collaborative filtering method - relies heavily on users' feedback while option two and four strongly depends on user profile and user feedback. Therefore, the system uses a combination of system requirements and methodologies in order to provide a more effective recommendation.

This system is implemented using a central database approach. A PostgreSQL object-relational database management system and a Java Servlet Technology are used. The recommendation component has been implemented in Java.

2.2.2. Proximo

Proximo (Parle and Quigley, 2006) consists of a PC-based recommendation system using sample paintings and a rating system in addition to an application running on a Java-enabled Bluetooth mobile phone. It is a location-aware recommendation-driven system for use within indoor environments such as museums and art galleries.

The indoor positioning works by “sniffing out” the fixed Bluetooth devices or low-cost beacons deployed in the area of use. This room-level accuracy means improved precision and is accurate enough for this type of application. The mapping service, on the mobile handset, displays a map of the intended area of use. Items provided by the recommendation system are displayed as icons over the map of the active area on the mobile application. This provides both location information and an interface for the user to select between the different items. The user can scroll between the icons to view information about the corresponding item.

The recommendation system collects and stores data from multiple user interactions that is used to create user profiles based on a weighted nearest neighbors algorithm. The items with the highest predicted ratings are those which will be recommended to the user. Once the user has given an initial rating, the location and other domain-specific information relevant to these items are transferred to the handset-resident application. The mobile application constantly monitors the users location and displays the active areas of the building (the area they are in) accordingly. The paintings on the tour are also displayed in a different color that highlights them from the others.

When a painting is selected there are a number of actions which a user can take. There is a facility for the user to provide a rating for the painting. It have a messaging feature similar to GeoNotes where messages can be left at a specific location and can only be accessed from that location within a certain context.

Parle and Quigley conducted tests under a gallery context at the University College Dublin School of Computer Sciences and Informatics building, where the system was able to guide users through a gallery and provide recommendations using similar user ratings to paintings. Users were asked to fully complete their tour of the Gallery and to answer to a questionnaire. They were also asked to give a rating for each painting on their tour and to write at least three messages themselves (about anything they liked) whilst on their tour. The last two tasks were to be completed using the application running on the mobile phone.

The recommendation system was tested by examining the predicted rating and actual rating for the paintings on each user’s tour. The absolute difference or error between these two values is used to measure the success of the predictions. The overall success of the system was calculated by finding the mean absolute error over all the predictions made. In this case the mean error was around 20%.

2.2.3. m-ToGuide

m-ToGuide (Schneider and Schröder, 2003) is a project sponsored by the Information Society Technologies (IST) Fifth Framework Program of the European Commission. The project is targeted for the European tourism market and offers to tourists a broad array of information and services. A portable, handheld terminal is used to exchange information between the m-ToGuide central system and the tourist. GSM (Global System for Mobile Communications)/GPRS cellular telephone networks provide the transmission backbone of the system. All information and services delivered to the tourist will be relevant to his specific location (location-based) and tailored to that end-users’ personal profile.

The m-ToGuide consortium consists of seventeen partners from six different countries (United Kingdom, Germany, Italy, Spain, Sweden and Israel). All of these partners are technology suppliers or tourism service providers. Motorola's research and development division, headquartered in Israel, was the responsible for the project and have assembled the m-ToGuide group.

This service consists on providing a device to tourist that serves as a mobile personal guide. Project objectives are the follows:

- Implement this service on 2.5G (GPRS) in a way that enables compatibility with 3G (UMTS);
- Provide users with a roaming enabled data service;
- Provide users with the capability to customize the service according to user defined profiles, having a profiling database and tracking users' preferences, to support decision making;
- Provide various types of content (maps, text, audio, and visual materials);
- Offer a Smart Wizard tool that produces a optimized tour and makes the real time decisions;
- Offer m-Commerce capabilities;
- Incorporate security, billing and privacy into the system;
- Consume external services to provide booking and ticketing;
- Deploy the trial in three different European countries representing three different cultural environments and touring patterns;
- Provide multilingual content in English, German, Italian, Spanish and French;
- Achieve a level of 99% in Quality of Service (QoS).

The applicative objective is to integrate a multidisciplinary mobile location based tourist service that includes an intelligent decision wizard to provide filtered information from multiple content providers.

The business objective, of the project, is to create an economically sustainable model of value flow that appeals to end users as well as to tourist service providers.

m-ToGuide is a dedicated device to assist tourists on the field, being like a normal PDA. This terminal provides the end user with relevant information and gets inputs from him. The terminal uses GPS in order to get the tourist location and GPRS modem in order to communicate with the system.

The communication between the terminal and the system is done trough the Smart Wizard (see figure 2). The Smart Wizard is the central piece of m-ToGuide system because it is responsible for providing responses to each request from the terminal, *e.g.*, maps, information, search results, tours, routs, etc. and the communication between the different subsystems.

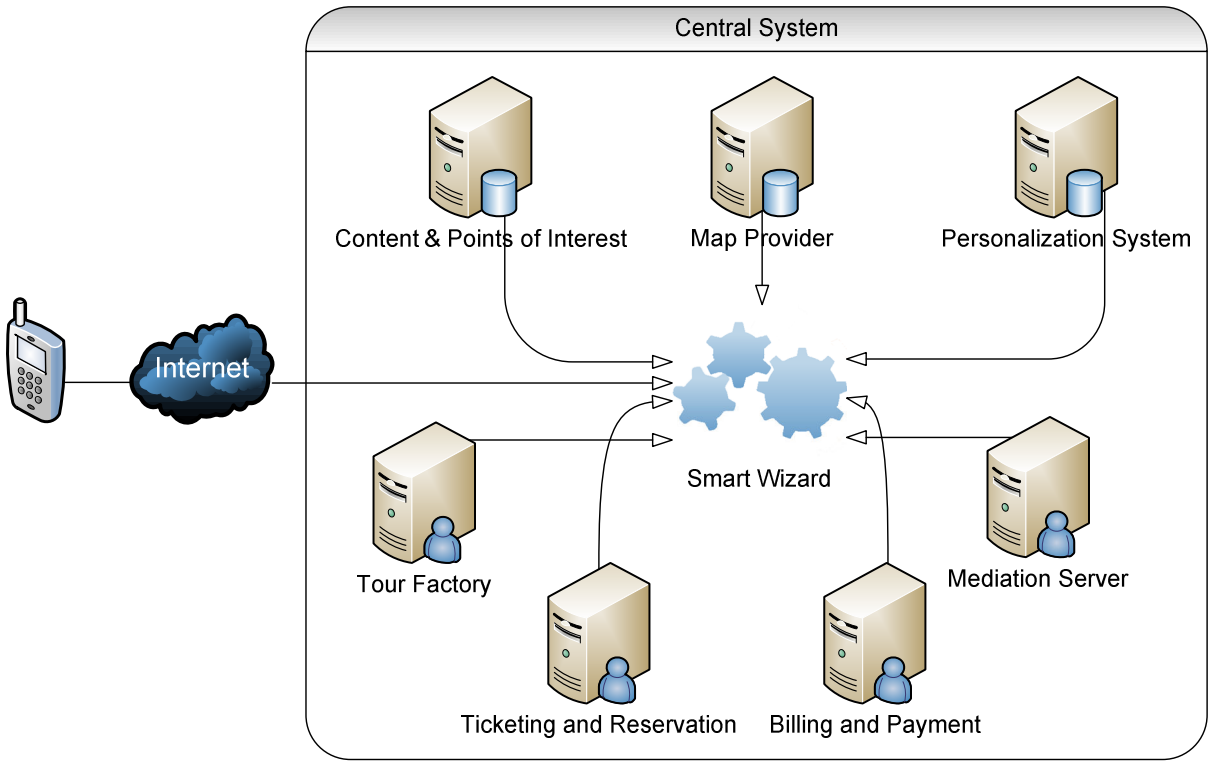


Figure 2 - m-ToGuide System Architecture

For maps and content, Smart Wizard asks the Mediation Server that is connected with the mapping engine and with the content providers. Then Mediation Server responses with the relevant information (maps, routs and content) to the Smart Wizard. If information related to the user profile is required, the Smart Wizard will ask the Personalization System for the user profile and accordingly will retrieve the data. Each selection and response at the user side is transmitted to the Smart Wizard that is teaching the Personalization System. Billing and payment system is used in order to provide billing services and to allow payment. Ticketing and reservation system enable the user to purchase tickets.

m-ToGuide is seen as most useful in larger cities, where there is a vast quantity of information and activities to choose from. In addition, m-ToGuide is also useful for "last minute" trips, and requires little or no planning in order to pick up a terminal and go. It is divided into three main usage scenarios:

- Multi-day Use, tourists planning a visit of more than one day to a city, purchase the m-ToGuide from their travel agents prior to departing, or from their hotels upon arrival. m-ToGuide service is pre-paid as part of the holiday package, or on leaving as part of the hotel bill. The audience for the multi-day use are primarily tourists, including family groups;
- Single-day Use, for tourists visiting a city for just one day. They rent m-ToGuide service from a tourist information office. Because battery life is sufficient for one day, no charging unit is required. The audience for single-day use are business travelers as well as tourists, including family groups;
- Hour/Single Tour Use, tourists with less time or interest in a full day of m-ToGuide services can rent m-ToGuide terminal at any of visitor attractions in the city. The audience for this case is business travelers as well as tourists, including family groups.

A trial project was performed and tested with an on-the-go ticketing facility that allows tourists to make bookings and reservations directly from the terminal via GPRS. The trial results indicated that the system was useful but the charged prices weren't well accepted by users.

2.2.4. Deep Map

Deep Map (Malaka and Zipf, 2000) realizes the vision of a future tourist guidance system that works as a mobile guide and as a web-based planning tool. Deep Map is a mobile system able to generate personal guided walks for tourists through the city of Heidelberg, in Germany, and to aid tourists in navigating through the city. Such a tour considers personal interests and needs, social and cultural backgrounds (*e.g.*, age, education and gender), type of transport (*e.g.*, car, foot, bike or wheelchair) as well as other circumstances (from season, weather, traffic conditions to time and financial resources). The core of Deep Map is a typical Geographical Information System (GIS).

The system can also handle spatial and topological queries while allowing navigation and route finding. Touristic information is location-dependent by nature. Each sight, building, hotel, restaurant, etc. have a spatial location. Moreover, during mobile use, tourist location is known and one important task is to relate the tourist's location to attractions in reach, as well as the tour he wants to take and to the goal he wants to reach. To respond to many questions about location, *e.g.*, Where I am, How do I get from A to B, etc., that need geographical knowledge must be managed by a GIS. To answer these questions, the system relates information from databases and other resources such as restaurant guides to the GIS. Of particular interest for the tourist applications are questions that relate to historical and temporal changes. It also provides temporal databases.

Therefore Deep Map is taking a step further, making use of the so-called agent-oriented software architecture. The agent based approach allows an easy re-use of components in different systems that may consist of a different set of agents and thus providing another range of services. This is especially important in this scenario where there are two quite different application platforms: a Web-based system for home users and the mobile system for tourists on site.

The GIS and databases are accessed through the following agents:

- Database agent retrieves non-spatial information from the database;
- Geo-spatial agent retrieves spatial information from the GIS and performs a range of geo-spatial computations on that information;
- Route agent computes and manages routes and their segments;
- Map agent generates and handles maps and their visualization.

These agents have been developed using the ArcView GIS as server platform. The agents themselves communicate via a message bus (Java Agent Management framework - JAMFrame). GIS agents are implemented using Java wrappers that communicate with the ArcView GIS, running on a specified server, via RPC (Remote Procedure Call). Since on the one side, only string messages can be passed via RPC and on the other side, messages on the bus are represented as Java objects,

these objects are converted from/to text. For this reason a XML (Extensible Markup Language) notation is used that allows the automatic conversion of Java objects from/to XML using the Java reflection API (Application Programming Interface).

Data is stored in an "event" based data model for historical geo-referenced data. This relational data model consists in relations of different types between locations (several types of spatial objects and their respective subclasses), persons (historic, real and legal persons) and historic events of different granularities. Media types and relationships between these multimedia documents with persons, places and events are geo-referenced through the coupling of the locations in the database with the GIS.

Map-Agent is a Java based Deep Map module that gets the geometric data of the spatial features that have to be displayed from a geo-server and renders this vector data on client side. Normal maps just contain 2D information, but Map-Agent includes 3D information to generate route instructions that do not sound as:

“go 205.4 meters straight, turn 30 degrees to the right and go 67.9 meters straight,”

but rather like,

“follow the street and turn right after the big red building and head towards the church.”.

The vision of Deep Map wants to allow the user to get personalized and easy access to a variety of information. In the future virtual tours in a 3D-reconstructed city will be possible. Because many types of data aren't only spatial but also temporal, *e.g.*, environmental, climate, or city development data, Deep Map handles 4D data, facing questions of tourists standing in front of a historical place like a ruin of a castle asking "how did that look like when it was not destroyed?". In this case tourists’ would like to turn back in time and on mobile device go through a virtual time travel, being displayed a reconstruction of that place as a virtual model.

Deep Map appears as a system with different faces. One version is a Web-based planning and exploring tool for virtual visits and pre-trip planning. The current mobile version is realized as a prototype that can be used for a limited area around Heidelberg castle.

2.2.5. CATIS

CATIS (Pashtan *et al*., 2003) is a context-aware tourist information system with a Web service-based architecture. The context elements considered to this project are location, time of day, speed, direction of travel, personal preferences and device type. This system will track users' location providing the user with information relevant to his location and time of day. For example, if the user is traveling at noon, a simple integration of the time context, the location and respective user preferences for restaurants will result on a list with restaurants to lunch. This is the primary goal of this project, user context, to simplify all the user-system interactions.

To do this, the system implements three adaption capabilities. First is the location and time-based adaption. With this information, the system will track users' position and time to provide them information relevant to that location and time of day. Second, is personal adaptation, the application provides services adapted to users specific profile (*e.g.*, users want to receive information about resources that match their personal tastes such as nearby restaurants that offer their preferred cuisine). At last, there is device adaption, knowing device features, the sent information will be more adapted for that device, for example, the device has a certain screen size and may not support certain image formats.

The architecture of this system is constituted by a thin client device that hosts a Web Browser (CATIS isn't a mobile application installed on a mobile device, it works via a browser), an application server that delivers web content customized to the users' context, an UDDI (Universal Description, Discovery and Integration) services directory that provides users with a centralized registry of tourist information services (*e.g.*, a restaurant finder service). A context manager is present to track the user dynamic context as well as user preferences. CATIS main goal is the ability to run on a lot of platforms, because of its web service-based architecture it uses a collection of Web Services to retrieve information.

On the first use of the system, the user must enter his preferences to complete his profile. After that, the system will manage his tracking with the GPS receiver, if it's available. If not, it will ask the users' current position.

When a tourist needs information and interrogates the system about, for example, restaurants in his vicinity, the system works in 5 steps (as shown on figure 3). The mobile device requests the information to the Application Server, which will query the Context Manager for user context information, such as location and restaurant preferences. It then sends an inquiry to the UDDI Server to get the addresses of the available restaurant Web Services. Then, Application Server sends a request to all the Web Services along with the users' location and desired distance from the user. Web Services search their databases for the appropriate addresses and filter out those that are too far away. The Web Services return a XML list to the Application Server. Finally, the Application Server filters the XML documents according to user preferences, prepares the presentation of the information and sends it to the client.

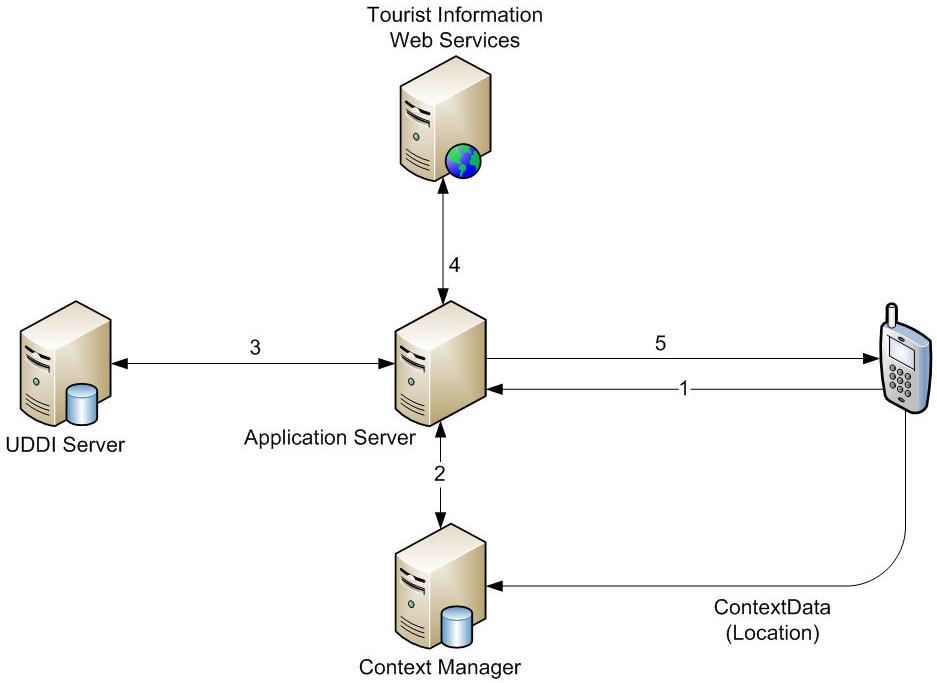


Figure 3 - CATIS System Architecture

This system is, nowadays, implementing only two dynamic context variables, namely time and location (in addition to static user preferences). In the future, it is planned to complete the other dynamic context variables and to investigate issues regarding the system scalability.

2.2.6. Systems Comparison

Although all the described systems are designed to help tourists on a trip, there are factors that distinguish them. To make a summary of the main features of the studied systems, a table with features that were considered to be important in this type of systems is presented, which were:

- Recommend Personalized Tour, capability to recommend a personalized tour, based on the tourist profile;
- Sightseeing Guide Information, ability to provide information about a specific sight;
- Personalized Information Delivery, capability to store user profiles and perform information filtering according to those profiles, thus supporting personalized information delivery;
- Booking, functionality to book a restaurant, a hotel and so on;
- Personalized Recommendations Based on Travel History, states if the system should recommend sights that not only match the user profile, but also their travel history;
- System Domain, describes if the system is ready to work in any place of the world and not only in a specific location.

Table 2 - Mobile Tourist Guides Comparison (Anacleto *et al.*, 2010a)

	Recommend Personalized Tour	Sightseeing Guide Information	Personalized Information Delivery	Booking	Recommend Based on Travel History	System Domain
MultiMundus		√	√			
GeoNotes		√	√			√
MobiDENK		√				
MacauMap		√				
DeepMap	√	√	√		√	
Proximo	√	√	√		√	
CATIS	√	√	√		√	√
mToGuide	√	√	√	√		√
TIP	√	√	√		√	√

Analyzing table 2, can be concluded that all of them have guiding capabilities but, none of them offer all the described features together. However, MacauMap has a very practical and important feature, which is the ability to help tourists find out how to travel from one to another location using public transports. Also GeoNotes is a very simple application because users construct the information about points of interest. But this has a negative point that is the trust of the information given by the community. (Anacleto *et al.*, 2010a)

MultiMundus and Proximo are the only systems that need a structured environment to work. They work inside museums and the location is retrieved using Bluetooth beacons or Wi-Fi antennas. This limits the functionality of the system, in order to work in other environments.

MobiDENK have an important feature that isn't present in the others systems, that is the adaptability of the system to blind people. But, it has been developed only for a tour to the Herrenhausen Gardens in Hanover.

Unlike MultiMundus, that have predefined user profiles to user select, TIP, Proximo, m-ToGuide and CATIS have capabilities to adjust users profile according to their tastes, actions made on the system and users similarity.

m-ToGuide, MultiMundus and Proximo systems were carried out tests, where some of them wasn't so positive. In the case of m-ToGuide, tourists liked to use the equipment and found it useful but complained about the price charged to use it. On the MultiMundus case, it has great acceptance by the tourists, but the equipment suffered some wear due to the intensive use.

The most complete systems are CATIS and TIP. Both of them can be improved, with some features like booking, augmented reality with 4D representations of sights, like DeepMap, and a module to help people with handicaps.

One aspect that is also very important focuses over the capabilities of mobile device, that are very limited, and that none of the described systems take as a main concern. CATIS works on every mobile platform, because it's a web-service based system. On the other hand, the use of Web Services in a mobile phone is time-consuming (parsing HTTP plus a SOAP (Simple Object Access Protocol) header is a very expensive processing task to a mobile device) and imposes some restrictions, so it's preferable to use a standalone application, even though it forces the creation of different applications to every mobile operating system.

Chapter 3
Mobile Operating Systems

A smartphone is a mobile phone that offers more advanced computing ability and connectivity than a contemporary basic phone. They may be thought as handheld computers integrated within a mobile telephone. A smartphone allows user to install and run applications based on a specific platform. It runs complete operating system software providing a platform for application developers. The first smartphone was released to the public in 1992. It had no physical buttons, but had a touch-screen (Schneidawind, 1992).

The Symbian OS was launched by Ericsson in 1996, and today is one of the most used mobile operating system, principally by Nokia. Microsoft had responded with Windows CE (Compact Edition) in 2001, which served as the basis for the Windows Mobile OS shipping with some smartphones today. However, in terms of market share, Windows Mobile has been in steady decline. In 2008, its market share dropped to 14% - down from 23% in 2004 (Chen, 2009). Until 2007 it has a big part of the market together with Symbian and RIM (Research In Motion, Blackberry's Operating System). But it is leaving behind, mainly because its user interface that is very complicate to use compared to the new players. Also new versions of operating system aren't coming out, so frequently.

In 2007, Apple introduced their first iPhone with iOS. It was initially expensive costing, though it had a full touch screen, a large finger-pressable icons and a very user-friendly interface which was revolutionary at the time. It also was the first mobile phone to contain a usable, fast and user-friendly web browser. This operating system has revolutionized the market, with sales having an unexpected growth. But, the only phones that can have this OS are the mobile devices sold by Apple.

Because nobody wants to be left behind, in response to iOS arises Android OS, in 2008. Android is an operating system, based on Linux kernel, designed for mobile devices such as cellular phones, tablet

computers and netbooks. It was initially developed by Android Inc. (a firm later purchased by Google) and lately broadened to the Open Handset Alliance. This consortium is composed by 71 hardware, software and telecom companies devoted to advancing open standards for mobile devices. The big innovation that this OS brings is the liberty for a user to modify all aspects of the system, from contact manager to image explorer, everything can be changed to the user taste.

So, why Android was the mobile operating system chosen for this application? First of all, Nielsen Company predicts that in 2011 smartphone's will be the majority of mobile phones, with the devices used by half of cell phone subscribers, or one hundred and fifty million people by mid-2011 (Nielson Company, 2010). Therefore a smartphone is a smart choice, and on the beginning of this project, in 2009, this OS was to demonstrate strong growth, compared to the others. The perspective of growth for the next few years is very good, like Gartner (Bradley, 2009) predicts, it will be the second most used OS on mobile devices. Secondly, Google is one of the biggest corporations on the world and they are expanding their business with their own Operating System for mobile phone. Android give Google the perfect platform to deliver all their content into one easy-to-manage bundle. So they have a platform at the end user hand to deliver their content perfectly. Third reason is because it is an open platform and open source, so it means it can be developed easy and cheap with less commercial issues. A lot of developers around the world can use API for develop and share their own Android application. Finally, it is very user friendly, easy to learn and despite being a robust Operating System it allows a full customization by the user.

Nowadays, it can be confirmed that this was a good choice. Since today's penetration of smartphones, in United States of America, is 25% of the mobile phones market share. While the iPhone has been the headline grabber over the last few years in the smartphone market, Google's Android OS has shown the most significant expansion in market share among current subscribers. Android's rise is even more noticeable among new smartphone subscribers in the last six months where Android has nosed past Apple's iOS in the last quarter to grab a 27% share of those recent smartphone subscribers (figure 4).

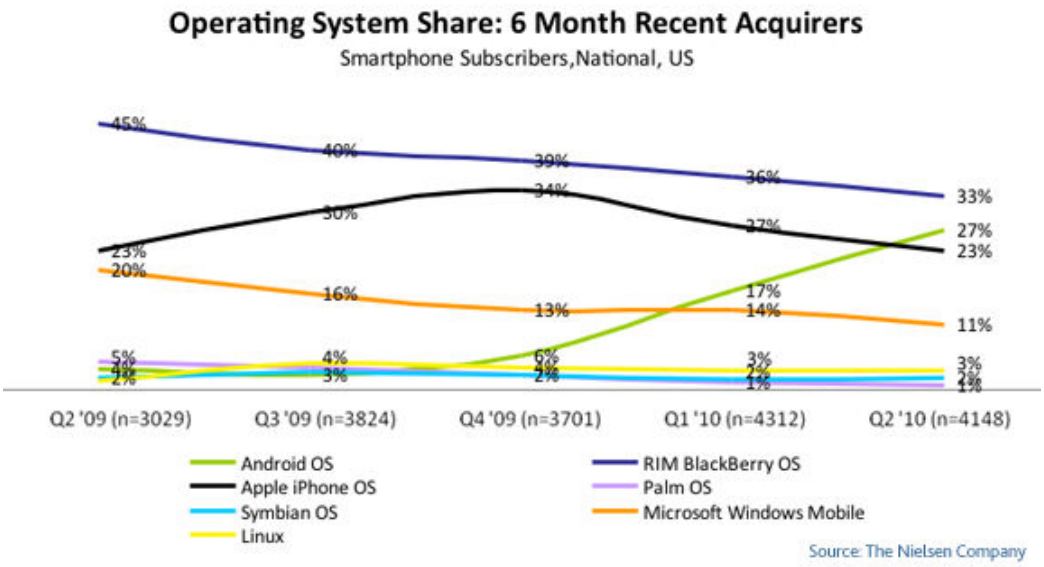


Figure 4 - Mobile Operating System Share (Nielson, 2010)

This happens mainly because Android is already being used in many different terminals of various brands, covering a larger number of kinds of people, and iOS is only used by Apple in their iPhones. In order to cover a wider market, PSiS Mobile was developed in the Android platform.

3.1. Android Architecture

Android applications are developed using Java. Android itself isn't a language, but rather an environment to run applications. As such, theoretically any distribution or Integrated Development Environment (IDE) to begin the development of applications can be used.

To keep with the Open Handset Alliances theme of truly opening the mobile development market, Eclipse is the recommended IDE for Android applications. Eclipse is one of the most fully featured, free and easy to work Java IDEs available. This consortium has released an Android plug-in for Eclipse that allows the creation of Android-specific projects, compile them and use the Android Emulator to run and debug them. Still it is possible to create Android apps in other IDEs, but the Android plug-in for Eclipse creates certain setup elements such as files and compiler settings automatically. The help provided by the Android plug-in for Eclipse saves precious development time and greatly reduces the learning curve, which means more time to create applications.

To begin the creation of Android projects, Android SDK (Software Development Kit) is needed. It contains all the Java code libraries needed to create applications that run specifically on the Android platform. The Android SDK also contains libraries to applications access into core Android features such as those associated with cell phone functions (making and receiving calls), GPS functionality and text messaging. These libraries make up the core of the SDK and will be the ones that most be used often.

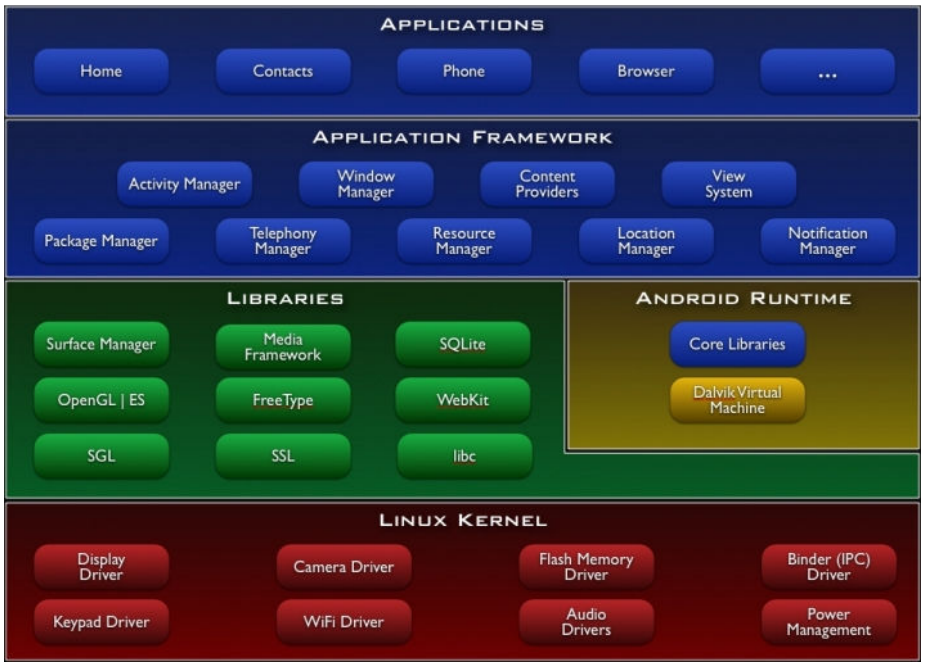


Figure 5 - Android System Architecture (Google Inc, 2010)

As can be seen on figure 5, internally Android uses Linux for its memory management, process management, networking and other operating system services. The Android phone user will never see Linux and programs never make Linux calls directly.

The next layer above the kernel contains the Android native libraries. These shared libraries are all written in C or C++, compiled for the particular hardware architecture used by the phone and preinstalled by the phone vendor.

Some of the most important native libraries are the follows:

- Surface Manager: Android uses a window manager similar to Vista or Compiz, but it's much simpler. Instead of drawing directly to the screen buffer, drawing commands go into off-screen bitmaps that are then combined with other bitmaps to form the display that user sees. This lets the system to create all sorts of interesting effects such as see through windows and fancy transitions;
- 2D and 3D Graphics: Two and three-dimensional elements can be combined in a single user interface with Android. The library will use 3D hardware if the device has it or a fast software renderer if it doesn't;
- Media Codec's: Android can play video, record and playback audio in a variety of formats;
- SQL Database: Android includes the lightweight SQLite database engine, the same database used in Firefox and in Apple iPhone. This can be used for persistent storage on applications;
- Browser Engine: For the fast display of HTML (Hypertext Markup Language) content, Android uses the WebKit library. This is the same engine used in the Google Chrome browser, Apple's Safari browser, Apple iPhone's and Nokia's S60 platform.

Also sitting on top of the kernel is the Android runtime, including the Dalvik virtual machine and the core Java libraries. The Dalvik VM (Virtual Machine) is Google's implementation of Java, optimized for mobile devices. All the code for Android will be written in Java and run within the VM.

Sitting above the native libraries and runtime, is the Application Framework layer. This layer provides the high-level building blocks that will be used to create PSiS Mobile applications. The framework comes preinstalled with Android, but can be also extended with new components if it is needed. The most important parts of the framework are the follows:

- Activity Manager: This controls the life cycle of applications and maintains a common "back stack" for user navigation;
- Content Providers: These objects encapsulate data that needs to be shared between applications, such as contacts;
- Resource Manager: Resources are everything that goes with the program that isn't code;
- Location Manager: To Android phone knows where it is;
- Notification Manager: Events such as arriving messages, appointments, proximity alerts, alien invasions and more can be presented in an unobtrusive fashion to the user;

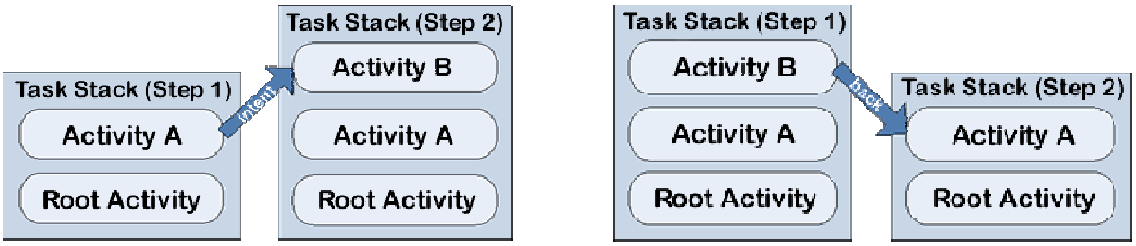
On standard Linux or Windows operating systems, user can have many applications running and visible at once in different windows. One of the windows has keyboard focus, but otherwise all the programs are equal. Users can easily switch between them, but it's their responsibility to move the windows around to see what they are doing and close programs that aren't needed anymore.

Android doesn't work that way. In Android, there is one foreground application, which typically takes over the whole display except for the status line. When the user turns on its phone, the first application he sees is the home application (figure 6). This program typically shows a background image, a search or clock widget, and a scrollable list of other applications that user can invoke.

Figure 6 - Android Home Application

When the user runs an application, Android starts it and brings it to the foreground (figure 7a). From that application, user might invoke another application, or another screen in the same application, and

then another and another. All these programs and screens are recorded on the application stack by the systems' Activity Manager. At any time, user can press the “back” button to return to the previous screen on the stack (figure 7b). From the user’s point of view, it works a lot like the navigation history in a web browser. Pressing “back” button it returns to the previous page.



a) b)

Figure 7 - Task Stack: a) new Activity invocation; b) back to previous Activity

Internally, each user interface screen is represented by an Activity class. Each Activity has its own life cycle. An application is one or more Activities plus a Linux process to contain them. That sounds pretty straightforward, but it isn't. In Android, an application can be “alive” even if its process has been killed. Putting it in another way, the Activity life cycle isn't tied to the process life cycle. Processes are just disposable containers for Activities. This is probably different from every other system.

During its lifetime, each Activity of an Android program can be in one of seven states, as shown in figure 8. The developer, don't have control over what state the program is in. That’s all managed by the system. However, a notification is sent when the state is about to change through the onXX() method calls, where XX is the new state.

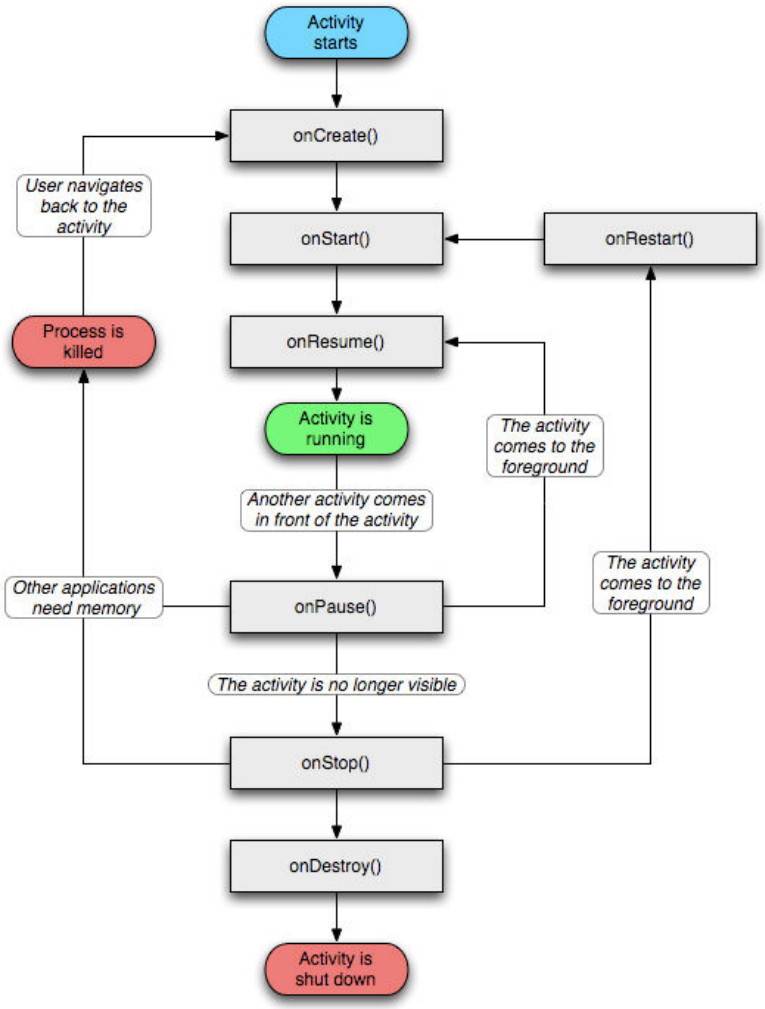


Figure 8 - Activity Lifecycle (Google Inc, 2010)

These states can be overridden in the Activity class, and Android will call them at the appropriate time:

- onCreate(Bundle): This is called when the Activity first starts up. It can be used to perform one-time initialization such as creating the user interface. onCreate() takes one parameter that is either null or some state information previously saved by the onSaveInstanceState( ) method;
- onStart(): This indicates the Activity is about to be displayed to the user;
- onResume(): This is called when Activity can start interacting with the user. This is a good place to start animations and music;
- onPause(): This runs when the Activity is about to go into the background, usually because another Activity has been launched in front of it. This is where should be saved the programs' persistent state, such as a database edited record;
- onStop(): This is called when the Activity is no longer visible to the user and it won't be needed for a while. If memory is tight, onStop() may never be called (the system may simply terminate the process);

- onRestart(): If this method is called, it indicates that the Activity is being redisplayed to the user from a stopped state;
- onDestroy(): This is called right before the Activity is destroyed. If memory is tight, onDestroy() may never be called (the system may simply terminate the process).

Activities that aren't running in the foreground may be stopped or the Linux process that houses them may be killed at any time in order to create new Activities. This will be a common occurrence, so it's important that the application must be designed from the beginning with this in mind. In some cases, the onPause() method may be the last method called in the Activity, so that's where the data should be saved.

All the application code and the used resources are encapsulated in a APK (Android Package) file, which is then used to distribute and install the application.

As mentioned earlier, every application runs in its own Linux process. The hardware forbids one process from accessing to another process memory. Moreover, each process has its own virtual machine, creating an isolated execution environment for each application (figure 9).

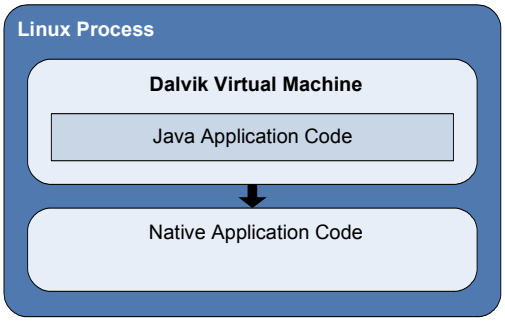


Figure 9 - Execution Environment of an Android Application

Another special feature in the implementation of Android applications is the fact that to each application is assigned a specific user id. This allows that the application resources aren't available for other applications, due to lack of access permissions.

In Android exists the possibility that an application uses components from another application, if that other application permits. This mechanism allows viewing an application as a set of components possibly independent, and not as a block of code with a single starting point (*e.g.*, a method or function "main").

In addition, accesses to certain critical operations are restricted and must be specifically asked for permission to use them in a file named AndroidManifest.xml. When the application is installed, Package Manager either grants or doesn't grant the permissions based on certificates and, if necessary, user prompts. Android can even restrict access to entire parts of the system. Using XML tags in AndroidManifest.xml, can be restricted who can start an Activity, start or bind a Service, Broadcast Intents to a Receiver, or access the data in a Content Provider.

A few objects are defined in the Android SDK that every developer needs to be familiar with. The most important ones are Activities, Intents, Services, Content Providers, Broadcast Receivers and Resources.

3.1.1. Activities

An Activity is a user interface screen. Applications can define one or more Activities to handle different phases of the program. As discussed earlier, each Activity is responsible for saving its own state so that it can be restored later as part of the application life cycle.

3.1.2. Intents

Intent is a mechanism for describing a particular action, such as “pick a photo”, “call home”, or “open the browser”. In Android, just about everything goes through Intents, so there are plenty of opportunities to replace or reuse components. Each Intent is associated with a component, which is invoked when the Intent is called.

For example, there is an Intent for “send an email”, which can be invoked when the application needs to send an email. If a new component needs to be associated with the Intent (send an email) it can be registered as an Activity to replace the standard email program. The next time somebody tries to send an email, they’ll get the option to use that program instead of the standard one.

Intents can be used with the method startActivity (Intent) to initiate an Activity, with the broadcastIntent (Intent) to send it to all interested BroadcastReceiver components and StartService (Intent) or bindService (Intent, ServiceConnection, int) to communicate with a service in the background.

Intent Provides are a great facility for making a connection between the runtime codes from different applications. It is basically a passive data structure holding an abstract description of an action to be executed. The primary pieces of information on Intent are: the action and data. The action is the general action to be performed, as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc. The data is the data for operation, as a registered person in the database from the contacts list, expressed in URI (Uniform Resource Identifier).

Example, ACTION_EDIT content://contacts/people/1 - Initializes the Activity that allows the edition a person information, present in the contact list, with the identifier 1.

3.1.3. Services

A service is a task that runs in the background without the users' direct interaction, similar to a UNIX daemon. For example, considering a music player, the music may be started by an Activity, but it's supposed to keep playing even when the user has moved into a different Program/Activity.

So, the code that does the actual playing should be in a service. Later, another Activity may bind to that service and tell it to switch tracks or stop playing. Android comes with many built in services, along with convenient APIs to access them.

3.1.4. Content Providers

A content provider is a set of data wrapped up in a custom API to read and write it. This is the best way to share global data between applications. For example, Android provides a content provider for contacts. All the information there: names, addresses, phone numbers and so can be shared by any application that wants to use it.

3.1.5. Broadcast Receivers

The broadcast receivers act as an event listener. These events are usually initiated by the system and can be notifications like: system time was changed, the battery is weak and that a photograph was taken, among others.

3.1.6. Resources

A resource is a localized text string, bitmap or other small piece of non code information that a program needs. At build time all the resources get compiled into the application. Resources are created and stored in the *res* directory inside the project. Android resource compiler (AAPT, Android Asset Packaging Tool) processes resources according to which subfolder they are in and file format. For example, PNG and JPG format bitmaps should go in the *res/drawable* directory, and XML files, that describe screen layouts, should go into the *res/layout* directory. The resource compiler compresses and packs resources, and then generates a class named "R" that contains identifiers to use to reference those resources in the program. These are a little different from standard Java resources, which are referenced by key strings. Doing it in this way allows Android to make sure that all the references are valid and saves space by not having to store all those resource keys.

Chapter 4

Mobile Recommendation System in PSiS

In this section PSiS Mobile architecture will be presented and will be discussed the reasons and the importance of a mobile system to support the central system (PSiS main project). Current mobile limitations will be presented as well.

PSiS only interacts with tourists through a web application only accessible from a browser, but it's indispensable to have a tool to assist tourists "on the field". Thus, a mobile tool to be integrated in the PSiS project, called PSiS Mobile, was studied and developed. This tool also takes into account the tourist current context and nearby sights context.

In this preliminary phase, PSiS project, that includes PSiS Mobile, will be limited to data from metropolitan area of Porto, Portugal. But, it is designed so that no data or user restrictions are imposed. PSiS project is composed by two pieces: the server-side and mobile client. All the main information like user profiles, history and similarity values are compiled on the server. In other words, all the recommendation aspects are on the server, since it classifies sights with a rate to a specific user. There is also a complete database with all information about points of interest in a certain city/region, and a complete users' portfolio as well as their visit history.

Mobile client is a very important piece in all the system, because it supports the user on the go. With a PDA, user can see the tour plan recommended by PSiS and information according to his context. With this the system can offer more effective recommendation about places to visit and re-planning the original visit in real time. The system interacts with the user providing information about nearby sights

to see. These points of interest are recommended according to user profile and context. Shows trip planning for current day, that can be re-arranged according to current user context, for example, if a tourist is behind schedule a planning algorithm is executed to re-plan the tour. It also shows favorite sights stored on the system.

With PDA specific hardware and software users' current context can be known, *i.e.*, its location, day/time information and traveling speed. With this information, more information can be known, for example, can be known what is the weather forecast for that location at that moment, to refine the recommendation (to not suggest outdoor spaces to visit). With traveling speed, planning can be made more effective by calculating the time that takes to get from one to another point of interest. Besides the already mentioned information: on, ahead or behind schedule.

PSiS Mobile application manages some basic recommendation routines only. What this mean is that it will not classify (or rate) points of interest, but only shows the results to the user, filtering the database content based on current context. For example, if a user likes Chinese food, certainly a Chinese restaurant has a higher classification value according to the user preferences (might not happen if classification is given using collaborative filtering, since the restaurant might have a negative classification by similar users).

Mobile client shows points of interest, for that user and for a specific category, ordered by classification (downward; higher classification appears first). After visiting some points of interest, the user can provide feedback about the visited place.

A possible use case can be defined like this: a tourist registers on PSiS using the web application, defines his profile (demographic information) and requests a recommendation for a trip that has two days, in a specific city. After that, the tourist loads the generated information into the mobile application. This load can be done, for example via Wi-Fi or USB (Universal Serial Bus). Now, the mobile application has all the information about the places to visit that match the user profile. With context-aware information, points of interest to visit can be suggested more effectively and, if necessary, a re-plan of the initially PSiS generated trip can be recommended. Besides, the user can inform the system if whether or not he liked a suggested point of interest.

When tourist is going to see a point of interest, PSiS Mobile shows detailed information about it in order to the tourist acquire more knowledge about what he is going to see, for example, the history of a museum.

Augmented Reality is also implemented in PSiS Mobile, meaning that the user will have the option to pointing the PDA to the point of interest direction and access to its detailed information. These details will include information like pictures of the point of interest in other seasons of the year (*i.e.*, covered in snow). This application offers built-in social networking too, so the user can share his pictures with the community in a matter of seconds. Despite all the features to be implemented in this application, it is intended that it be must smooth and easy to understand. To facilitate navigation throughout the application it is essential that the number of clicks between various features is kept at the minimum.

To conclude, it is pretended to implement a real application that really helps people on seeing what they expect to see, or going where they like. It is important to develop an optimized communication mechanism to ensure that a tourist doesn't waste too much time just to gather the necessary information.

4.1. System Architecture

In PSiS case scenario, there is a server side with Microsoft technology, and all the recommendation system is working under .NET framework and stored procedures on server database. The database, that is present in the same physical server, is implemented on SQL Server 2008.

At user side, there is a PDA running Android OS. The problem is that Android uses Java technology. Resuming, there are two different implemented modules with different technologies that need to communicate. Another issue is the low RAM (Random-access Memory) memory capacity: only 288MB for the whole system, so there must be very careful in the mobile application development. On the other hand, this PDA is equipped with HSDPA interface that allows up to 2 Mbps up-link and 7.2 Mbps down-links speeds. But, in order to conserve battery power and decrease latency on these devices, it is necessary to stay off the "radio" as long as possible, download as few bytes as possible and avoid time consuming data parsing routines, since they consume too much battery power. Consequently, there must have a balance between usability, features, and performance.

There are many considerations at the mobile application tier, including data availability, communication with middleware, local resource utilization and local data storage. In addition, many business factors need to be considered.

Since there are two different technologies communicating with each other, and the base system is already implemented, a middleware must be created to bridges communications between these two technologies, see figure 10. This means that the mobile middleware will play a crucial role on the system. (Anacleto *et al*., 2010b)

Some of the important features of the middleware include security, data synchronization, device management and the support for multiple devices.

When extending an application into mobile devices, the challenges mentioned in chapter 4 needs to be effectively addressed. The architecture needs to consider components that work in conjunct to address these challenges.

Because this is an occasionally connected application (smart client), a temporary database is used on the mobile device to permit access to some data without constantly use the network and to allow the application to work without internet connection. But it has multiple limitations, like no access to new points of interest and advanced recommendations.

First of all, after requesting a recommendation for a trip, all the necessary data is transferred from the server and stored on the mobile device. This is necessary, because the low Internet speed rates on

mobile phones and possible network unavailability. When is said necessary data, it means, the information of all the points of interest that will be on the planning schedule and other points of interest nearby the first ones. This approach is useful if the tourist wants, or is necessary, to re-plan the schedule in real time.

All the collected data, photos, user feedback, user context, and others will be stored on the device. At the end of the tour all these data will be sent to the server to posterior analysis to improve future recommendations.

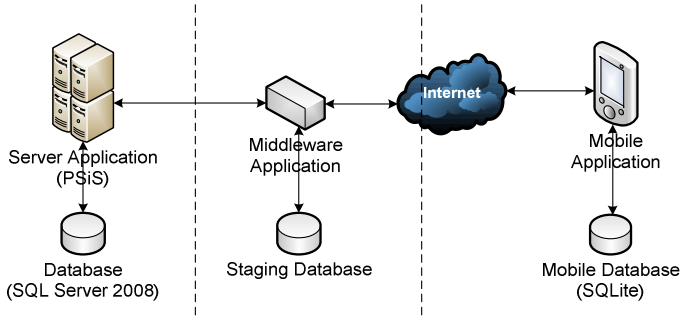


Figure 10 - PSiS Mobile Architecture

The system architecture can be summarized saying that:

- The existent system will doesn't be changed;
- The Middleware Application is a component that will reside on the Server Side and will be developed on Java, with directives to permit the communication between the existing system and the mobile application. This middleware includes a user authentication module, a data synchronization service to synchronize data between the system and the mobile device and a data access manager with a staging database;
- The mobile application runs on Android devices and is used to capture/send data from/to the field. The application also has a synchronization component to synchronize the handheld data with the server database. Also, it contains a communication manager that is responsible for the communication protocol with the Middleware Application, accepting responses and making all the requests; A database is used to enhance the application performance, besides the other necessary layers;
- Internet connection is used to retrieve/update itinerary information, points of interest information and personal preferences. Data is uploaded and downloaded automatically without user intervention.

Nowadays the bigger problem for mobile devices is the low battery power, because the network problems referred before are being tackled with easy access in large cities to free Wi-Fi connection. The application is prepared to preferentially use the Wi-Fi connection, because it is faster and has much less latency than a 3G connection.

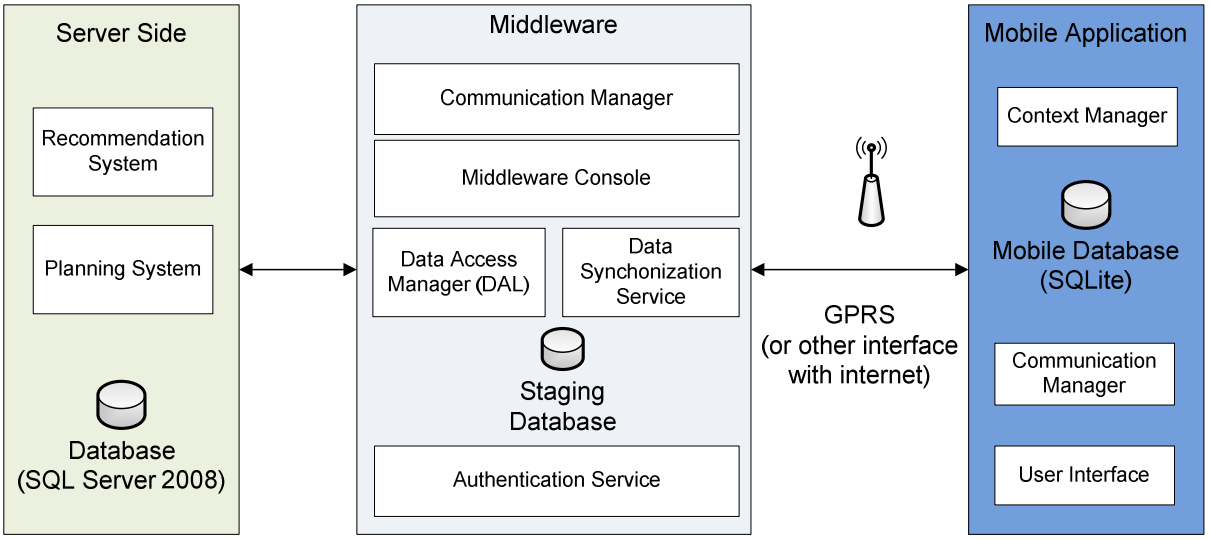


Figure 11 - Detailed Solution Architecture

As can be seen on Server Side a recommendation system attributes a rating to each point of interest according to the personal profile. It recommends sights to visit, restaurants to eat and hotels to rest. Personal profile is also stored at the server, as well as the trips history and user context. The profile is stored and changed along the time, according to users' feedback and relationships with other users. The planning system is responsible for planning a trip based on the recommendation previously made for that place and to that person. (Anacleto *et al.*, 2010b)

PSiS complete architecture is presented on figure 11. Here is given a detailed description of each module. Starting with the middleware component, it has the following modules:

- Data Access Manager (DAL) - Consists on a piece of code that communicates with the staging database and server database. (insert, update and delete data and have preconfigured methods that return ad-hoc real-time data to the mobile device). Contains also a business logic to do specific activities that aren't part of the Middleware core, but part of the mobile solution. In this case it gets location updates captured from the device and uses them to retrieve information about the place to help the tourist;
- Staging Database - This module manages the data that is at the staging database, where a replica of all transaction tables with mobile users is stored. It handles in-queue, out-queue and data archiving. Because data conflicts may arise, this module is responsible for gracefully managing and handling those conflicts. Database Synchronization is also possible through this module;
- Communication Manager - It's the most important piece of software at Middleware component, this service is responsible for bridging communications between the applications. It composes messages to be sent from the back end to the mobile device and vice-versa, sending data to the “out queue”, picking it from the “in queue”, and use the received messages from the mobile device to call the Data Access Manager. To have effective data exchanges, data

optimization and data security, are necessary. All the transferred data is compressed, to consume less time on the data transmission between Middleware and the Mobile Application;

- Data Synchronization Service – This is a core component of the Middleware. Incoming data from the mobile device is received into the in-queue and the outgoing data is pushed to the mobile device from out-queue. It has background synchronization from the device and maintains a user-wise queue and checks for new records to be sent to the mobile device;
- Authentication Service - Manages mobile device users through a user list that is linked with the main system users to use the same authentication on mobile devices. Also, it authenticates the mobile user during login process and synchronization. It also has the capacity to handle multiple connections at the same time;
- Middleware Console - Is responsible to make the visual interface to system administrator know what is happening. Here is where logs can be seen and all the existing activity between the Server Side, Middleware and the Mobile Application.

The following modules were implemented in Mobile Application:

- Communication Manager - Establishes connection to the network and optimizes the way data is sent to the Middleware (compressing the data). It is also responsible for authenticating a user with the Middleware if there is network connectivity, otherwise authenticates with credentials available locally. Also calls the methods defined in the DAL (Data Access Layer) to have real-time data on the mobile device, although, if connectivity is not available then it is used the data existing in the mobile database;
- Mobile Database - Manages data in the device database, applying and composing the data. It is responsible to clean up temporary data from the database;
- User Interface - It is the face of the mobile solution, it validates the user input with some business rules and interacts with internal hardware.

Chapter 5

Server Communication Evaluation

In this section will be presented a case study which assesses the fastest and most reliable method to exchange information between the server side and the mobile system.

It's clear that current mobile devices still have several limitations compared to traditional computers. It was under those limitations that began to question: Which is the best way to exchange information between the server and mobile client in order to minimize these limitations?

To answer this question, a case study was performed. This case study involves the transmission of points of interest present in the servers' database into the mobile device. These points of interest are subsequently written in the mobile local database. A HTC Hero with Android 1.5 was used to run the client application and a normal notebook PC was used as server to perform this test. Both were connected to the same Wi-Fi network, with a speed of 54 Mbps.

To examine what is the best method to perform the actual transmission, five metrics were used:

- Duration of the entire process (server request, data reception, deserialization and record of data on local database);
- Average CPU load;
- Memory used;
- Total bytes sent;
- Total bytes received.

The Android SDK doesn't include methods to obtain the desired information (*e.g.*, the process CPU load). Instead, and since the Android operating system is based on the Linux kernel, the information

is located inside operating system files associated with the process. These files must be accessed and read through file I/O operations.

In this case, the desired file, named *stat*, is in the directory *"/proc/PID/"* and is composed of several lines, where each represents a different parameter. These lines are always updated by the kernel.

5.1. Inter-process Communication Flow

There are several ways to accomplish the exchange of information between a server and its clients. In this case, was chosen to study two of the most used exchange protocols: Java Socket API and HTTP (REST, REpresentational State Transfer Web Services).

The other types of Web Services were left out because they are too heavy for a mobile environment, *i.e.*, they have bigger headers than the REST method, thus increasing the amount of traffic used.

5.1.1. Sockets

Raw sockets can be used to exchange information quickly. This was the first tested approach.

First of all, a raw socket client and server modules were implemented. The server permanently listens for client connections. When a connection is established, the server creates two new threads: one for sending data to the client, and another for receiving data from the client. Since there are two different threads for sending and receiving data, the exchange can be performed asynchronously, avoiding waiting states by the client application.

After choosing the communication protocol, it was necessary to define the information structure in order for the two parties, in action, to "understand" each other, so it was used an XML structure. The data was deserialized using SAX Parser.

With this system message sizes were reduced. This is because there aren't any headers (*e.g.*, HTTP and SOAP headers).

However, this system poses several problems in the sockets management. Besides the apparent need to specify a hard-coded and very inflexible communication protocol, raw sockets also need further implementation for error detection and transaction control.

5.1.2. HTTP

HTTP is one of today's most popular client-server communication protocols. Being a relatively old and widely used protocol, the errors that were found using raw sockets, don't have emerged with this protocol because this API already handle errors, facilitating its use and implementation.

The only downside to the previously described raw socket communication protocol is the size of the data frames sent/received. This happens because the HTTP header is added to the sent and received data.

The size of the header along with the *ACK packages* sent and received to validate the information transaction varies between 6% and 10% of the total file size to be transferred. For example, for a file with 1.875 Mb of size, the client receives 2.048 Mb (9% more than the size of the original file).

As this protocol supports the transfer of various types of files, it was chosen to test the exchange of data structured in three different formats:

- XML, because it is one of the most popular formats used to store and exchange information in a structured and hierarchical way. In this scenario was used three different XML parsers: DOM (Document Object Model), SAX (Simple API for XML) and Pull. The first was chosen because it is one of the most used. Finally, the last two claim to be the fastest on doing the XML files parsing;
- JSON (JavaScript Object Notation), which base is identical to XML, but tries to be a lighter format;
- Protocol Buffers, which is a format for serialization/deserialization developed by Google. In particular, it was designed to be faster than XML so the primary purpose of Protocol Buffers is to facilitate network communication focusing in simplicity and performance.

5.2. Empirical Analysis

In this section will be presented the results for each of the previously described methods: HTTP and raw sockets.

To ensure more accurate results, it was made four different types of tests regarding the exchange of information. The first test consisted on sending information about only one point of interest. The second test consisted in sending 250 points of interest at once. In the third test, were sent all the 461 points of interest present in the database. Finally, a more thorough test was made, consisting in the transmission of four times, all the points of interest giving a total of 1884 points of interest.

These results are shown in the form of charts, where there is a chart for each metric.

5.2.1. First Test - 1 POI

In this first test, was tested both methods (raw sockets and HTTP), using only one point of interest.

The original file sizes are, 1009 Bytes (XML), 665 Bytes (Protocol Buffers binary file) and 779 Bytes (JSON).

Analyzing figure 12, it appears that the fastest method is HTTP using Protocol Buffers, followed by HTTP using JSON.

The raw socket method proved to be slow mainly because the connection system initialization, which is a time consuming process, especially when it tries to control errors that may exist in the connection.

However, it was the raw socket method that had fewer bytes transferred between server and client (figure 13a and figure 13b) followed by the binary serialization of Protocol Buffers. Finally, HTTP with XML is the longest of all.

In this case weren't provided charts for the CPU load and memory metrics because the process is completed so quickly that it is not possible to obtain significant values (the readings are made per second).

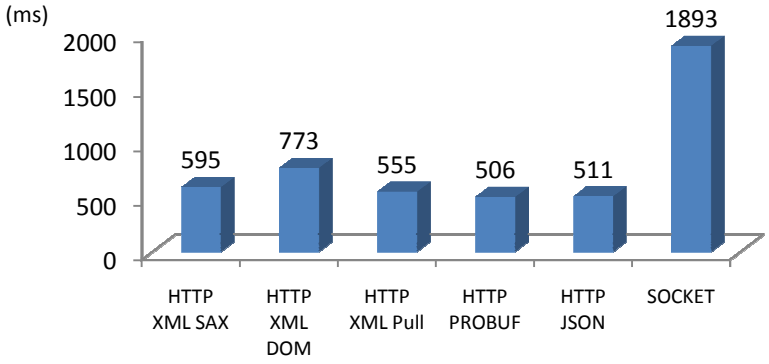


Figure 12 - Process Duration In First Test

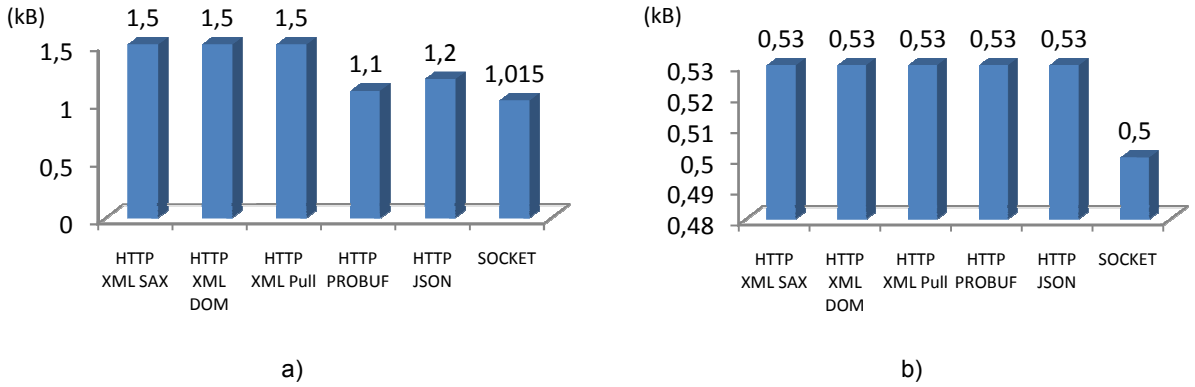


Figure 13 - a) Kbytes Received On Mobile In First Test; b) Kbytes Sent To Server In First Test

5.2.2. Second Test - 250 POIs

In the second test were transferred 250 points of interest. One of the most relevant findings allowed us to analyze that the XML parsing algorithms have significant differences of performance between them (figure 14). The DOM proved to be the slowest and SAX proved to be the fastest, surpassing Protocol Buffers that, only in this test wasn't the best.

In this case the XML file has 253 kB (kiloByte), Protocol Buffers binary file has 195 kB and JSON file has 227 kB.

Looking at figure 16a and figure 16b, the XML serializations are still heavier than JSON and Protocol Buffers. Socket method consumes less system resources, CPU (figure 15a) and memory (figure 15b),

than the others because it doesn't have so many parsing routines. Meanwhile the whole process still takes a long time to execute.

Protocol Buffers was the one that has transferred less bytes, being better than socket method. This happens because Protocol Buffers uses data compression.

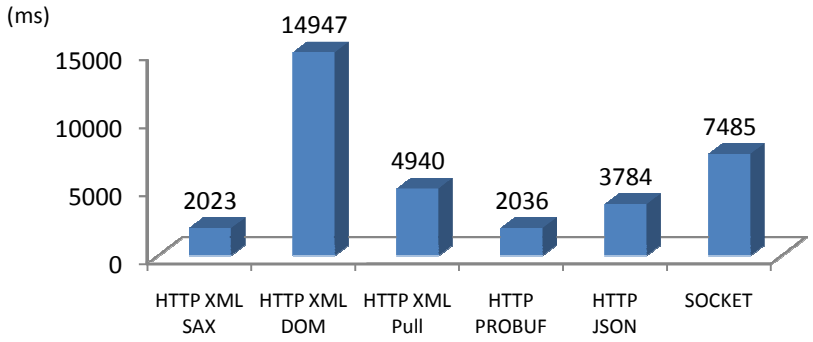
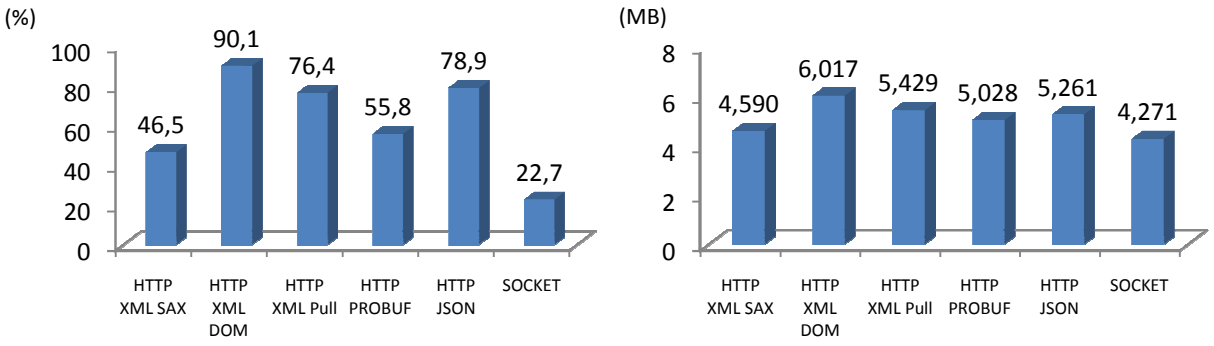


Figure 14 - Process Duration In Second Test



a) b)

Figure 15 - a) CPU Utilization In Second Test; b) Memory Utilization In Second Test

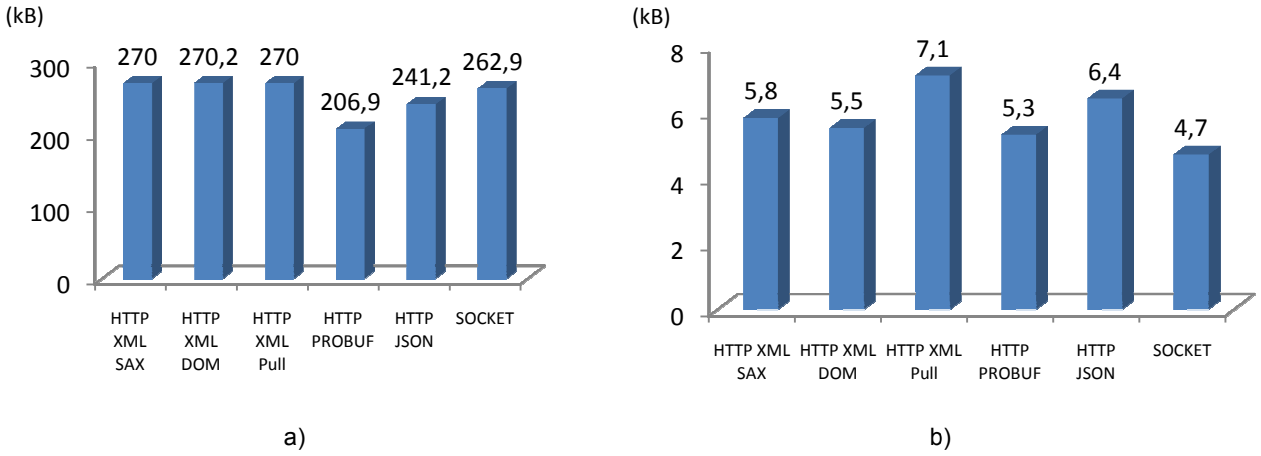


Figure 16 - a) Kbytes Received On Mobile In Second Test;b) Kbytes Sent To Server In Second Test

5.2.3. Third Test - 461 POIs

In the third test, it was transferred 461 points of interest. The results follow the same pattern of events of the previous tests where Protocol Buffers was the fastest of all, though only for a little margin. These files have a size of 375 kB for XML, 256 kB for Protocol Buffers and 313 kB for JSON.

JSON behaved as expected, its serialization is lighter than XML (figure 19a and figure 19b), but it has a weak decoder (which is built on the Android platform) and becomes slower when compared with the SAX Parser. Although JSON decoder uses more CPU (figure 18a), but less memory (figure 18b) than SAX Parser.

Comparing the second with the third test, there is an interesting observation regarding Protocol Buffers. Although the third test uses double of points of interest, the difference in processing time between both is just 300ms (milliseconds). The second test has a processing time of 2036ms (figure 14) and the third test has a time of 2316ms (figure 17).

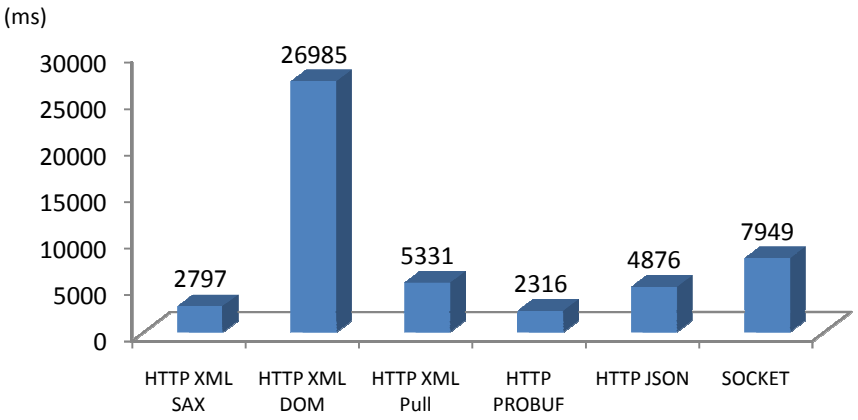
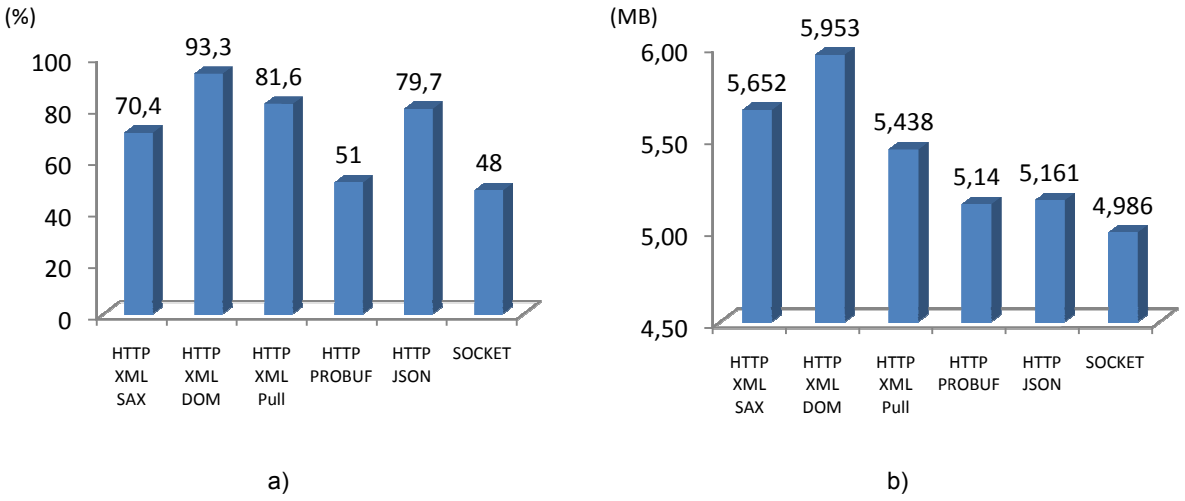
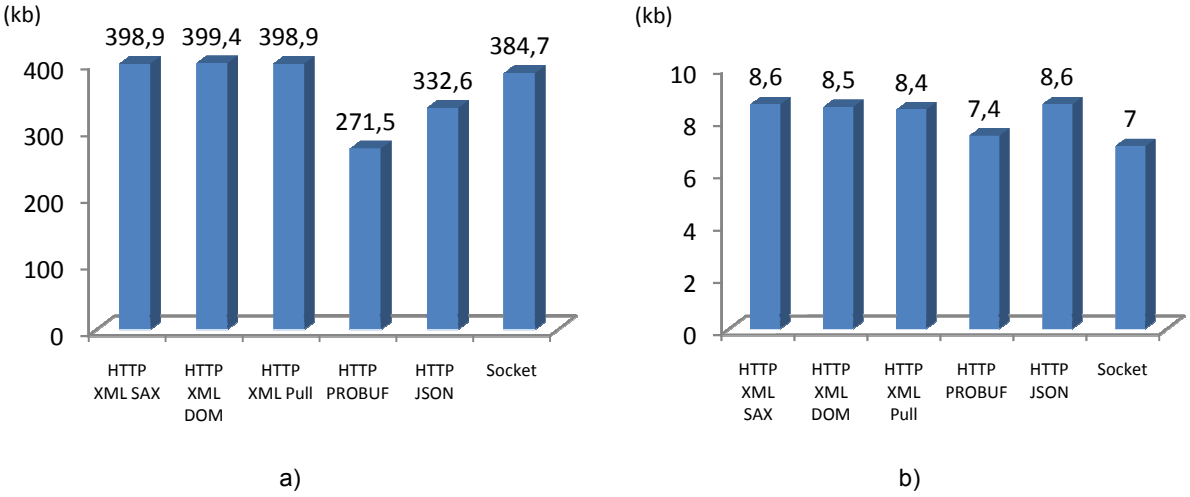


Figure 17 - Process Duration In Third Test



a) b)

Figure 18 - a) CPU Utilization In Third Test; b) Memory Utilization In Third Test



a) b)

Figure 19 - a) Kbytes Received On Mobile In Third Test; b) Kbytes Sent To Server In Third Test

5.2.4. Fourth Test - 1844 POIs

Finally, it was decided to do a more thorough test to obtain stronger results and to denote differences. In this test it was transferred 1844 points of interest.

Notice that only results for two methods are provided. This happened because all the others gave an "Out of Memory" error due to the PDA lack of memory. These two methods were the ones that had produced better results: the SAX Parser (XML file size was 1875 kB) and Protocol Buffers (Binary file size was 1276 kB). Both used almost the same system resources (see figure 21a and figure 21b).

In this test a great difference in performance between Protocol Buffers and SAX (in favor of Protocol Buffers) can be seen, especially in the transferred data (figure 22a and figure 22b). Protocol Buffers transferred about 600 kB less data and in two fewer seconds than the SAX parser (figure 20).

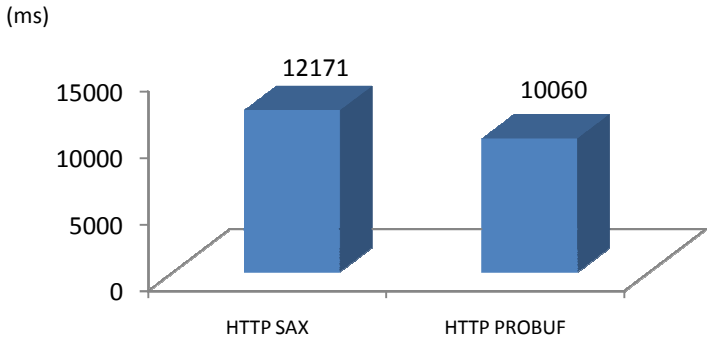


Figure 20 - Process Duration In Fourth Test

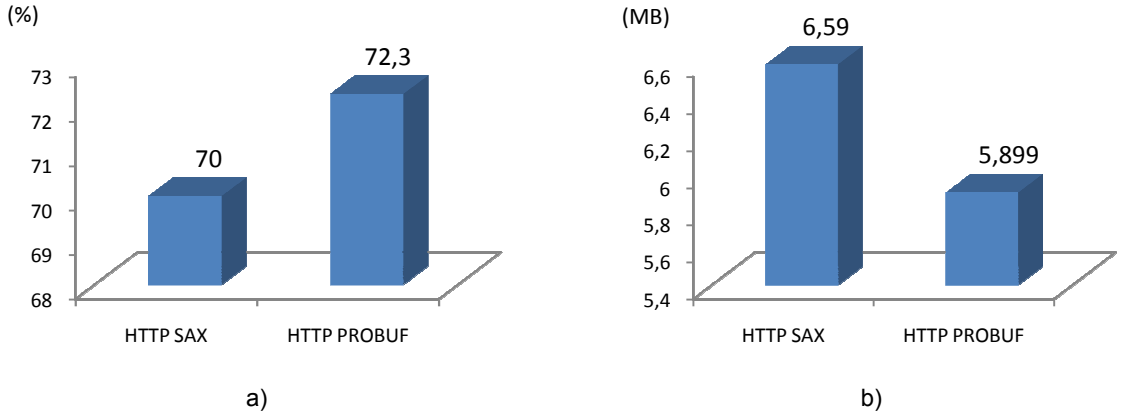


Figure 21 - a) CPU Utilization In Fourth Test; b) Memory Utilization In Fourth Test

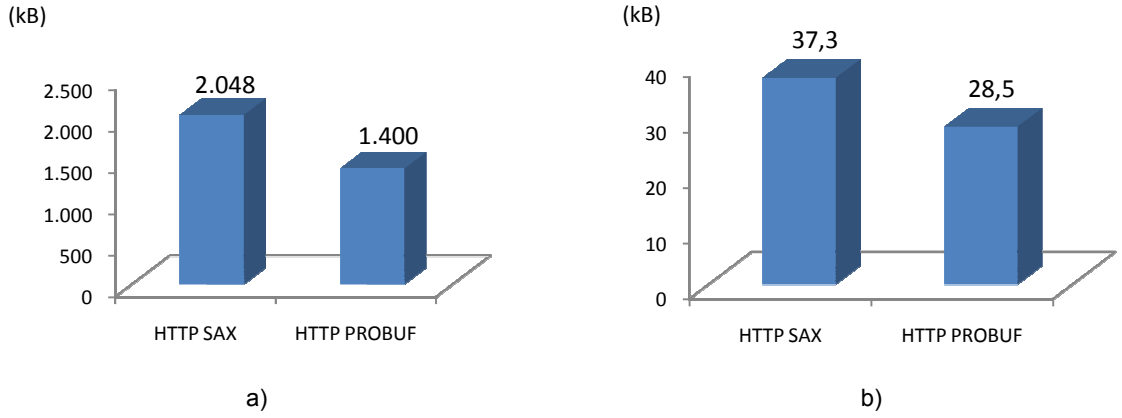


Figure 22 - a) Kbytes Received On Mobile In Fourth Test; b) Kbytes Sent To Server In Fourth Test

5.3. Conclusion

It must be remembered that the purpose of these tests was to ascertain which method is more reliable and faster to transfer information. Therefore, will be listed the most important conclusions that led to the selection of one of these methods.

In theory, socket approach seems to be the right choice. In practice, some important disadvantages compared to the others approaches were found. This implementation proved to be error prone and slower than other existent approaches. Doing the math of cost over benefit between this approach and HTTP, it was concluded that the gains of transferred kB's between the two points of raw sockets, don't outweigh the associated disadvantages.

It was left behind due to the few advantages that they actually bring, compared to the HTTP protocol. Also, raw sockets are much more complex and hard to work with. It's like reinventing the wheel when it already exists. On the other hand, HTTP is reliable and is able to perform error handling, natively. This was the chosen method for the implementation of PSiS Mobile.

Therefore, and starting with XML, the study showed that after all it isn't so slow to parse a XML file. This method is only slightly behind the others regarding file size, since it includes multiple tags and no compression.

Then, and as expected in this field, JSON occupies less space. It is one of its main objectives. However, it demonstrates to be slower than the best XML parser's.

Speaking of the XML parser's, it is noteworthy that DOM is definitely the slowest and most complex to work with method. The SAX ends up having a performance at all similar to Protocol Buffers, which proved to be the lightest of them all and the fastest in almost all tests. These two are, according to this test, the best approaches. SAX is overtaken by the Protocol Buffers when it comes to speed, thus can be concluded that Protocol Buffers is the best serialization method.

Finally, it appears that according to the previous statements, the HTTP method in conjunction with Protocol Buffers is the mechanism to be used on the exchange of information between the server side and the mobile system.

Chapter 6
Development and Results

This is the most important chapter, in which all the performed work in the project is presented. The theoretical work was already presented in the previous chapters. All implementation details, decisions and architecture of various modules of PSiS Mobile will be described.

Although today there are already new versions of the Android platform (version 2.2, called Froyo), all the mobile application was developed using version 1.5 (Cupcake), since it was the existing version on mobile device available for tests. One of the advantages of this approach is its backwards compatibility, so the application will work on all devices with versions above 1.5. But there aren't only advantages because the latest operating system functionalities are unavailable.

The application, at mobile side, is divided into four mainly pieces (figure 23). As saw earlier, Activities have states and they don't run all the time. So, a service was developed to run routines that must be always executing.

Context Service incorporates the following modules:

- Location Manager, that is responsible for retrieve GPS coordinates about users' position;
- Weather Service module connects to the WorldWeatherOnline web service, to get forecast for the users' position. This Web Service devolves an XML with that information and the system is using it to filter points of interest according to their outdoorness. For example, if it is raining, the tourist probably doesn't want to be wet, so a fully outdoor attraction will not be suggested to see;
- Phone Status Manager is responsible to get information about mobile device context. Battery status, internet connectivity (Wi-Fi or 3G) and GPS activity are the information that

this module deals. It reads this parameters and broadcast this information to the others' application modules;

- DateTime Manager deals with all information that involves hours. In other words, it returns current system hour and compares dates;
- Planning module incorporates the algorithm that is in charge of replanning the original route for the tourist, the description about this module is on section 6.5;
- Tracking module, as the name suggests, it tracks users' movements, registrant GPS coordinates where the tourist went, how many time he stays on a point of interest, etc.;

The other components are the Activities, which are the user interface of the application, the DAL, which handles all the classes that interacts with data and mobile SQLite database.

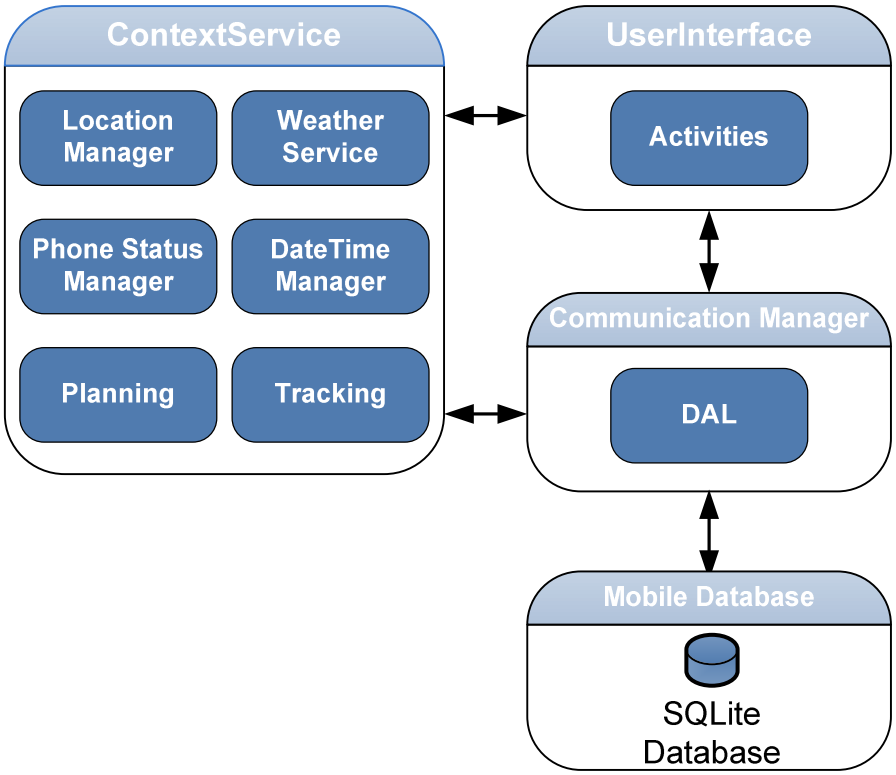


Figure 23 - PSiS Mobile Implementation Architecture

6.1. Activities

There are many Activities on this application, which one of them have a specific task. An Activity is a user interface concept. These Activities represent a single screen in the application and contain one or more views. Views are the objects present on the interface (Text views, Edit Views, etc.).

In figure 24 is presented the possible user interactions between user interface screens. The functions of all the Activities are described later in this section.

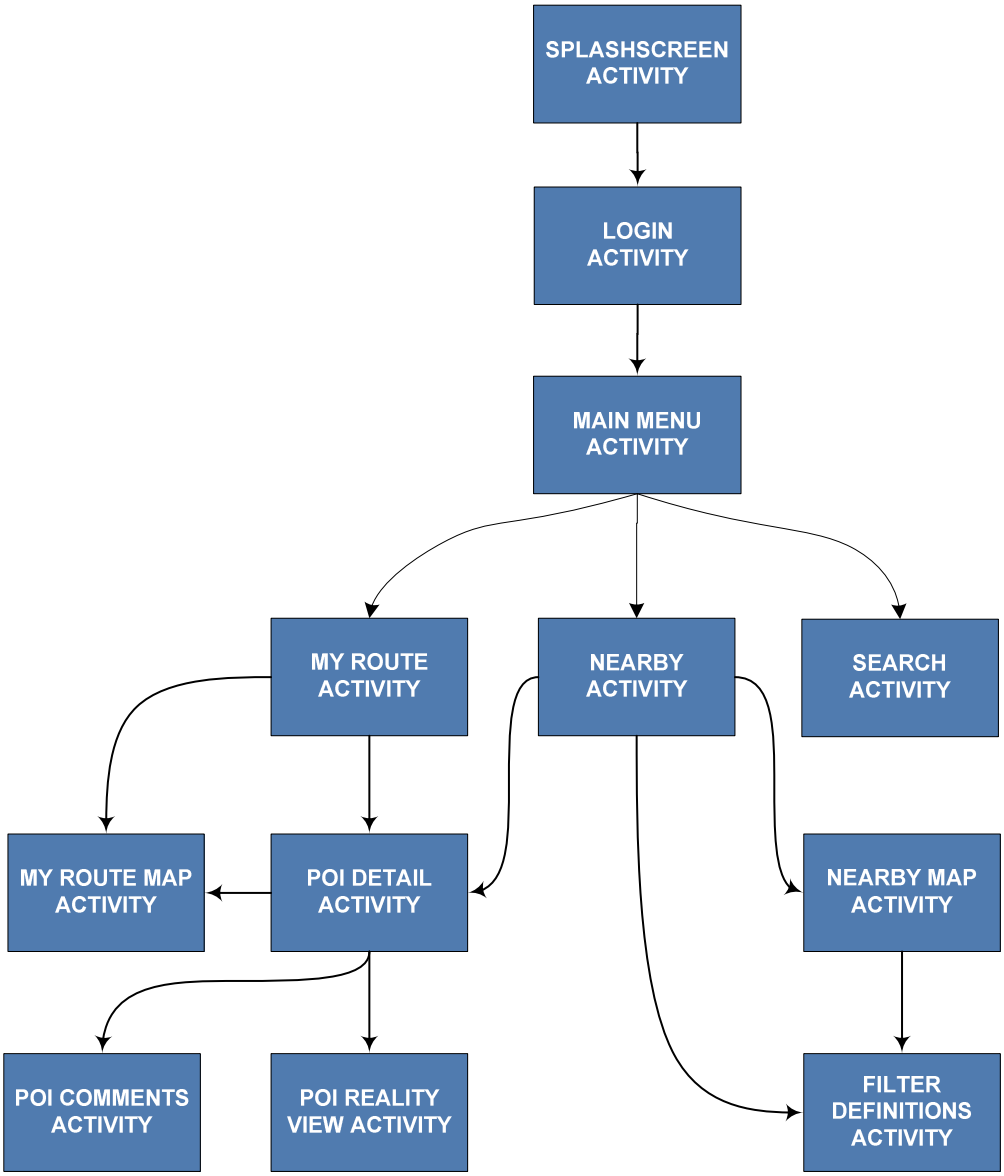


Figure 24 - Application Navigation

SplashScreen Activity is the first screen presented to the user when he enters the program. It is a simple image that introduces the application, as can be seen on figure 25a. This screen is active during 3000 ms, or until the user clicks on the screen. All these pre-defined values can be simply configurable on a XML file, presented on *res/values* application directory (as explained on section 3.1). So, to change these values it isn't necessary to make changes on application code.

a) b)

Figure 25 - a) SplashScreen Activity; b) Login Activity

After passing this first Activity, Login Activity is shown to the user (figure 25b). Here, he can have two options, make login with its current user or, if he isn't already registered in the system, goes to the registering web page, on PSiS portal. When user makes login for the first time, he is questioned if he wants to save his login information on the phone. This information is saved into SharedPreferences. The SharedPreferences class, included on Android API, provides a general framework that allows save and retrieve persistent key-value pairs of primitive data types for an application. It can save any primitive data: boolean's, float's, integer's, long's and string's. This data will persist across user sessions (even if application is killed). Next time user wants to enter on the application, he doesn't need to enter his login data again, application simple retrieves that information from SharedPreferences.

Every time user enters in the application and an internet connection is available, ContextService, verifies if any data from mobile device needs to be updated with server data. For example, planned routes, points of interest actualizations, etc.. More information about this process can be seen on section 6.2.

When login is successful, the main menu appears. This main menu is basically a tab menu, where the user can navigate between, MyRoute Activity, NearBy Activity and Search Activity. While this screen initializes, its created and lunched the context service. By default MyRoute Activity is shown.

Figure 26 - My Route Activity

MyRoute Activity, presented on figure 26, shows the programmed route, if any, for the current system day. It shows the points of interest, arrive hour and depart hour. As the tourist visit the points of interest, they are going "out" of the displayed route. Every time tourist arrives to the first point of interest, presented on the route, a warning is showed. This dialog questions the user if he wants to see the details of the arrived point of interest (opening POIDetail Activity - figure 28). With this, he can know more about what is going to see. After see the point of interest a dialog is presented to classify it. Also, if he is behind or ahead schedule, a re-plan can be suggested, according to knowledge acquired by the replanning module. User can also remove points of interest presented on the route.

Figure 27 - My Route Map Activity

On that same Activity, user can navigate to another's. Simply press's phone "menu" button, and can interact with MyRouteMap Activity, figure 27. It displays the directions from user current location to the

next point of interest presented on route. This Activity is only displayed if an internet connection is available, because this itinerary is retrieved from Google Maps WebService.

System raises Google Maps Web Service sending start and destiny GPS coordinates being returned a KML (Keyhole Markup Language) file that is parsed to create this route on an overlay over the Map. It was necessary to create this class, because Android API doesn't have any method that can do this. After KML file is parsed, for each straight (that starts and destiny on a certain latitude and longitude), is designed on the overlay a blue color line, which in the final ends up to giving the appearance presented on figure 27. In start and destiny points, a blue circle is designed, that represent them.

Now explaining this in code, it was developed using only two classes, MyRouteMapActivity and MyRouteMapActivityOverlay. The first one sends an HTTP request to the following link,

*"http://maps.google.com/maps?f=d&hl=en&saddr=**STARTLATITUDE**,**STARTLONGITUDE**&daddr=**ENDLATITUDE**,**ENDLONGITUDE**&ie=UTF8&0&om=0&output=**kml**"*

and receives, a KML file. This file is parsed by the pairs (latitude, longitude) of directions, existing on the file. For each one of this pairs, MyOverlay class is invoked. If it is the first or last pairs of directions a picture is designed, if not, a line is drawing like is shown on presented code:

```
1.  canvas.drawLine(startPoint.x, startPoint.y, endPoint.x, endPoint.y, paint);
```

To draw the lines on this overlay, the drawLine() function is used, it takes as parameter the start and end points, using X and Y axis, to draw that line on the map.

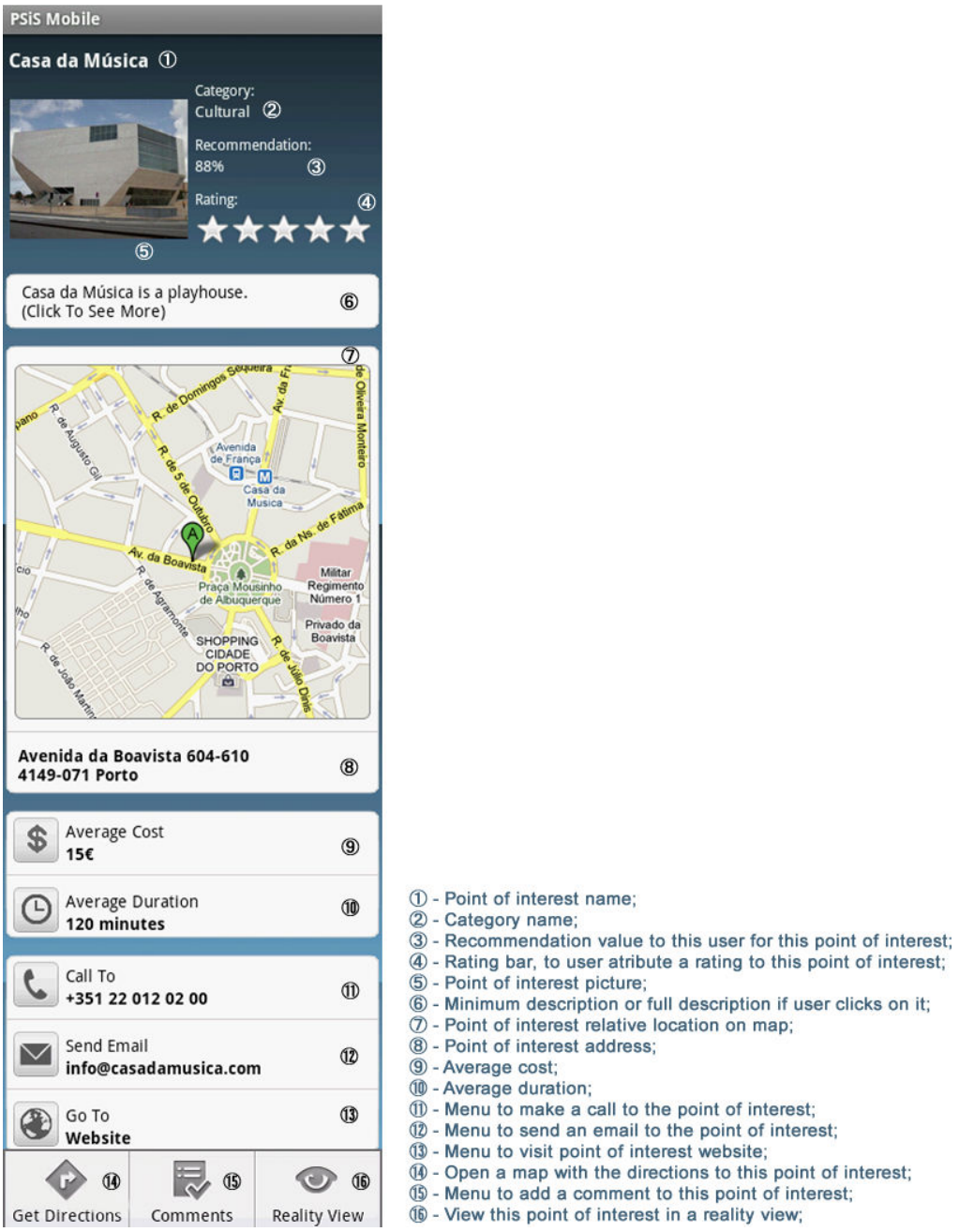


Figure 28 - POI Detail Activity

Apart MyRouteMap Activity, another Activity that user can goes is the POIDetail Activity, figure 28. To do this he simply selects the desired point of interest. POI Detail tends to give a simple overview of a point of interest, without heavy and unnecessary information. First, it shows a picture of the point of interest, his category, the percentage of recommendation for that point of interest and a rating bar to user classify it. After, a short description is shown, if user wants to see a full description, only has to click above short description and automatically the screen auto-updates to show more details about the sight.

A map, with sight location, is presented to the user for a better orientation. It is also shown point of interest address.

Additional information is also accessible to the user: Average Cost; Average Duration and Contact Information (telephone, email and website). If tourist wants to call to the sight, he only needs to click on the phone number. When the click is made an Intent is formed to perform a call. For email and sight website the procedure is the same.

Intent example:

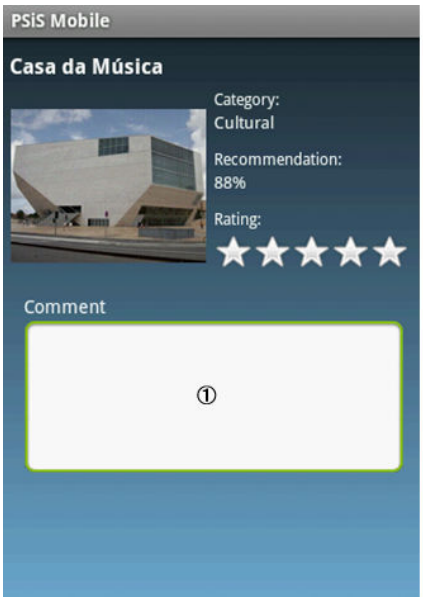
```
1.  Intent intent = new Intent(Intent.CALL_ACTION);
2.  intent.setData(+351220120200);
3.  startActivity(intent);
```

First of all a new Intent object is created along with the pretended action. In this case the CALL_ACTION makes a call to a certain number. In line 2 the call number is specified. With all the data defined, then is lunched that action, in other words, phone starts the calling to that number.

As seen on figure 24, user from this POI Detail Activity can interact with three more Activities: MyRouteMapActivity, POICommentsActivity and POIRealityViewActivity.

MyRouteMap Activity, as explained before, shows the directions to the point of interest from actual tourist position.

POIComment Activity, figure 29, for now only saves on database user comments to a point of interest. But in the future it will show the comments of the other system users. This isn't implemented because main application, PSiS, doesn't have already the social features working. However the comments are already being saved.



① - Comment area;

Figure 29 - POI Comments Activity

The third one is RealityView Activity, which provides an augmented reality of the point of interest and is explained on section 6.6.

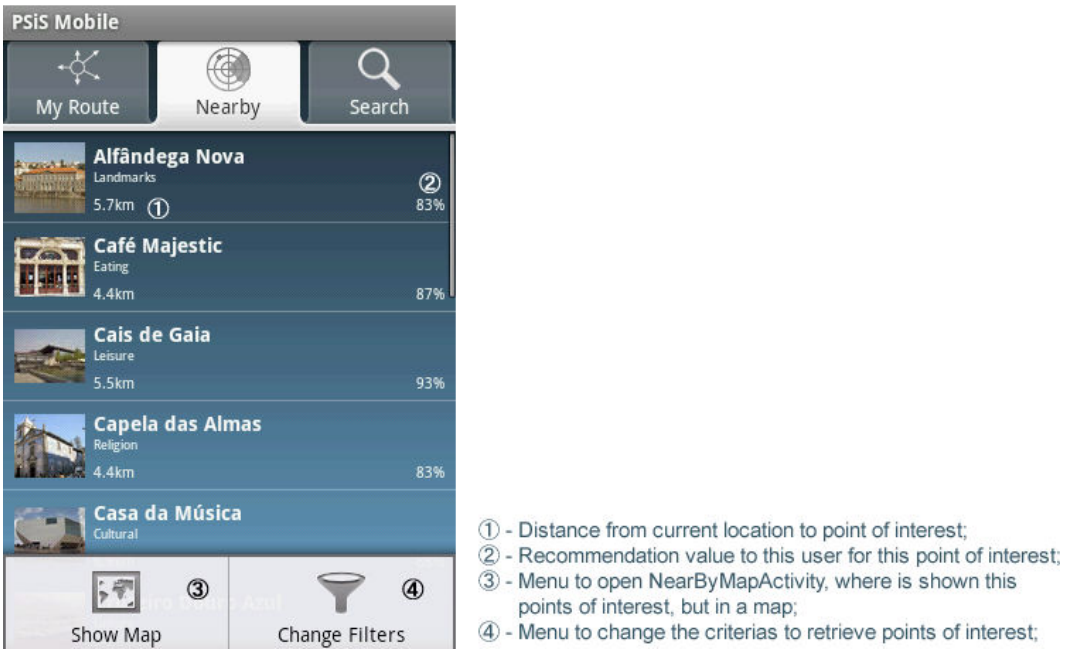


Figure 30 - Near By Activity

NearBy Activity, figure 30, is very useful and can be used when the tourist doesn't have a planned route. This screen presents points of interest nearby user, which are more recommended to him. The interface is very similar to MyRoute Activity, but instead of arrive and depart hours, it shows the recommendation percentage and the distance to the point of interest.

The distance between tourist location and points of interest is calculated using Haversine formula with current user GPS coordinates and point of interest coordinates.

The Haversine formula remains particularly well-conditioned for numerical computation even at small distances – unlike calculations based on the spherical law of cosines. It is an equation important in navigation, giving great-circle distances between two points on a sphere from their longitudes and latitudes. Basically it returns the distance in straight line between two points.

Haversine formula:

$$d = R.2.atan2(\sqrt{a}, \sqrt{(1-a)}) \quad (1)$$

Where d is the distance between the two points, R is the earth radius (used value was 6371km), $atan2$ is a variation of the arctangent function and a is given by the formula:

$$a = \sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1).\cos(\varphi_2).\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right) \quad (2)$$

Where φ_1 is the latitude of point 1, φ_2 is the latitude of point 2, λ_1 is the longitude of point 1 and λ_2 is the longitude of point 2.

This formula as a pseudocode, it's represented like this:

```
1.  R = 6371km
```

```
2. Δlat = lat2- lat1
3. Δlong = long2- long1
4. a = sin(Δlat/2)^2 + cos(lat1) * cos(lat2) * sin(Δlong/2)^2
5. c = 2 * atan2(√a, √(1-a))
6. d = R * c
```

First is defined the earth radius, on second line is calculated the difference between the second point latitude and first point latitude. On third line, is calculated the difference between longitudes, of the same two points.

On fourth line is calculated the 'a' formula. Then haversine formula is calculated to finally be multiplied with the earth radius, returning the distance between the two GPS coordinates in a straight line.

From NearBy Activity user can take three actions, show point of interest details (calling POIDetail Activity), change from this list view to a map view (NearByMap Activity) where is shown points of interest locations on a map and change content filtering on FilterDefinitions Activity.

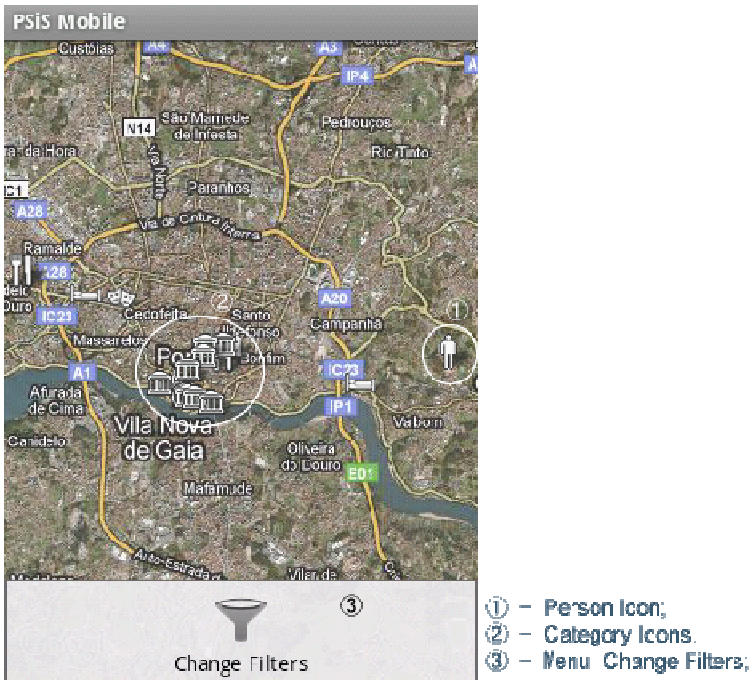


Figure 31 - Near By Map Activity

NearByMap Activity, figure 31, is likely the Nearby Activity, the only difference is that the information is shown on a map, and each of point of interest is representing by an icon. The icon is corresponding to his category, for example, for a restaurant category an icon with a knife and a fork is shown to represent that sight. When user click on a icon, POI Detail Activity appears to show sight information.

Current user position is also represented on map by a "person" icon. In this screen, user can go to FilterDefinitions Activity, as also is possible on NearBy Activity.

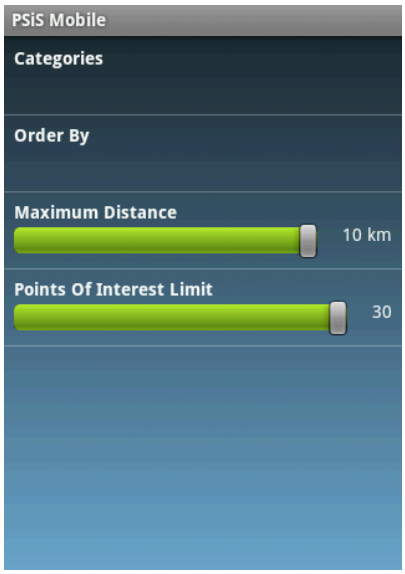


Figure 32 - Filter Definitions Activity

The system already makes an information filtering, based on content filtering. It takes all the data on database and filters information upon certain criteria. These criteria are: Weather Conditions, as said earlier, based on the current weather it can show open air sights or not; Point of interest Schedule according to system current hour, if a point of interest is already closed, it will not be shown; and system hour, if it is lunch or dinner hour, the system will rather recommend restaurants instead of sights to see.

Besides these criteria, user can filter information, using Filter Definitions Activity (figure 32), based on four criteria:

- He can choose what categories he wants to see, *e.g.*, if he only wants to see nearby restaurants;
- Order presented information, only on NearByActivity, by name, distance and rating;
- Maximum point of interest distance in relation to him, this can take a value from 0.1 km to 10 km;
- Amount of presented points of interest, from only 1 to 30 sights;

The Search Activity is presented on figure 33. It allows the user to search through a set of points of interest by name, keyword or a word presented on point of interest description.

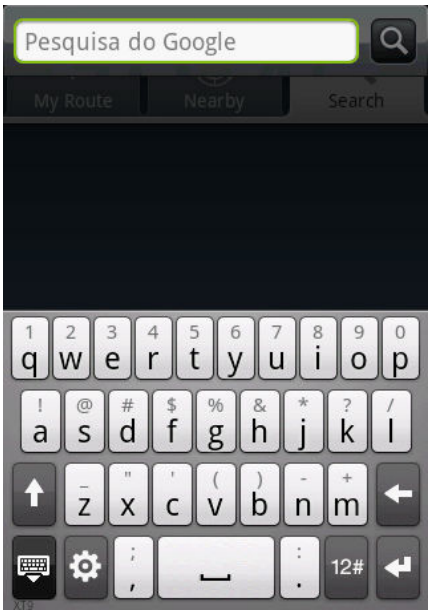


Figure 33 - Search Activity

This user interface was created using the Android Search Dialog API. This is useful to avoid a manual implementation of user interface items (*e.g.*, a search box). When the user invokes the search, a search dialog appears at the top of the screen and, in the screen bottom, a virtual keyboard is shown. When the user executes the search, PSiS Mobile application uses the input data to search for information in the database.

In the latest versions of Android (from 1.6) it is possible to do voice search.

6.2. DAL

PSiS Mobile DAL is a Java library that provides simplified access to data stored in database or in the server. This layer was made to have two access types and, in an intelligent way, chose what source to use. If there are many data to show to user, it only uses local database to retrieve that data, but if no sufficient data is available to show and an internet connection is available, a request is made to the server. Allowing user interface layer to be created with a high level of abstraction, because in both ways is returned a reference to a complete object with all attributes presented on the desired table.

DAL is composed by two main *packages*, as can be seen on figure 34, in *data package* are all the objects that represent the tables, and that are filled with data from *data.database package*. This last package includes all the classes to create the database on mobile device. The first time that application runs on a mobile device, it creates the model necessary to store data. Each one of these tables are responsible for create it on database and for getting data from mobile database or server database through middleware. This classes implements methods to create the table, insert, update and delete any data. All of application packages and their classes can be seen on the Annexes section.

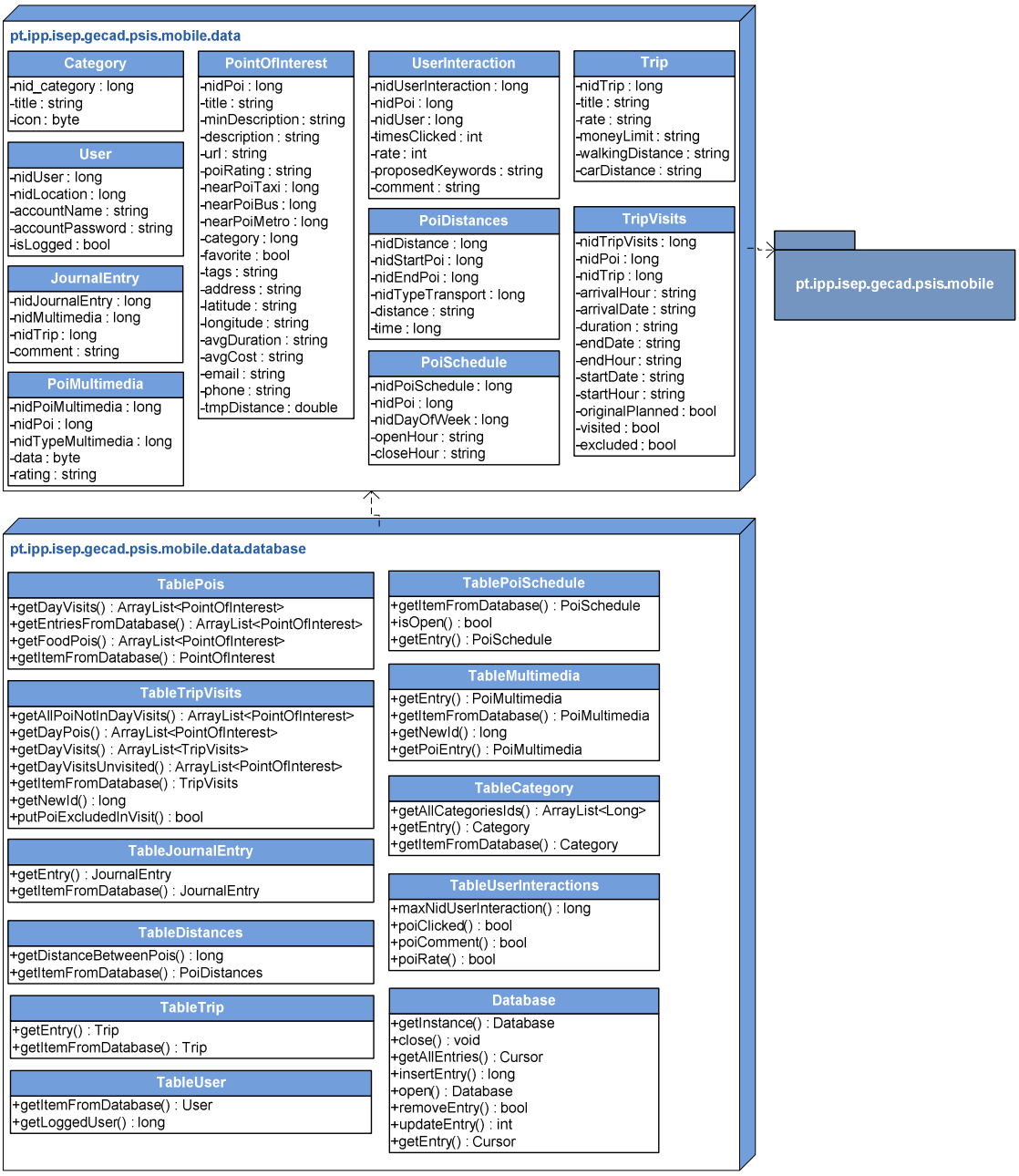


Figure 34 - DAL Structure

To speed up the appearance of the results to the user a staging database is used, like said earlier. Database schema is presented on figure 35 and a brief description of the principal tables is made after.

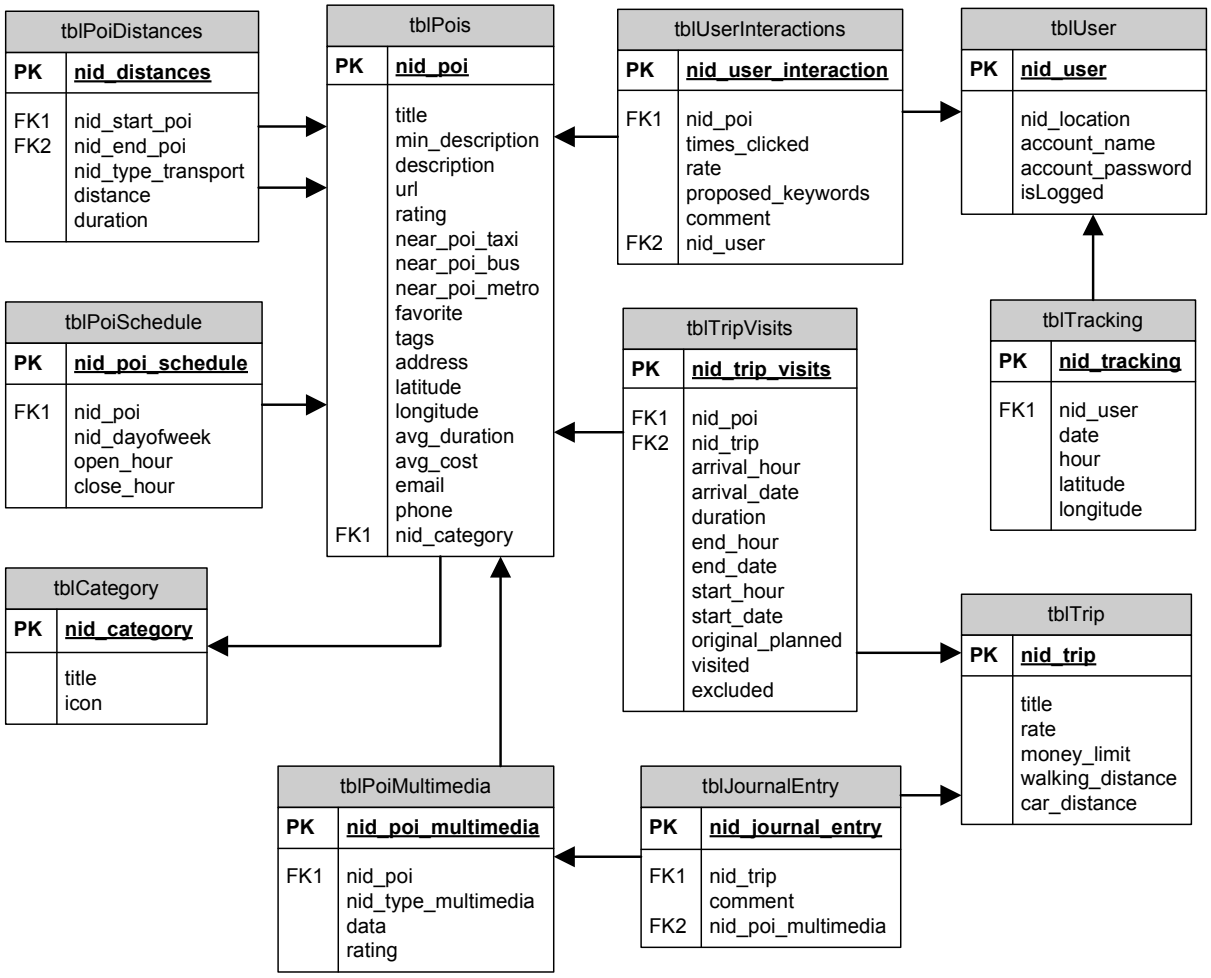


Figure 35 - Database Schema

Table tblPoiDistances is used to save the distances between points of interest. With this distances stored on the database, application don't need to calculate them all the time.

Planned trip for a day, is stored on table tblTripVisits, being changed according to time that tourist spends on it, or if a re-plan must be done. This is the principal table to send to the server, because it has the feedback of the planned tour. Also table tblUserInteractions is a "feedback" table. It records how many times user clicked on a point of interest, if a user click several times on a point of interest it can be interpreted that he likes it. Comments and rate giving by the user is also stored there.

The other tables are simple to understand and are basically a copy of the main database.

6.3. Middleware

When the application is installed on the mobile device only the database is created on it, no data is inserted. So it's necessary to pass information to tourist mobile device. It is here that middleware starts his working. First of all, when user makes login for the first time and an internet connection is available, mobile device will request the data to middleware, then the middleware will get data from server database for that user.

Middleware was developed on Java language, using Java Servlets API, and is running on a Tomcat server. Like was discuss on chapter 5, the chosen protocol was HTTP and the data is sent on Protocol Buffers messages. The request is intercepted by the Servlet and the information is treated, then when there are results a response is given to the mobile client, using the Protocol Buffers messages.

This module is constituted by six piece, or sub-modules, as demonstrated on figure 36. It has a Staging Database, running in a Microsoft SQL Server. Middleware database is identical to mobile database, only includes in all tables three more fields:

- Date when the information was sent to the mobile device;
- User identification, to register the user that receives the data;
- A Boolean field, that represents the necessity or not, to update that data on mobile device, *e.g.*, if user meanwhile generates a new route, this field turns to *true* value to indicate that it's necessary to update it on mobile device.

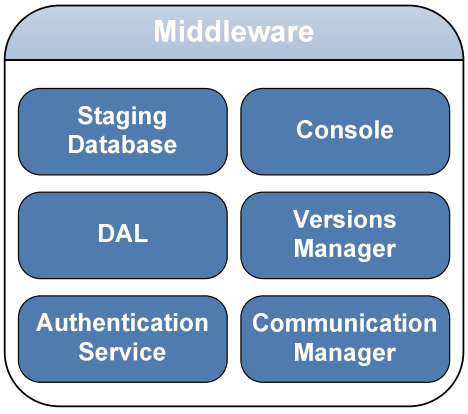


Figure 36 - Middleware Architecture

For all data that is transferred to the mobile device, a copy is saved in the middleware. So it lets to have more control of data versions and doesn't overload mobile device with this hard task. The sub-module that is in charge of make this verification is the Versions Manager.

The third middleware sub-module is a Console where all the logs can be shown. Whenever there is a request and a response, it is saved in a log table, which can be consulted in this module using a graphic interface (a web page).

A DAL is also present to create an abstraction from data present on middleware and server database.

An Authentication Service is present in this architecture, but not yet implemented. What it is going to do is to ensure data security, trying to not letting third parties to access unsuitable data. Although the transmitted data aren't very sensitive, apart from login, no private data is sent on requests or responses.

Communication Manager is the six and more important sub-module. It is responsible for all the messages, it knows what to do with a received message and to redirect them to the appropriate

location. Also if Internet connection is unstable the system adapts to it, sending/receiving only one result at a time instead of a list with many results. This feature is useful to not exist loss of information. Protocol Buffers messages are a serialization format with an interface description. To create these messages, first of all they must be described in a file like this:

```
1.  package pt.ipp.isep.gecad.psis.middleware.messages.poi;
2.  option java_outer_classname = "PoiMessageProtos";
3.  message Poi {
4.    required double nid_poi = 1;
5.    required String title = 2;
6.    required String min_description = 3;
7.    required String description = 4;
8.  }
```

First of all is indicated what is the *package* that the created class will be in. On second line is presented an option parameter, to specify class name. On third line is described what the message name is, in this case "Poi". After this, as presented on lines four to seven, is described which fields the message will have and what the type of them (String, double, integer, etc.). As well as the order of the fields (*e.g.*, 1, 2, 3, and so).

After making the file and saving it with .proto extension, it is necessary to compile the proto file using the Protocol Buffers compiler, which is available on API web page. This can be achieved using the following command:

```
protoc -I=$SRC_DIR --java_out=$DST_DIR $SRC_DIR/poimessage.proto
```

After that a PoiMessage.java file is generated in the specified destination directory. In this file has been generated a class called PoiMessageProtos, nested within which is a class for each message specified in PoiMessage.proto. Each class has its own Builder class that is used to create instances of that class.

Both messages and builders have auto-generated assessor methods for each field of the message. Messages have only getters while builders have both getters and setters. Here are some of the assessors for the Poi class (implementations omitted for brevity):

```
1.  // required double nid_poi = 1;
2.  public boolean hasNidPoi();
3.  public double getNidPoi();
4.
5.  // required String title = 2;
6.  public boolean hasTitle();
7.  public String getTitle();
8.
9.  // required string min_description = 3;
10. public boolean hasMinDescription();
11. public String getMinDescription();
12.
13. // required string description = 4;
14. public boolean hasDescription();
15. public String getDescription();
```

In this case is generated two methods for each field presented on protos file. The first method is used to know if in the message is present on that field or not. The second is used to retrieve the information of that field.

Meanwhile, Poi.Builder has the same getters but it includes also the setters to permit the inclusion of information to the respective field:

```
1.  // required double nid_poi = 1;
2.  public boolean hasNidPoi();
3.  public double getNidPoi();
4.  public Builder setNidPoi(double value);
5.  public Builder clearNidPoi();
```

The message classes generated by the Protocol Buffers compiler are all immutable. Once a message object is constructed, it cannot be modified. To construct a message, a builder must be construct first, then message information must be filled with the set methods, then the builder's build() method is called.

Here's an example of how as instance of Poi is created:

```
1.  Poi Casadamusica = Poi.newBuilder()
2.                        .setNidPoi(12)
3.                        .setTitle("Casa da Música")
4.                        .setMinDescription("This is a minimum description")
5.                        .setDescription("This is a description")
6.                        .build();
```

In this piece of code is created an instance of a Poi message, using the Builder generated in this class. On line two to five, is putted the desired information for that points of interest. On line six is invoked the method build() to compile the message.

Finally, each Protocol Buffers class has methods for writing and reading messages using the Protocol Buffers binary format. These are the two necessary methods to do the writing and reading:

- void writeTo(OutputStream output): serializes the message and writes it to an OutputStream;
- static Poi parseFrom(InputStream input): reads and parses a message from an InputStream.

In figure 37 is presented a sequence diagram which shows the system steps since user asks for points of interest nearby a certain location (latitude, longitude) until he has the result. Tourist starts to go to NearBy Activity that requests a quantity of points of interest to the TablePoi class from DAL, that will respond with an array of points of interest. The DAL, as before explained, first inquires mobile database and, if there aren't the desired quantity (defined in Filter Definitions Activity) of points of interest and an internet connection is available it will request middleware, using the Protocol Buffers GetPois message. Then that amount of points of interest is sent to the mobile device using a Protocol Buffers SendPois message. If internet connection isn't available, the application only shows the points of interest present on mobile database. More information about middleware class's structure can be seen on Annex B.

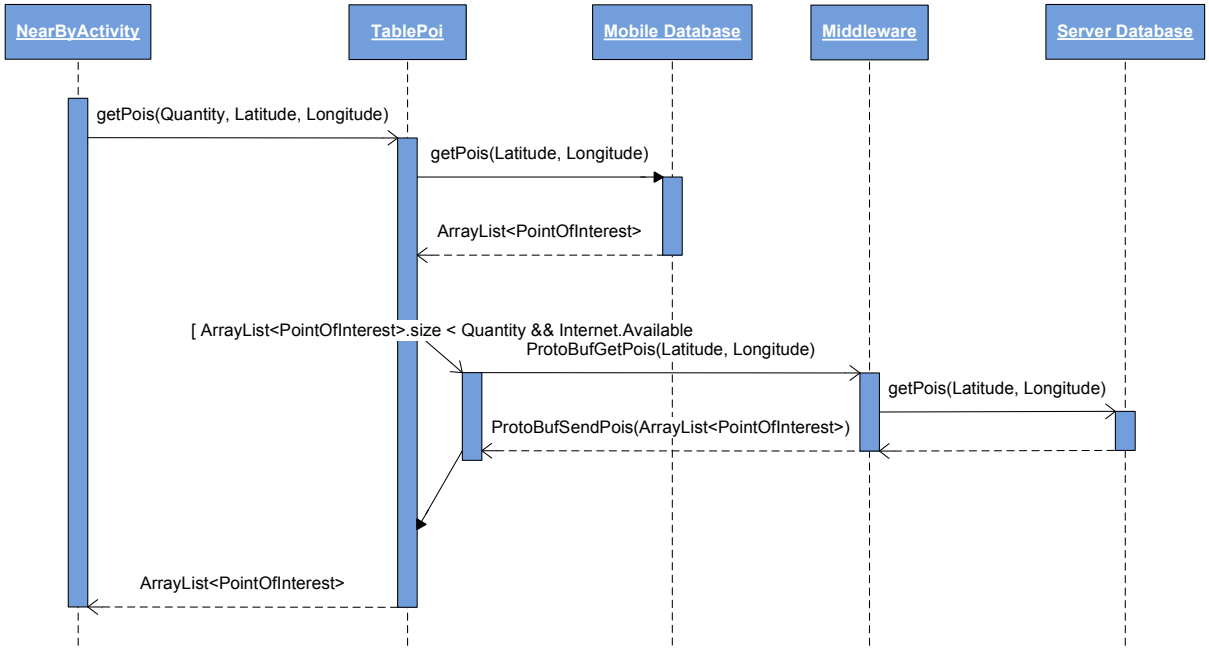


Figure 37 - Sequence Diagram Get Points of Interest

There are several messages created to make the communication between the server and the mobile device. Each of one was created using the Protocol Buffers compiler. The following messages were created to make request to the server:

- doLogin(String username, String password): To send the information that user have introduced to make login;
- getPois(Categories, Distance, Order, maxResults, UserLocation): To obtain points of interest that correspond to the sent parameters;
- getTripVisits(Date): Message to get the points of interest to visit in a specific date;
- doReplan(PoisVisited, PoisLeft, PoisExcluded, UserLocation): When a more elaborated planning must be done, this message invokes a replanning on server planning module, sending the relevant data to make a planning;
- sendUserInteraction(UserInteraction): Whenever a user interaction is made it is registered on mobile database, but when an internet connection is available it send that information to server to the information be updated;
- sendPhoto(MultimediaItem): Users can take photos on the go, so they must be sent to server using this message. The method to send them is equal than the previous described message.

When a request is made a response must be sent, so were implemented the following messages to send to the mobile device:

- doLoginResponse : Boolean - Indicates if the login was successful or not;

- sendPois : ArrayList<PointOfInterest> - Transfer the desired points of interest, according to the request. If a point of interest returned by this array already exists on mobile database, only his id field is filled;
- sendTripVisits : ArrayList<TripVisits> - Devolves the TripVisits to visit on a date, sending the schedule and the points of interest;
- sendUpdates : HashMap<Table, Arraylist> - This message was created to mobile device database be updated, whenever is necessary.

6.4. Tracking Module

This module, like the name indicates, is capable of observing a person on the move and sequentially record or read location data. Whenever an update of person GPS coordinates occurs this module is invoked. It is a simple class, that is capable of save tourist route and pop up information about what is happen. Figure 38 contains a flowchart with the actions that are executed when coordinates changes.

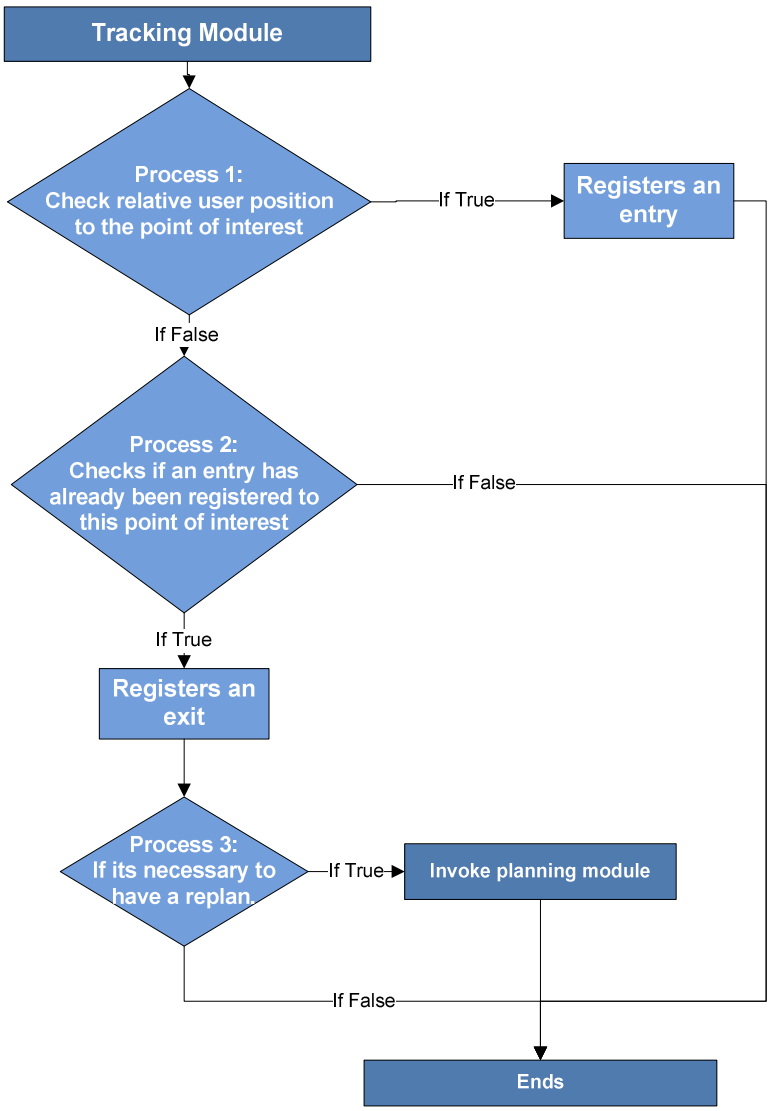


Figure 38 - Tracking Module Flowchart

When Tracking class (through method checkProximity(Location)) is invoked, it does some verifications, like is seen on figure 38.

First of all is compared the tourist position relatively to the location of the first point of interest present on route planning. If tourist is less than twenty meters from the point of interest, then is registered an entry on that point of interest and a dialog is shown to the user to question if he wants to see point of interest details. This value was chosen because of GPS accuracy that is about fifteen meters. But there is a limitation, the application only have the GPS coordinates of the entrance for the point of interest and not the coordinates to define point of interest area. Second decision is only raised if the first one fails. It checks if an entry has already been registered for that point of interest and tourist is left the building (if he was gone away twenty meters from that POI). Tourist direction is given using mobile device compass and the last given GPS coordinate, where a comparison between the last known location with the actual location is done. Thus, the direction vector of travel is known. If false, then process ends and nothing is done. If true, then an exit from that point of interest is registered and a dialog is shown to user classify and give a comment for that point of interest. Next it is checked if a replanning is necessary. Replanning module is only invoked if user is ahead or behind schedule more than 10 minutes, what it does is explained on section 6.5. This whole process is executed without user intervention.

6.5. Replanning Module

When tourists are travelling the planning conditions can change for innumerous reasons. For instance, if a point of interest is already closed, if there is more people on the queue to buy tickets and it takes too long, if tourist is tired and don't want to walk more or simply it take too long to arrive to the sight, etc.. Because these constrains can happen on the field, a replanning algorithm was implemented. It is simple and fast since tourist wants a fast response to his problem. Since mobile devices are slower than a normal computer, a complex algorithm like a travelling salesman problem wasn't implemented.

Instead, a decision tree based solution was developed. This is a decision support tool that uses a tree-like graph of decisions and their possible consequences. So, it was managed to have a fast algorithm that request low system resources and the response is almost instantaneous. It computes a decision in about 100 ms, being almost imperceptible to user.

This module is part of Context Service and complements the Tracking Module. When this module is invoked by tracking module, it starts the interactions checking if tourist is ahead or behind schedule. These are two different situations, if he is ahead schedule a point of interest can be inserted between the last visited point of interest and the next to visit. If he is behind, more problems can occur as the next point of interest is already closed and other situations.

As can be seen in figure 39, the algorithm starts in decision 1. If is behind schedule, decision 2.1 is executed, it verifies if there is any point of interest after the last visited. If not, nothing will be done to schedule, because there aren’t more points of interest to visit. If there are, passes to decision 2.2 that

verifies if point of interest schedule is compatible with the new arrival and departing hour and the delay isn't more than 30% of visiting time for that point of interest. If not, nothing is done. If it bothers and it isn't a restaurant, point of interest is removed from original trip and a new point of interest will be recommended that can be inserted on left time, advancing to decision 1.1. If it's a restaurant and is still open, nothing is done, because food points of interest are especially to this context. On the original planning user can choose in what time he wants to eat, so that schedule doesn’t must be changed. If it is closed, another restaurant will be recommended. The best choice is returned by process 1.6.

When a user is ahead schedule or a new point of interest must be recommended, decision 1.1 is executed. It tries to get from database only the points of interest that were excluded earlier from this route. These are tested first because it supposedly has more rating to this user, than the others. On decision 1.2 is obtained from database all points of interest that have a visiting time smaller than the existing left time gap. If there are any results on each of this process's, a decision is called to check the viability of the inclusion of points of interest on the planning. This, "Is viable for that schedule", decision what does is, for all the retrieved points of interest, see if travel time from current position to new point of interest, plus visiting time, plus travel time from new point of interest to next point of interest, is equal or less than the ahead schedule time. If there isn't any point of interest then backs to the previous process. Otherwise, if only one sight fits on that rules, then it is returned. When exists more than one point of interest that is viable to introduce on planning only one must be chosen through getBestChoice method.

From a list of points of interest, getBestChoice method assigns a planning rating to each one. This rating is calculated based on a rule:

- If spent time, travelling times plus visiting time is equal to gap, then it receives 100 points. For each minute less, one point is subtracted;
- The best point of interest recommendation rating to the user, receives 100 points, each percentage less represents one point less;
- The point of interest with less travel time receives 100 points, and the others receives less one point for each minute more;
- The result is the sum of this three values dividing by three. It was chosen to equally these values with the same weight, because all of them have some importance.

If decision 1.1 and 1.2 don't retrieve any result, then decision 1.3 is invoked. It checks if start hour to the next point of interest planned on trip can be advanced. If yes, than it returns. If no, then will be checked if an interchange, between a point of interest already present on the plan, can be done, changing the position of the two points of interest.

This is a basic approach, and not an elaborated one, that would be heavy to the phone and would not bring great benefits. So, whenever the tourist leaves one point of interest is checked if it's necessary to make a replanning, not being necessary constantly recalculate the planning.

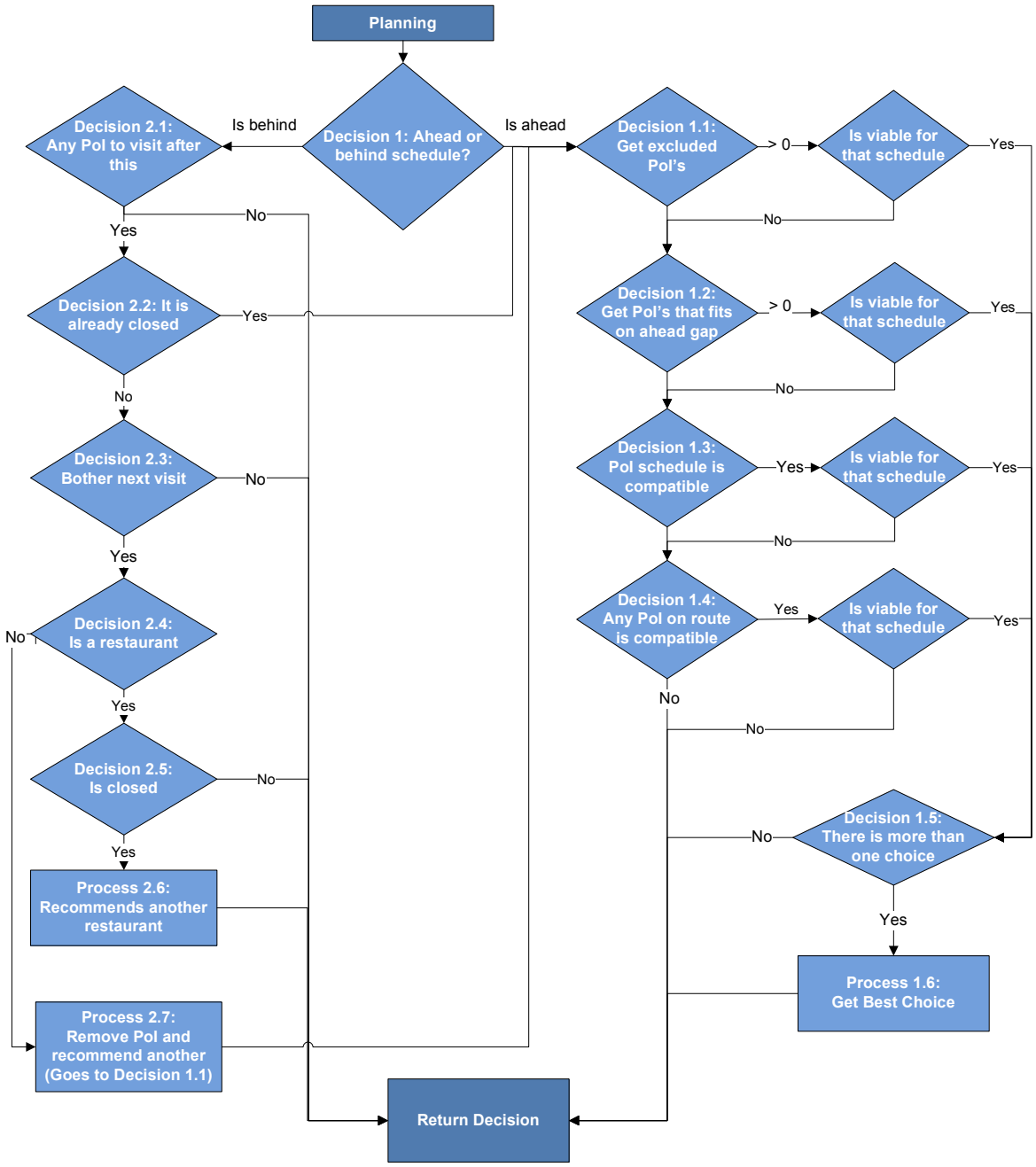


Figure 39 - Replanning Module Flowchart

6.6. Reality View

Reality is the state of things as they actually exist. This is a function of PSiS Mobile, see points of interest like they really are. Using the mobile device camera, tourist has access to see the sight on phone and take a picture. This function is available on menu "Reality View" on POIDetail Activity.

Figure 40 - Reality View Activity

After tourist takes a photo and save it on phone, picture is ready to send back for the server to be shared on the social network (not yet implemented). Comments can be done and sent as well.

But another interesting feature is available, that is augmented reality (figure 40). With augmented reality, reality can be augmented by virtual computer layers.

In this case, when user is in front of a point of interest, more information of that sight is shown, including pictures from other users on other occasions or perspectives.

To do this will be necessary a camera, a compass and a GPS receiver. The layers are computed through graphics 2D in canvas, that are easy to add over camera layer.

When developing an application that needs to draw something more specialized or control the animation of the graphic, a Canvas object can be used. Through Canvas it is possible to draw in a bitmap area that is placed in the application window.

Canvas class contains several methods to design standard objects that can be used, *e.g.*, drawText(), drawBitmap(), drawRect(), among others. The Drawable object has its own draw() method that takes the Canvas as an argument.

These elements are being created on a view over the image captured by the mobile devices camera. A class apart (DrawOnTopView that extends a View) was created for dealing with the creation of them. The coordinates for the location of the objects on the screen have to be given, because they are draw in a fixed position. An example of this is below:

```
1.  @Override
2.  protected void onDraw(Canvas canvas) {
```

```
3.    picture.setBounds(18, getHeight()- 10 - picture.getIntrinsicHeight(), 18 +
            picture.getIntrinsicWidth(),getHeight()-10);
4.    picture.draw(canvas);
5.    super.onDraw(canvas);
6.  }
```

On lines 1 and 2 of this code is defined what method is necessary to override. In this case is the onDraw method. This method is invoked when the objects are designed on the user interface.

On line 3 is presented the code that must be written to specify the location on the screen of the picture to draw. In the first two arguments are passed the coordinates of the top left vertex, in the X and Y axis respectively. On the third and fourth argument are passed the coordinates of the bottom right vertex, in the X and Y axis respectively. After defining these values, a picture is drawn on canvas (screen) like can be seen on line 4.

6.7. Results

After the development of this application a small test on the field was performed. First of all, a new user (with a specific profile) was created in the system, along with a trip planning for a day in Porto based on the user profile. Afterwards, the generated test route on PSiS framework was transferred to PSiS Mobile, and followed in real world. The planning results are depicted in a screenshot of the PSiS Website (figure 41) and in table 3.

Table 3 - Test Route

Point of Interest	Arrival Hour	Departure Hour
Pousada do Porto - Palácio do Freixo	-	9:00
Mercado Ferreira Borges	9:15	9:20
Estátua Infante D. Henrique - Príncipe Navegador	9:20	9:25
Palácio da Bolsa	9:25	10:15
Alfândega Nova	10:25	10:30
Igreja de S. Pedro de Miragaia	10:35	10:40
Estátua "O Cubo"	10:50	11:00
Ponte D. Luís	11:15	11:25
Cais de Gaia	11:45	12:00
Cruzeiro Douro Azul	12:00	15:30
Capela das Almas	16:00	16:10
Café Majestic	16:30	17:15
Mercado Bolhão	17:25	17:40
Câmara Municipal do Porto	17:50	18:00
Estátua de D. Pedro IV	18:00	18:20
Torre dos Clérigos	18:30	19:00
Restaurante Cufra	20:00	21:30
Casa da Música	21:30	23:30
Porto Palácio Hotel	23:25	-

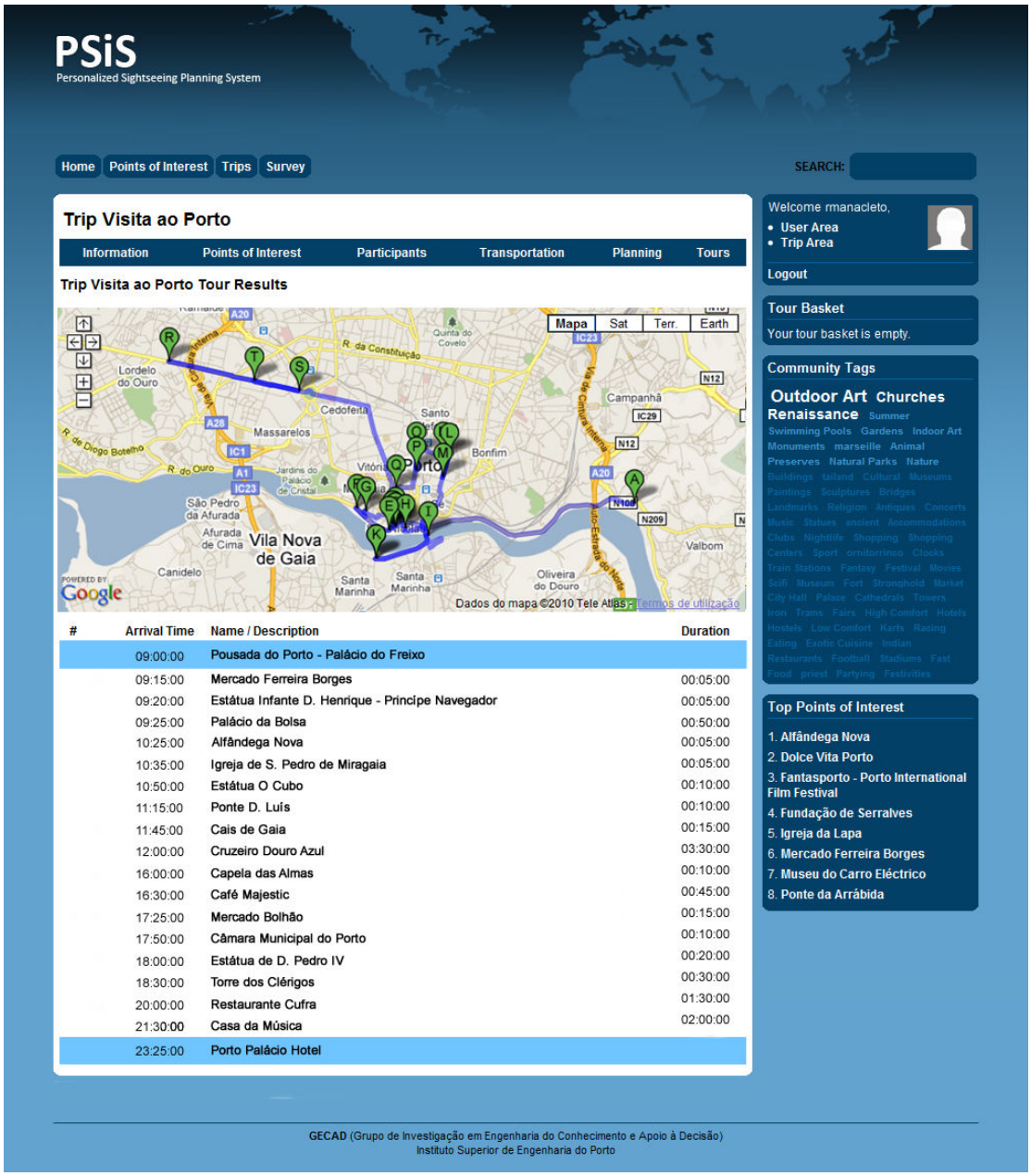


Figure 41 - Screenshot of Test Route in PSiS Website

PSiS Mobile didn't let us down and worked as expected. In order to test the replanning algorithm a simulation with some unscheduled delays was performed. For instance, in one of the evaluation tests when visiting "Palácio da Bolsa", where we only remain thirty minutes. So, according to a simulation where the user is ahead in time, for twenty minutes, the application behaved as expected.

The replanning algorithm (see section 6.5) started on Decision 1 (Ahead or behind schedule?) and went to Decision 1.1 (Get Excluded POIs), for more detail see figure 39. Because there aren't any excluded points of interest, the algorithm jumps to Decision 1.2 (Get POIs that fits on ahead gap) and selects all the points of interest present in the mobile device database, that have a visiting time less or equal than twenty minutes, and that aren't already included on the route (see Table 4).

Table 4 - Suitable Points of Interest for the Schedule in Figure 41

Point of Interest	Duration (minutes)	Distance To (minutes)	Distance From (minutes)	POI Rating (%)
Igreja de S. Francisco	20	1	3	83
Igreja do Bonfim	10	12	14	82
Farol/Capela de S. Miguel-O-Anjo	10	8	7	88
Sé Catedral	20	5	6	90
Igreja da Nossa Senhora da Vitória	10	4	4	80
Igreja de Santo Ildefonso	10	7	7	81
Igreja das Carmelitas	10	5	5	86
Jardim Botânico	15	10	10	82
Chafariz S. Lázaro	10	3	3	80
Ponte D. Maria Pia	10	10	12	88
Praça da Batalha	20	8	9	89
Ponte da Arrábida	20	10	8	94
Ponte Infante D. Henrique	8	6	6	96

All of these points of interest were open for visits on the desired schedule: from 9:55 to 10:25 (a gap of thirty minutes). Because there is more than one point of interest to fit on the route, the algorithm goes through Decision 1.5 (There are more than on choice) and the Process 1.6 (Get best choice) is called to get the best choice.

Table 5 contains the values calculated in Process 1.6, where the highest score is given to "Igreja de S. Francisco". Time Spent, POI Recommendation and Travel Time values were calculated according to the formula present in section 6.5.

Spend Time is calculated as follows:

$$TimeSpent = 100 - (GAP - (Duration + Distance\ To + Distance\ From)) \quad (3)$$

The result must be less or equal to one hundred. Using as an example the first sight, "Igreja de S. Francisco", time spent is given by:

$$TimeSpent = 100 - (30 - (20 + 1 + 3)) = 100 - (30 - 24) = 100 - 6 = 94$$

POI Recommendation is calculated based on the range of POI Rating values (e.g. as presented in table 4). In table 4, the highest POI Rating value, of 96, scores 100 points on POI Recommendation (see table 5). The remaining POI Recommendation values are calculated as follows:

$$POIRecommendation = 100 - (High\ POI\ Rating - POI\ Rating) \quad (4)$$

Following the previously presented example POI Recommendation is given by:

$$POIRecommendation = 100 - (96 - 83) = 100 - 13 = 87$$

Travelling Time is calculated based on the lowest travelling timing. This lowest value gets 100 points on Travelling Time. The remaining Travelling Time values are calculated as follows:

100 - ((Distance From + Distance To) - Lowest Value). Using the second point of interest on the list, "Igreja do Bonfim", as example the travel time is:

TravelTime = 100 - ((Distance From + Distance To) - Lowest Value (5)

Using the second point of interest on the list, "Igreja do Bonfim", as example travel time is given:

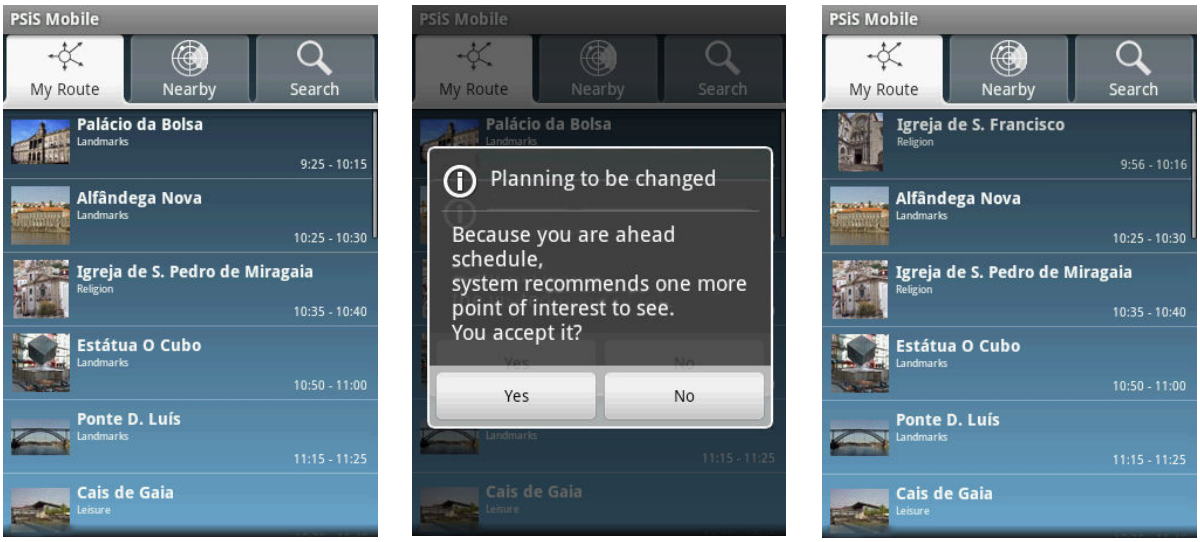
TravelTime = 100 - ((12 + 14) - 4) = 100 - (26 - 4) = 100 - 22 = 78

Table 5 - Best Choice Results For Suitable POIs on Table 4

Point of Interest	Time Spent (%)	POI Recommendation (%)	Travel Time (%)	Total
Igreja de S. Francisco	94	87	100	94
Igreja do Bonfim	-	86	78	55
Farol/Capela de S. Miguel-O-Anjo	95	92	89	92
Sé Catedral	-	94	95	63
Igreja da Nossa Senhora da Vitória	88	84	96	89
Igreja de Santo Ildefonso	94	85	90	90
Igreja das Carmelitas	90	90	94	91
Jardim Botânico	-	86	84	57
Chafariz S. Lázaro	86	84	98	89
Ponte D. Maria Pia	-	92	82	58
Praça da Batalha	-	93	87	60
Ponte da Arrábida	-	98	86	61
Ponte Infante D. Henrique	90	100	92	93

Although "Ponte Infante D. Henrique" has a higher POI Recommendation value, it was the second choice proposed by the planning algorithm since it is farther away from the previously visited POIs than the chosen point of interest. All points of interest with time of visit plus travel time higher than the gap value (thirty minutes), don't have a score on the Time Spent column.

figure 42 depicts the steps required for the user to accept the schedule replanning proposed by the planning algorithm. figure 42a contains the original route. Then, when the user leaves the point of interest, "Palácio da Bolsa", a warning appears asking the user if he wants to change the original plan (see figure 42b). When the user accepts the new proposal a new plan is presented, as in figure 42c.



a) b) c)

Figure 42 - Replanning steps in the PSiS mobile application

Although, in this test, we arrived behind schedule (15 minutes after 20h) at the Cufra restaurant, the schedule was not replanned since the point of interest Cufra is a special type of point of interest (as described in section 6.5.

Chapter 7

Conclusions and Future Work

This chapter will take the reader through a set of conclusions related to the development, evaluation and analysis of PSiS Mobile. Advantages and innovations brought by the PSiS Mobile application are some aspects to be discussed. Also, some future work is presented that may enhance and increase the application value.

The main objective of this thesis was to develop an application to a mobile device, which extends PSiS project to support effectively a tourist on his stay at a city. PSiS Mobile is a mobile recommendation system and planning support tool, which is designed to provide an effective support during the visit of a tourist, providing context-aware information and recommendations about points of interest to visit based on tourist preferences and his current context.

Since adaptation is a very important issue in recommendation systems, the implementation of new features, like context-aware, is very important to provide to it, more information about the user. Transparency is also an issue when providing recommendation, since for many users' knowing how the recommendation is given is an important factor to trigger trust.

Over the last decade, mobile devices have an incredible growth rate on sales and became very attractive to developers make applications to them, principally tourism related application. In that sense, a state of art on tourism mobile guides and tourism recommendation systems is presented, which are useful to assist a tourist on visiting a city for the first time and in planning his staying.

In order to achieve the main objective of this thesis the main concerns present in the development of a mobile application in a client-server environment was introduced. Because mobile devices have many limitations, the developed mobile application had to be carefully designed, mainly, because slow processors and low amounts of memory. A case study to find out which is the fastest and lightest way to exchange information between a server and a mobile client, was also presented. From this study, it was concluded that the best method was HTTP plus Protocol Buffers.

Although mobile devices have many constraints, PSiS Mobile was developed to provide a good user experience, giving tourists a fast and user friendly tool including context-aware adaption, a route planning system, augmented reality and built-in social networking features. All these features provide to the user important and significant details about what he is seeing or is about to see..

Because PSiS and PSiS Mobile are developed on different programming languages a middleware has been created. It ensures that the mobile device receives all the necessary data to assist the tourist in a fast and reliable way. The middleware was developed in Java using Servlets. All the data is serialized using Protocol Buffers and transferred using the HTTP protocol.

At the present PSiS is lacking the necessary data to provide consolidated results for mobile device utilization "on the field". The main feature that still isn't implemented in the PSiS main system is the handling of feedback data sent by the mobile device. The mobile application sends information to the server, but nothing is done, *i.e.*, no profiles are updated with retrieved feedback.

As future work, it will be needed, to have some feedback from real tourists about what must be improved on mobile application. Also it is necessary to complete the middleware, since it as some features that are not working. At this time, it doesn't call the planning algorithm for it carry out a new planning. Reality view and social network features need to be deeply explored, but only when server side supports it. Application must be adapted to an audio guide, as it is uncomfortable to be always looking to the mobile device screen, because it is small and forces the human vision. Also all the users' actions on the application must be recorded and interpreted, to define better the tourist profile. But actually without server support is hard to test and develop that type of features.

In the future, PSiS Mobile can be integrated with a system that provides indoor guides. This is currently the main goal regarding future work: a wearable wireless sensor network that is capable of locating users inside buildings. With this system, it will also be possible to recommend artwork inside a gallery according to tourist profiles.

References

[Almeida, 2009] A. Almeida, "Personalized Sightseeing Tours Recommendation System", The 13th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2009, Florida, USA, 2009.

[Anacleto *et al*., 2010a] R. Anacleto, N. Luz and L. Figueiredo, “PSiS Mobile”, International Conference on Wireless Networks (ICWN2010), Las Vegas, USA, 2010.

[Anacleto *et al*., 2010b] R. Anacleto, N. Luz and L. Figueiredo, “Personalized Sightseeing Tours Support Using Mobile Devices”, World Computer Congress 2010 (WCC2010), Brisbane, Australia, 2010.

[Biuk-aghai, 2004] R. Biuk-aghai, “MacauMap: Next Generation Mobile Travelling Assistant”, In Proceedings of Map Asia, 2004.

[Bradley, 2009] T. Bradley, PCWorld Magazine Online, [Online - 30/08/2010], [http://www.pcworld.com/businesscenter/article/173601/symbian_and_android_to_lead_mobile_os_market_in_2012.html], 2009.

[Chen, 2009] B. Chen, Android Army Pumped for All-Out Attack on iPhone, [Online - 30/08/2010], [http://www.wired.com/gadgetlab/2009/10/phones/], 2009.

[Espinoza *et al*., 2001] F. Espinoza, P. Persson, A. Sandin, H. Nyström, E. Cacciatore and M. Bylund, "GeoNotes: Social and Navigational Aspects of Location-Based Information Systems", Conference on Human Factors in Computing Systems, pp. 43-44, 2001.

[Google Inc, 2010] Google Inc., "Android Developers" [Online - 30/08/2010], [http://developer.android.com/index.html], 2010.

[Hariharan, 2008] K. Hariharan, "Extending Enterprise Applications to Mobile Devices", The Architecture Journal, 2008.

[Häubl and Dellaert, 2004] G. Häubl and B. Dellaert, "Electronic Travel Recommendation Agents and Tourist Choice", University of Alberta, School of Retailing, 2004.

[Hinze and Buchanan, A. Hinze and G. Buchanan, "Context-awareness in Mobile Tourist

2005] Information Systems: Challenges for User Interaction", International Workshop on Context in mobile HCI at the 7th International Conference on Human Computer Interaction with Mobile Devices and Services, Austria, 2005.

[Klante *et al*., 2004] P. Klante, J. Krosche and S. Boll, "AccesSights - A Multimodal Location-Aware Mobile Tourist Information System", in Proceedings of the 9th International Conference on Computers Helping People with Special Needs (ICCHP'2004), Paris, France, 2004.

[Kropfberger *et al*., 2007] M. Kropfberger, R. Tusch, M. Jakab, J. Köpke, M. Ofner, H. Hellwagner and L. Böszörmenyi, "A Multimedia-Based Guidance System for various Consumer Devices", Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WEBIST '07), pp. 83-90, Barcelona, Spain, 2007.

[Krosche *et al*., 2004] J. Krosche, S. Boll, and J. Baldzer, “mobiDENK – Mobile, Location-Aware Multimedia Support in Modern Monument Conservation,” IEEE Multimedia, vol. 11, no. 1, 2004.

[Luz *et al*., 2010] N. Luz, R. Anacleto and A. Almeida, “Tourism Mobile and Recommendation Systems - A state of the art”, International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE2010), Las Vegas, USA, 2010.

[Malaka and Zipf, 2000] R. Malaka and A. Zipf, "Deep map - Challenging IT Research in the Framework of a Tourist Information System", Springer Computer Science, Wien, New York, pp. 15-27, 2000.

[Nielson, 2010] Nielson, Smartphone Penetration, [Online - 30/08/2010], [http://blog.nielsen.com/nielsenwire/online_mobile/android-soars-but-iphone-still-most-desired-as-smartphones-grab-25-of-u-s-mobile-market/], 2010.

[Parle and Quigley, 2006] E. Parle and A. Quigley, "Proximo - Location-Aware Collaborative Recommender", Adaptive Hypermedia 2006, Workshop on the Social Navigation and Community-Based Adaptation Technologies, Dublin, Ireland, 2006.

[Poslad *et al*., 2001] S. Poslad, H. Laamanen, R. Malaka, A. Nick and A. Zipf, “CRUMPET: Creation Of User-Friendly Mobile Services Personalized For Tourism", Second International Conference In 3G Mobile Communication Technologies, pp. 28-32, 2001.

[Pashtan *et al*., 2003] A. Pashtan, R. Blattler, A. Heusser and P. Scheuermann, "CATIS: A context-aware tourist information system", Proceedings of IMC 2003, 4th International Workshop of Mobile Computing, Rostock, Germany, 2003.

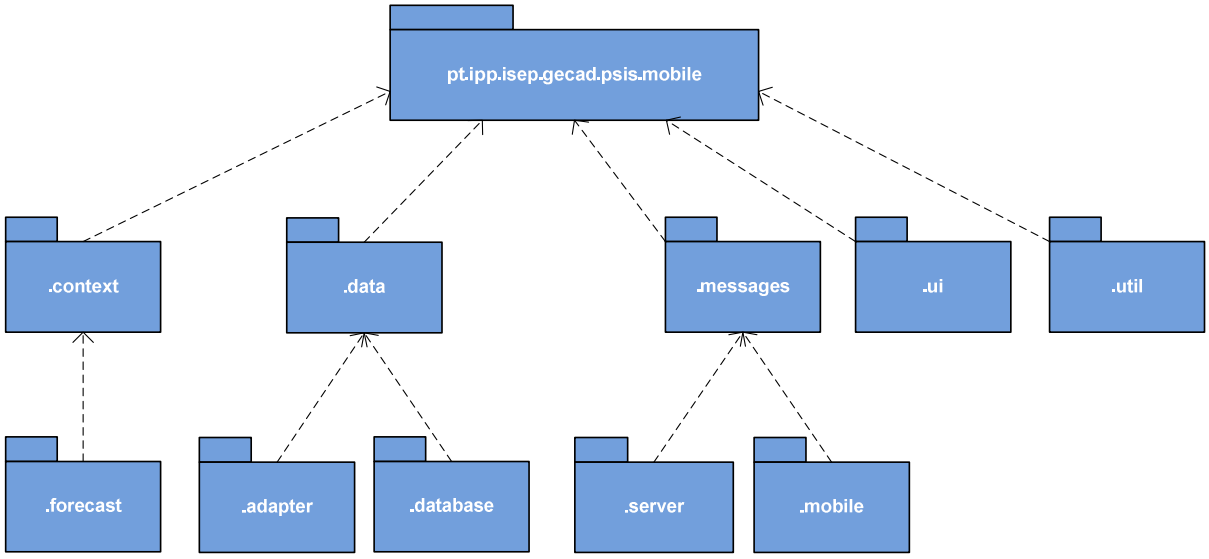
[Schneidawind, 1992] Schneidawind, J, "Big Blue unveiling", USA Today page 2B, November 23, 1992.

[Schneider and Schröder, 2003] J. Schneider and F. Schröder, "The m-ToGuide Project-Development and Deployment of an European Mobile Tourism Guide", Evolution of Broadband Services, Eurescom Summer 2003, Heidelberg, Germany, 2003.

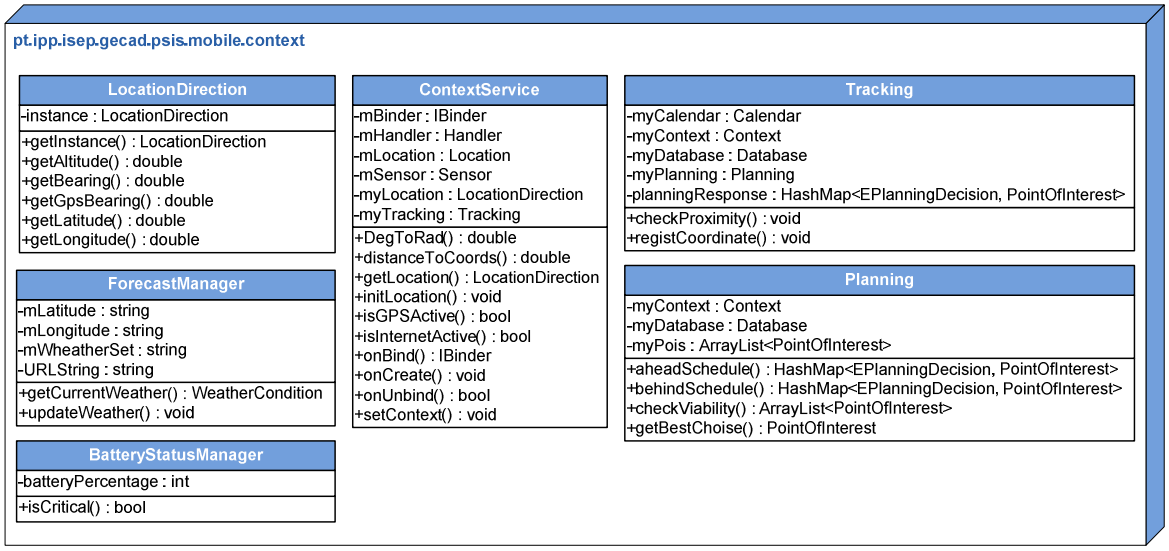
[Schuler, 2007] R. Schuler, "Mobile Application Architecture", 2007.

Annexes

Annex A - Package Diagram of Mobile Application

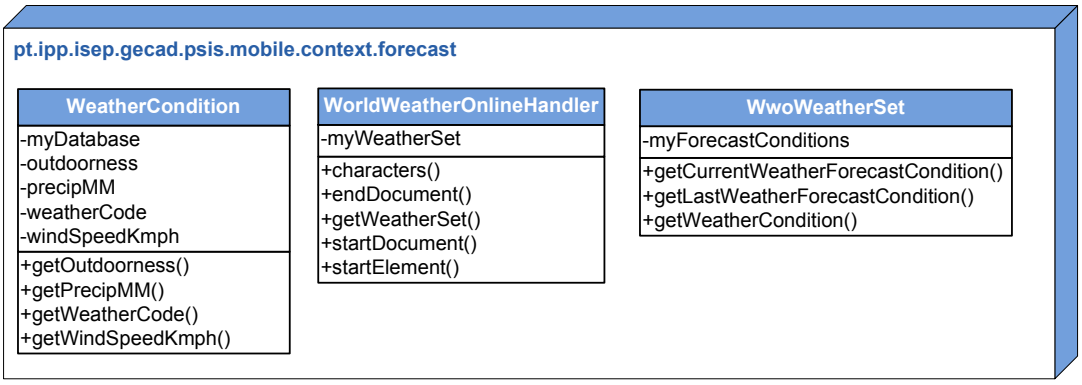


Annex A1 - Context Package Classes



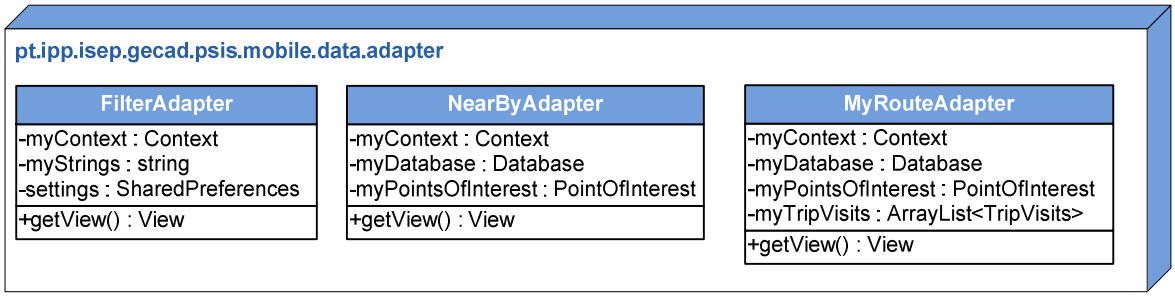
On package *pt.ipp.isep.gecad.psis.mobile.context* are present the classes that are responsible to retrieve mobile device context (Weather, Location, Battery Status, etc.). Also the class that creates the context service is here present.

Annex A2 - Forecast Package Classes



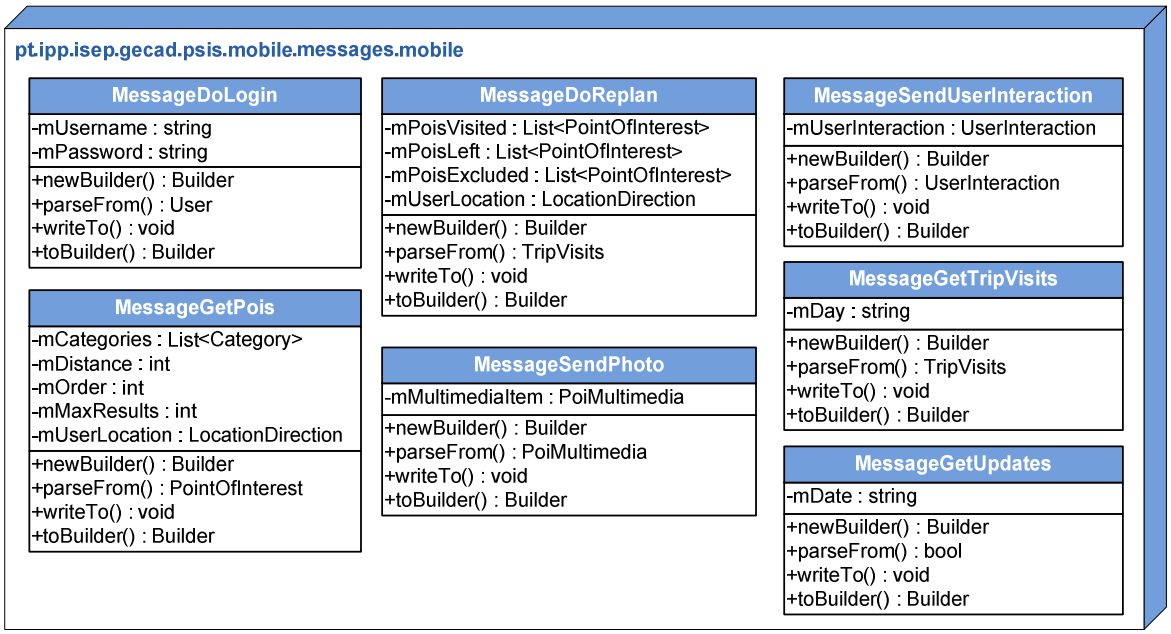
Inside *pt.ipp.isep.gecad.psis.mobile.context package*, exists the *package pt.ipp.isep.gecad.psis.mobile.context.forecast* where are the classes responsible for interpret the weather information retrieved by world weather online website.

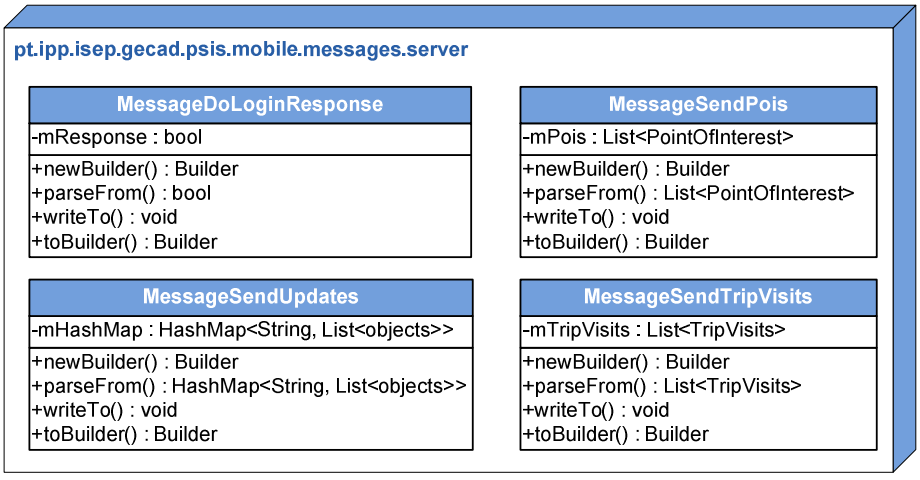
Annex A3 - Adapter Package Classes



pt.ipp.isep.gecad.psis.mobile.data.adapter package has all the classes responsible for the user interface lists for these Activities: Filter Definitions Activity, Near By Activity and My Router Activity.

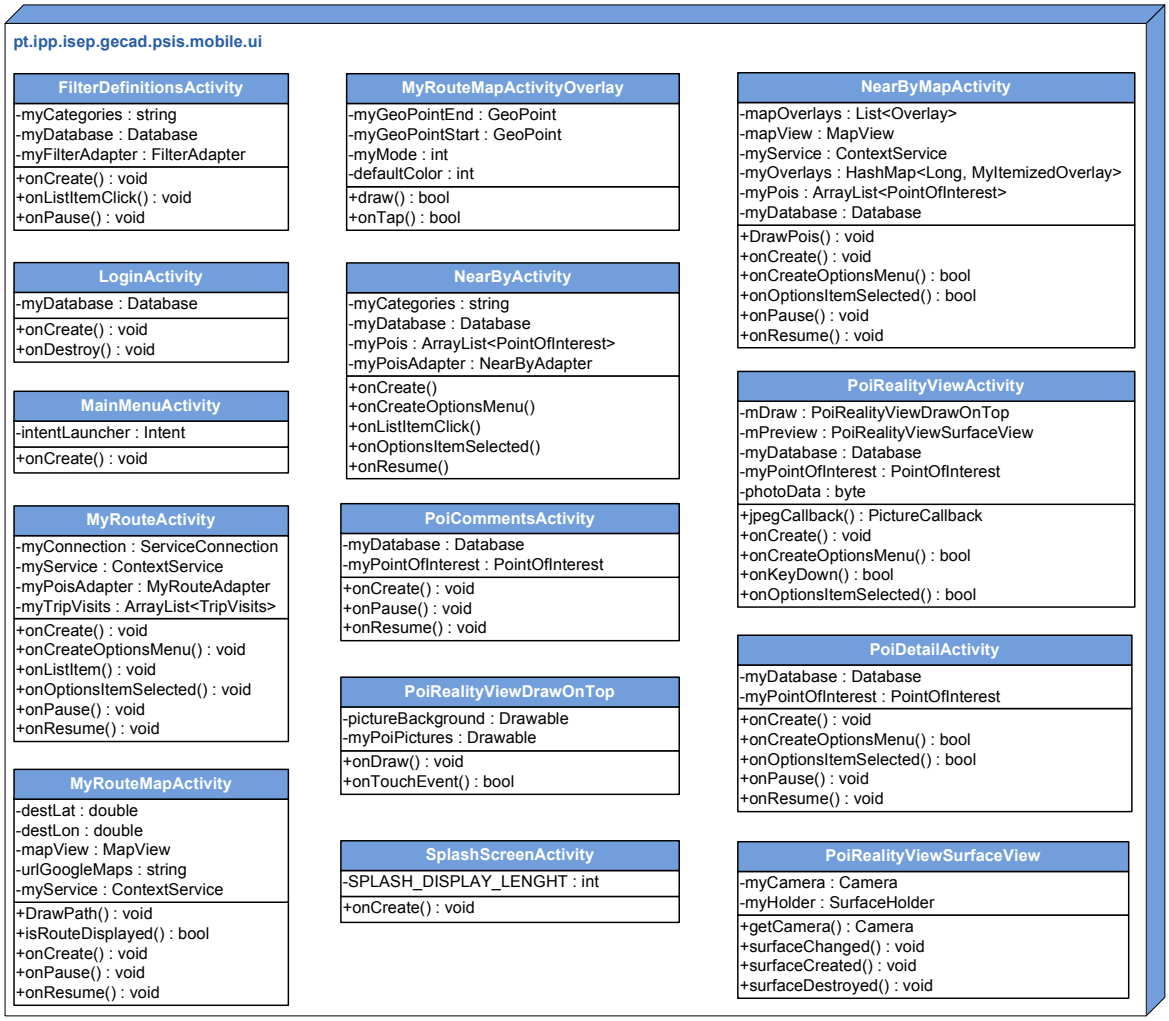
Annex A4 - Server and Mobile Messages Packages Classes





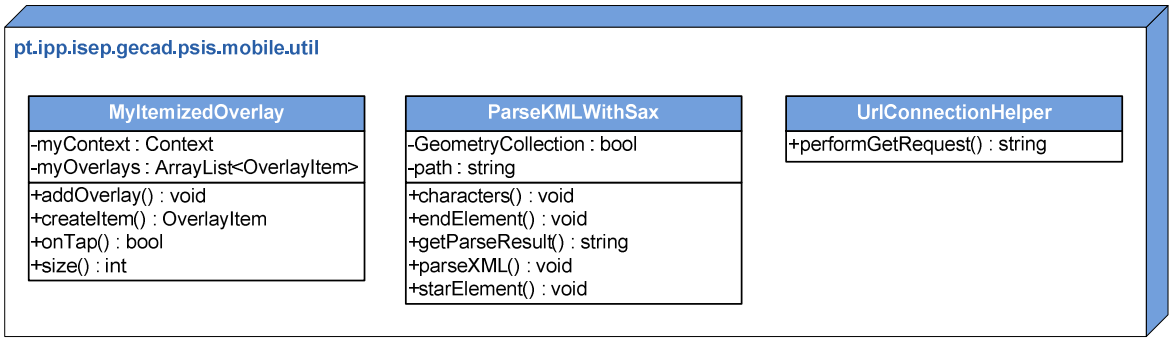
In *pt.ipp.isep.gecad.psis.mobile.messages*.mobile and *pt.ipp.isep.gecad.psis.mobile.messages*.server *packages* are present the classes responsible for the creation and interpretation of the messages exchanged between server side and mobile device.

Annex A5 - UI Package Classes



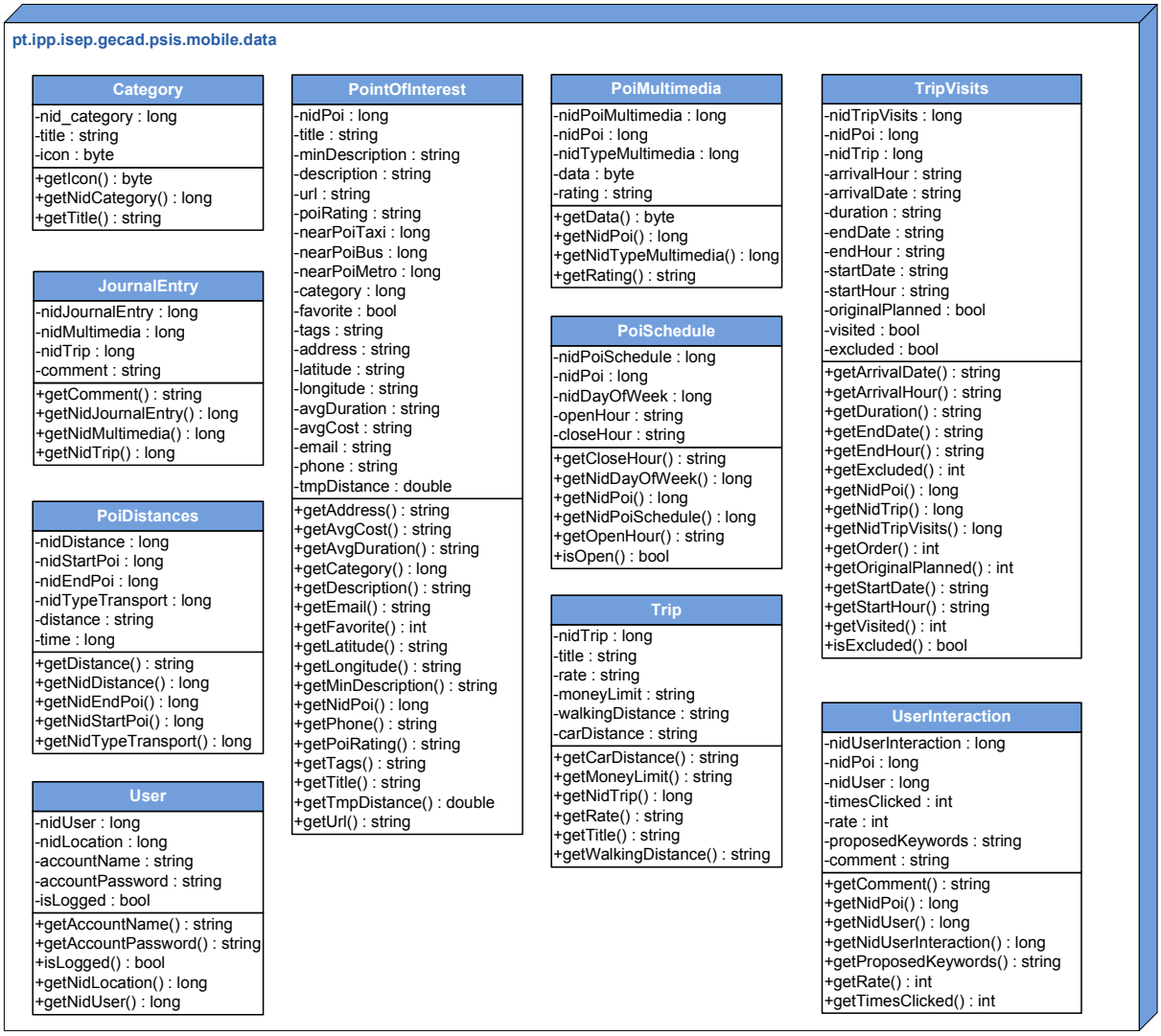
As the name indicates on *pt.ipp.isep.gecad.psis.mobile.ui* package is present all the classes that handle user interface for this application. Apart from PoiRealityViewDrawOnTop (draw objects over the image captured by camera), PoiRealityViewSurfaceView (handles camera phone actions) and MyRouteMapActivityOverlay (draws the route between two points on a map) each one of the remaining classes represents an Activity.

Annex A6 - Util Package Classes

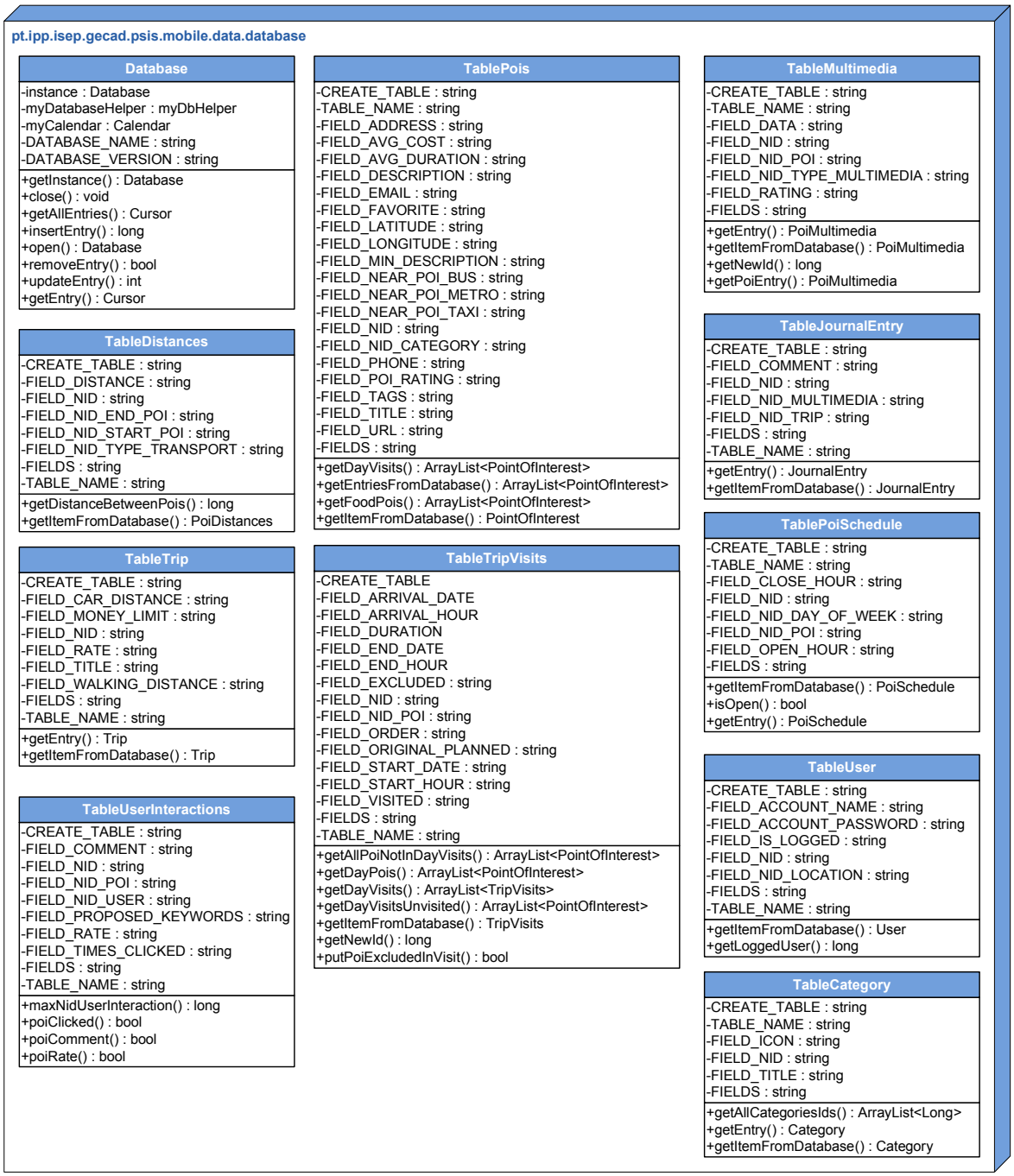


In *pt.ipp.isep.gecad.psis.mobile.util package* are present auxiliary classes for the application. MyItemizedOverlay draws the icons on a map overlay. ParseKMLWithSax parses the returned KML file from Google website retrieving all the path coordinates to then be draw over the map overlay. UrlConnectionHelper does a "Get Request" over a HTTP connection for a given URL.

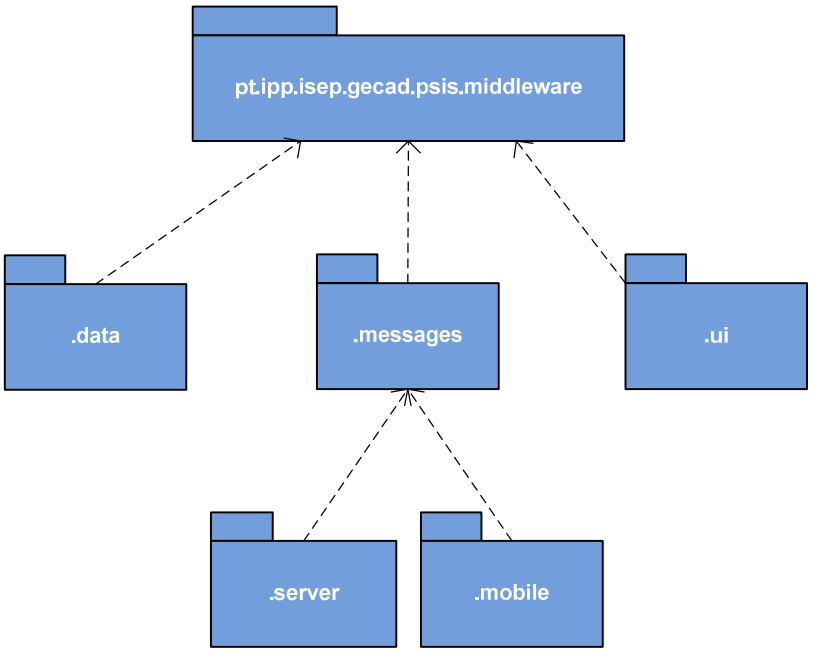
Annex A7 - Data and Database Packages Classes



Finally is presented the DAL *packages*, but this time with all the methods and attributes presented on those classes.

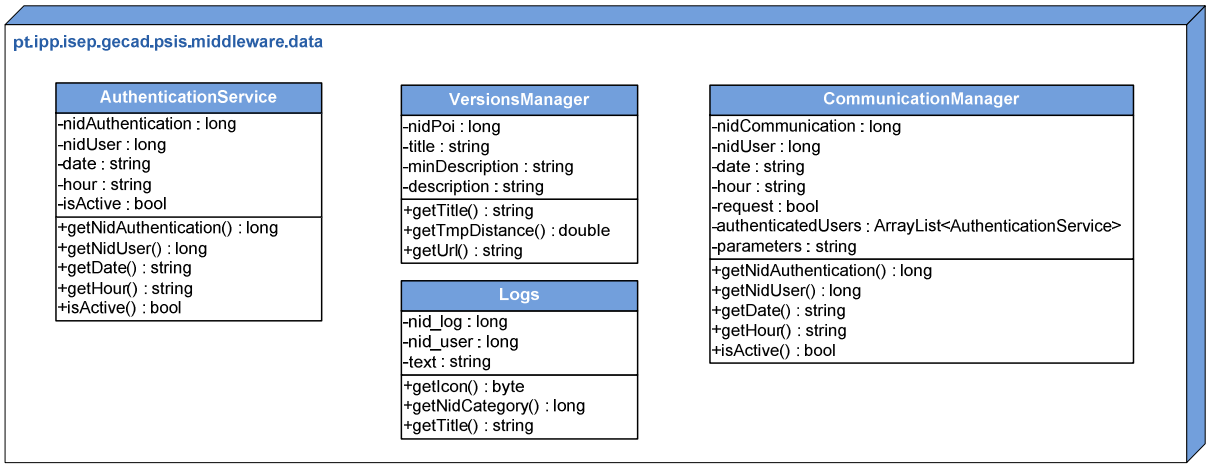

pt.ipp.isep.gecad.psis.mobile.data.database
Database
-instance : Database
-myDatabaseHelper : myDbHelper
-myCalendar : Calendar
-DATABASE_NAME : string
-DATABASE_VERSION : string
+getInstance() : Database
+close() : void
+getAllEntries() : Cursor
+insertEntry() : long
+open() : Database
+removeEntry() : bool
+updateEntry() : int
+getEntry() : Cursor
TableDistances
-CREATE_TABLE : string
-FIELD_DISTANCE : string
-FIELD_NID : string
-FIELD_NID_END_POI : string
-FIELD_NID_START_POI : string
-FIELD_NID_TYPE_TRANSPORT : string
-FIELDS : string
-TABLE_NAME : string
+getDistanceBetweenPois() : long
+getItemFromDatabase() : PoiDistances
TableTrip
-CREATE_TABLE : string
-FIELD_CAR_DISTANCE : string
-FIELD_MONEY_LIMIT : string
-FIELD_NID : string
-FIELD_RATE : string
-FIELD_TITLE : string
-FIELD_WALKING_DISTANCE : string
-FIELDS : string
-TABLE_NAME : string
+getEntry() : Trip
+getItemFromDatabase() : Trip
TableUserInteractions
-CREATE_TABLE : string
-FIELD_COMMENT : string
-FIELD_NID : string
-FIELD_NID_POI : string
-FIELD_NID_USER : string
-FIELD_PROPOSED_KEYWORDS : string
-FIELD_RATE : string
-FIELD_TIMES_CLICKED : string
-FIELDS : string
-TABLE_NAME : string
+maxNidUserInteraction() : long
+poiClicked() : bool
+poiComment() : bool
+poiRate() : bool
TablePois
-CREATE_TABLE : string
-TABLE_NAME : string
-FIELD_ADDRESS : string
-FIELD_AVG_COST : string
-FIELD_AVG_DURATION : string
-FIELD_DESCRIPTION : string
-FIELD_EMAIL : string
-FIELD_FAVORITE : string
-FIELD_LATITUDE : string
-FIELD_LONGITUDE : string
-FIELD_MIN_DESCRIPTION : string
-FIELD_NEAR_POI_BUS : string
-FIELD_NEAR_POI_METRO : string
-FIELD_NEAR_POI_TAXI : string
-FIELD_NID : string
-FIELD_NID_CATEGORY : string
-FIELD_PHONE : string
-FIELD_POI_RATING : string
-FIELD_TAGS : string
-FIELD_TITLE : string
-FIELD_URL : string
-FIELDS : string
+getDayVisits() : ArrayList<PointOfInterest>
+getEntriesFromDatabase() : ArrayList<PointOfInterest>
+getFoodPois() : ArrayList<PointOfInterest>
+getItemFromDatabase() : PointOfInterest
TableTripVisits
-CREATE_TABLE
-FIELD_ARRIVAL_DATE
-FIELD_ARRIVAL_HOUR
-FIELD_DURATION
-FIELD_END_DATE
-FIELD_END_HOUR
-FIELD_EXCLUDED : string
-FIELD_NID : string
-FIELD_NID_POI : string
-FIELD_ORDER : string
-FIELD_ORIGINAL_PLANNED : string
-FIELD_START_DATE : string
-FIELD_START_HOUR : string
-FIELD_VISITED : string
-FIELDS : string
-TABLE_NAME : string
+getAllPoiNotInDayVisits() : ArrayList<PointOfInterest>
+getDayPois() : ArrayList<PointOfInterest>
+getDayVisits() : ArrayList<TripVisits>
+getDayVisitsUnvisited() : ArrayList<PointOfInterest>
+getItemFromDatabase() : TripVisits
+getNewId() : long
+putPoiExcludedInVisit() : bool
TableMultimedia
-CREATE_TABLE : string
-TABLE_NAME : string
-FIELD_DATA : string
-FIELD_NID : string
-FIELD_NID_POI : string
-FIELD_NID_TYPE_MULTIMEDIA : string
-FIELD_RATING : string
-FIELDS : string
+getEntry() : PoiMultimedia
+getItemFromDatabase() : PoiMultimedia
+getNewId() : long
+getPoiEntry() : PoiMultimedia
TableJournalEntry
-CREATE_TABLE : string
-FIELD_COMMENT : string
-FIELD_NID : string
-FIELD_NID_MULTIMEDIA : string
-FIELD_NID_TRIP : string
-FIELDS : string
-TABLE_NAME : string
+getEntry() : JournalEntry
+getItemFromDatabase() : JournalEntry
TablePoiSchedule
-CREATE_TABLE : string
-TABLE_NAME : string
-FIELD_CLOSE_HOUR : string
-FIELD_NID : string
-FIELD_NID_DAY_OF_WEEK : string
-FIELD_NID_POI : string
-FIELD_OPEN_HOUR : string
-FIELDS : string
+getItemFromDatabase() : PoiSchedule
+isOpen() : bool
+getEntry() : PoiSchedule
TableUser
-CREATE_TABLE : string
-FIELD_ACCOUNT_NAME : string
-FIELD_ACCOUNT_PASSWORD : string
-FIELD_IS_LOGGED : string
-FIELD_NID : string
-FIELD_NID_LOCATION : string
-FIELDS : string
-TABLE_NAME : string
+getItemFromDatabase() : User
+getLoggedUser() : long
TableCategory
-CREATE_TABLE : string
-TABLE_NAME : string
-FIELD_ICON : string
-FIELD_NID : string
-FIELD_TITLE : string
-FIELDS : string
+getAllCategoriesIds() : ArrayList<Long>
+getEntry() : Category
+getItemFromDatabase() : Category

Annex B – Middleware Package Diagram



Messages package is equal to the described package on Annex A4.

Annex B1 - Data Package Classes



This package also includes all the classes presented on *pt.ipp.isep.gecad.psis.mobile.data* .

Annex B2 - UI Package Classes