INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Idroid - interest aware augmented reality

Ricardo Brilhante

Dissertation submitted to obtain the Master Degree in
Information Systems and Computer Engineering

Jury

President: Prof. Dr. Luís Eduardo Teixeira Rodrigues
Supervisor: Prof. Dr. Paulo Jorge Pires Ferreira
Co-supervisor: Prof. Dr. Luís Manuel Antunes Veiga
Member: Prof. Dr. Daniel Viegas Gonçalves

November 2012

Acknowledgments

I would like to thank Prof. Dr. Paulo Ferreira and Prof. Dr. Luís Veiga, my thesis advisors, for the total support, motivation and orientation throughout the elaboration of this work. Their orientation and advice contributed greatly for the quality of this thesis.

To all my MSc and BSc colleagues with whom i worked, I want to express my gratitude for all the support, motivation and help provided.

I would like to thank my parents for all the support and help. I owe them my success.

I would like to thank my girlfriend for all the support, motivation and patience during the completion of this thesis.

I would like dedicate this thesis to my late uncle, who passed away during the making of this work, due to pancreatic cancer.

Resumo

As aplicações móveis de realidade aumentada estão a ficar cada vez mais populares. A capacidade de misturar um mundo virtual com o real é uma ideia fascinante, e a possibilidade de obter tal sistema num dispositivo móvel faz com que seja ainda mais interessante, dada a sua ubiquidade inerente.

Assegurar ao utilizador a frescura da informação (de realidade aumentada), no seu dispositivo móvel, é bastante importante. Ao mesmo tempo, o utilizador não deve ser sobrecarregado com informação inútil, e a utilização da rede deve ser mínima, de modo a que a escalabilidade geral do sistema possa ser assegurada.

Neste trabalho apresentamos o Idroid, um sistema de Realidade Aumentada Baseado na Localização que é escalável e que fornece informação que o utilizador necessita, de acordo com os seus requisitos de frescura e os seus interesses (em relação aos pontos de interesse à sua volta). O Idroid faz uso de um modelo de consistência, consciente nos interesses, (Vector-Field Consistency) para limitar a quantidade de informação que é passada ao cliente, enquanto tem em consideração as restrições do mundo real (ex.: ruas, prédios, obstáculos, etc).

Abstract

Mobile augmented reality applications are increasingly more popular. The ability to merge a virtual world with the real one is a fascinating idea, and the possibility of having such a system available in a mobile device makes it even more interesting given its inherent ubiquity.

Ensuring the freshness of (augmented reality) information provided to the user on the mobile device is of utmost importance. At the same time, the user must not be overwhelmed by useless information, and the network usage should be kept to a minimum, so that scalability of the overall system can be ensured.

In this paper we present Idroid, a Location Based Augmented Reality system that is scalable and provides the information the user needs, according to his freshness and interest requirements (regarding the points of interest in his surroundings). It uses an interest-aware consistency model (Vector-Field Consistency) to limit the amount of information flow to the client, while taking into consideration the real world constraints (e.g. streets, buildings, obstacles, etc.).

Palavras Chave
Keywords

Palavras Chave

Serviços Baseados na Localização

Realidade Aumentada

Consciência da Localização

Gestão de Consistência

Gestão de Interesses

Computação Móvel

Keywords

Location Based Services

Augmented Reality

Locality-Awareness

Consistency Management

Interests Management

Mobile Computing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the past years, the interest in mobile devices has grown exponentially. According to Gartner, in the third quarter of the year 2011, the sales of mobile devices totaled 440,5 million, an increase of 5,6 percent compared to the same period last year.[1] One of the main reasons for this growth is the evolution in the software and hardware of these devices. With time, computers tend to increase in power and decrease in size, which makes them more wearable and more pervasive. This allows developers to create mobile applications that provide all kinds of services anytime and anywhere[1]. One of the most popular services that became possible with this development is the Location Based Services (LBS).

A LBS can be defined as a service that takes into account the geographic location of an entity[2]. This type of services allows users to get information of their surroundings and to be more aware of the world. According to Bellavista et al.[3], there are three main reasons that made the use of LBS in mobile computing become more popular. The first one is the appearance of position technology that requires less power from a mobile device. The second is the increasing development of LBS middleware technologies. Finally, the third one is the appearance of 3G mobile networks. Currently, it is fair to say that LBS and mobile computing are a perfect match. A user who wants to know information about a certain place, only needs to reach for his pocket, pick his mobile device and ask for the service.

Recently, another type of technology has emerged in mobile computing: Augmented Reality (AR). This technology is starting to become part of people's reality and mobile devices are one of the reasons for this growth. Mobile AR systems can be described as systems that combine real and virtual objects in real environment, run in real-time and mobile mode, align real and virtual objects with each other and where the virtual augmentation is based on dynamic 3D objects[4]. This means that the effort of creating an application that uses AR can be very high but the result can also be very pleasing to the eye. The biggest advantage of any Augmented Reality system is its user friendly interface. Its simplicity allows an inexperienced user to work with any AR application. Most of these systems only need the camera of the mobile device and the rest works in a non-intrusive way. Also, by combining real world with a virtual one, AR turns computer science more ubiquitous and even more a part of people's life.

With this development, combining these two technologies has become a reality. Nowadays, a user can grab his mobile device, point it to a street or a building and get the information regarding the surroundings through virtual objects. Figure 1.1 shows an example of these systems.

The goal of this work was to design and build an Augmented Reality system called Idroid. This system provides the user information about a certain place using augmented reality. The user, with a Smartphone or a Tablet, uses the camera of the device to show points of interest that are in his range of view. This system has to guarantee the following requirements:

[1] http://www.gartner.com/it/page.jsp?id=1848514

Figure 1.1: An example of Augmented Reality within a Location Based Service.

- Ensure efficiency in terms of network bandwidth;
- Provide high scalability;
- Display only the relevant information to the user;
- Guarantee the freshness of the results.

To fulfill the requirements mentioned above, Idroid has to deal with the following issues: i) reduce the amount of data passed through the network; ii) reduce communication channels where they are not needed; iii) despite challenges i) and ii), still guarantee the freshness of the information; iv) understanding what is relevant information for the user, realizing his needs and desires.

It is important to clarify the meaning of freshness mentioned above. In this case, we guarantee that the information presented in the mobile device is the most recent regarding the information within the database. This means that the results presented in Idroid are consistent with the ones stored in the database. Another type of freshness is regarding the real world and the database. Here, it is required the information of the real world to be the same as in the database. However, this type of freshness is not in scope of this project.

Some solutions, such as Layar[2] and Wikitude[3] already use location based services in an augmented reality system. In these cases, the user subscribes a service in order to get only the information he desires. However, none of these applications, provides a mechanism that guarrantees the freshness of the presented information.

Other solutions, like GeoPointer[5], make an effort to reduce the amount of data passed through the network by getting only the information that the user sees through the mobile device. The disadvantage in this solution is that a slight turn of the camera implies a new request. The increase of requests may lead to high latency, and therefore the freshness of the information cannot be guaranteed.

Solutions like the one presented by Reitmayr and Schmalstieg[6] attempt to efficiently request information from a database. A user normally points his mobile device to a building and then the application returns the information about it. In this kind of solutions the user only has knowledge of the building in front of him and not from the surrounding area. Another disadvantage is that this system normally works offline with static files that work as a database. In a distributed online service these solutions would have problems

[2]http://www.layar.com
[3]http://www.wikitude.com/en

of latency and data management.

Idroid is a system that combines augmented reality with an interest aware location based service, providing only the information relevant to user. To achieve high efficiency and still guarantee freshness, Idorid implements a consistency model that selectively schedules the updates based on their importance. Therefore, the messages passed through the network will decrease and less bandwidth will be required, increasing the scalability of the solution. Through information provided by the user, the system is able to identify the user's interests. Understanding the interests guarantees that Idroid only displays relevant information. The system also acts in a non-intrusive way. After the user select his interests, the application will automatically send new updates.

The remainder of the report is organized as follows. Chapter 2 describes the related work and briefly presents some relevant systems, chapter 3 describes the system architecture. Chapter 4 is focused on the key details of the implementation and chapter 5 analysis the obtained result after a thorough evaluation of the system. Finally section 6 concludes the report and discusses possible improvements that may be performed in the future.

Chapter 2

Related work

When creating a system such as Idroid, it is necessary to discuss some key aspects. Firstly, it is important understand what are the characteristics of a Location Based Service and make an overview over them. This overview is presented in section 2.1. Another issue is the information filtering through interest awareness. Section 2.2 presents this topic and discuss some approaches to solve this challenge. After covering these two aspects, it is necessary to address the challenges of creating an outdoor Augmented Reality system. The discussion of displaying virtual elements in the mobile device's screen is presented in section 2.3. Another goal of Idroid is to guarantee consistency on the displayed results, so, section 2.4 addresses this topic and discuss some algorithms that meet this goal. Finally it is necessary to make an overview of existing systems that work in a similar way of Idroid. In section 2.5 their implementations is presented and discussed.

2.1 Location Based Services

Nowadays, with the rapid development and widespread deployment of information and telecommunication technologies integrated with lightweight mobile devices and terminals, pinpointing location on the move has become a common exercise[7]. The rise of Location Based Services made them a useful tool for mobile computing. In order to create and implement a LBS it is important to fully understand their structure and how they behave. The remainder of this section classifies LBSs in regard of its basic functions.

2.1.1 Self-referencing and Cross-referencing

Location Based Services aim to get information to an end user. The type of information delivered can be regarding the user location or the location of another target. Therefore, there are two types of referencing in LSB: self-referencing and cross-referencing[3, 8].

- **Self-referencing:** A LBS that uses self-referencing is one where the user provides his location in order to get the information he desires. In this case, normally a user wants to know information about his surroundings. Applications like travel services [9] and tourism guiders use this type of referencing.
- **Cross-referencing:** A LBS that uses cross-referencing is one where the user requests information about the location of another target. This kind of referencing is normally used in tracking applications, where the user wants to know the location of a specific entity[10]. Examples of cross-referencing applications are emergency services and automatic vehicle location.

The decision of using self-referencing or cross-referencing depends on the kind of service it is going to be provided by the LBS application. If the objective is to provide information to the user regarding his location then self-referencing is used; if the information to be handed to the user is regarding the location of another target, then the application will use cross-referencing.

2.1.2 Single-target and Multitarget

Regarding the number of targets to be referenced by a LBS application there can be two types: single-target and multitarget[3, 8].

- **Single-target:** In single-target LBSs the major focus is on tracking one target's position. This type of LBS usually displays the targeted entity position on a map or in relation to nearby points of interest. Single-target LBSs are most commonly used in location-tracking services such as safety services and fleet management.
- **Multitarget:** In a multitarget LBS the focus is on interrelating the position of several targets among each other. Nowadays, this type of LBSs can detect the proximity of multiple targets[8]. Information services like tourism guiders and traffic-monitoring are some examples of LBSs that use multitarget.

In order to provide the user with information of his surroundings the use of a multitarget LBS becomes a necessity. Unlike single-target LBS, the multitarget type needs to store information about several different entities at the same time. The use of a database that stores this information becomes crucial. However, there are other ways to access information regarding a user's location, such as through peer-to-peer channels. This will be explained next.

2.1.3 Central and Peer-to-Peer

LBSs need to provide location-aware information; for this purpose there are two ways to retrieve such information: through a central server or via peer-to-peer channels[8].

- **Central:** Central LBSs use an application or a central location server. This server is responsible for providing and managing the information of the system. The central server is also responsible for responding to user's requests providing him with the information he desires.
- **Peer-to-Peer:** Peer-to-peer services[11] are self-organized, which means that position data is exchanged directly between users without the need of intermediary actors. These type of services rely on the information passed through the users. They are the ones responsible of managing the provided information. Peer-to-peer solutions work by providing each user a single piece of information that no other user have. When someone wants information the request is sent to the peer-to-peer network and then another peer will respond.

Although peer-to-peer services represent a more scalable solution, the reliability of the information passed cannot be verified. The advantage of having a central server is that there is an entity that can be responsible for the information. If the data stored on that server is not accurate there is only one entity to blame for. In the peer-to-peer solution that is not possible; there is no way to track which user passed the wrong information. Another disadvantage in peer-to-peer is the availability of the provided information. In p2p networks there can be queries that may not always be resolved. If a user that had a piece of information deletes it or stops sharing it, then it cannot be accessed by anyone in the network.

2.1.4 Outdoor and Indoor

In terms of physical spaces there can be LBSs that provide information for indoor environment and LBSs that work for outdoor environments[8].

- **Outdoor:** LBSs for outdoor environments are available in huge geographical areas. They are normally used for large and unconfined spaces. Outdoor LBSs make use of satellite or cellular positioning technologies in order to be aware of the geographical positioning of the information.
- **Indoor:** Indoor LBSs are confined to small closed spaces. The main functionality of this type of services is to assist the user inside buildings guiding him through halls and rooms[12]. Indoor LBSs rely on local positioning technologies to retrieve location-aware information.

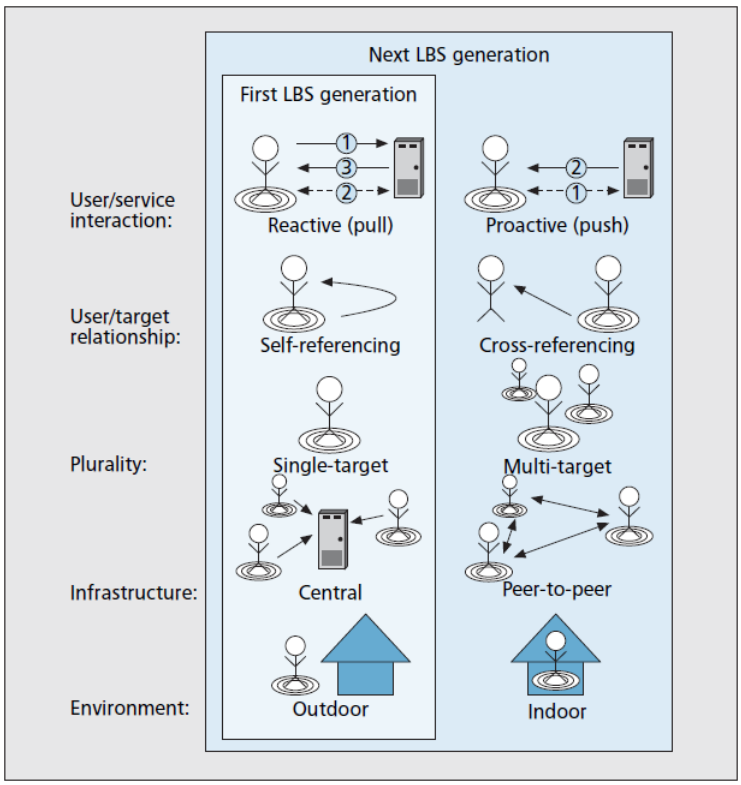

Figure 2.1: Functional classification of LBS.

Nowadays, the development of LBSs is more directed to outdoor environments than indoor. The main reason for this direction is due to the amount of challenges that can be overcome when using a LBS in real world. It is more helpful to get information of the surroundings in a wide open area than in a small confined building.

Push and Pull

It is possible to distinguish two different kinds of location services considering if information is delivered on user interaction or not: pull type services and push type services [13].

- **Pull type:** In the pull type services the user voluntarily requests the information that he desires from the services center[7]. They rely on the traditional request/response paradigm, where a client sends a service request and then the server replies with information regarding the client's location[14]. In this type of services the users "pull" information whenever and wherever they need[15]. Travel directions, mobile yellow pages and buying services are some examples of pull type services.

- **Push type:** In push type services[14] it is the server that automatically sends information to the client. The user does not need to request for information. This type of services uses the notion of user's preferences to know what kind of information is going to be "pushed". Push type services enable the scenario of delivering right information to the correct person on a specific time at a precise location[16]. Some examples of push type services are friend finders, traffic alerts and mobile advertisements.

Most of LBS applications use pull type services. The main advantage of this type is that the user only requests the information he desires. However, there are also some disadvantages, for example, every time the user wants to get information he has to query the system. In other hand, the push type services deal with

this disadvantage by creating profiles of the user, giving him personalized information. Creating this profile is not simple, and guaranteeing that the user gets, exactly, the information he desires is extremely hard to achieve. Some of the most usual errors on push type services is the discarding of important information, and the delivery of non-relevant one. The discussion of creating profiles and making push type applications will be discussed in section 2.2.

2.1.5 LBS types summary

Figure 2.1 represents an overview of the classes of LBSs mentioned above. First-generation LBSs are reactive, self-referencing, single-targeted and they focus on outdoor applications. Most of these services are the so-called finder services that provide to the user points of interest on his surroundings. However, advanced services have begun gaining more impact in the recent years. Examples of this new generation of LBSs are child finder services, that allows the parents to locate their children (reactive, cross-referenced and single-target), and friend finders, where users can mutually share their locations on demand (reactive, cross-referencing and multitarget).

2.2 Interest awareness

When creating a Location Based Service it is important to understand and realize the concept of interest awareness. This concept, also known as context-awareness, commonly refers to information about the user and the surrounding environment[17]. By knowing the user's situation, his preferences and the environment conditions, context-awareness can help improve the user interaction with the system.

2.2.1 User's preferences

User's interest is an important aspect in creating a system that provides only relevant information to the user. To generate interactive and personalized maps according to users' needs is an important component in today's recommendation service systems[18]. It is then important to realize what the user's intentions are and what he is looking for. There are several approaches to get this information and filter it for the user's interest.

In [9] a new model to understand the user's preferences is proposed. This model works on a city traveler help system that allows the users to get information about the surroundings. Users can query the information from the system or can simply receive all information based on their location. Each time, only the information title is passed and if the user wants to see details he has to issue a request by clicking the specified object. The presented model uses this concept to select and filter the information for the user. There are two classes of information that are pushed to the client: one is those the user requested to see the details; another is those discarded by the user.

The objective of this model is to determine where a new piece of information belongs. If it belongs to the first group, the information will be dispatched to the client otherwise it will belong to the second group, and will be filtered out. The authors use the key words from each piece of information for the classification process. For each piece they use a probabilistic analysis to determine which class the information belongs. To do this the server keeps a record of each user's behavior. When new information is retrieved from the database, the server of the system will split it into key words, then it will check the user's history and using a probabilistic method verifies if the information is interesting or not for the user. By doing this, this model tries to probe the user's preferences more subtly and in a smaller granularity making it less intrusive and approximately user preference aware.

Zu [18] proposes a model of personalized information recommendation service system based on GIS (PRSSG) that filters the information sent to the user. The model builds profiles of the users with respect to several personal characteristics, such as age, name, hobbies, occupation, etc. This profile is built in an initial phase by having the user answering several questions; then, an initial user interest model is established and

the system starts filtering information. This filter adapts its algorithm throughout the searches made using the user's feedback about the results. PRSSG provides a personalized system in a location based service that adaptively guarantees the search for information of the user interest.

Aloqa[1] is a proactive application that notifies the user about points of interest, events and friends around him. Aloqa comes pre-configured with a variety of useful content, called channels, which can be added or removed from the user's profile according to his preferences. Each channel has a type associated to it; there are channels for restaurants, concerts, close by friends, etc. When the user changes location, the application refreshes automatically and notifies the user of the available services around him. In this situation the problem of disturbing the user whenever a new service is discovered could arise, yet the Aloqa application provides a set of settings that allow the user to select if he wants to be notified with a ring noise or a buzz, if he just wants to update the background and then he will check it later, or if the user wants to turn the channel off for a period of time. This way, Aloqa pushes information to the user's mobile device proactively and without troubling him with the search parameters.

After studying the above systems, we realized that it is impossible to create an exact user's profile without the need of any interaction. In [9], the system requires the user to remove the information he does not want. Only after this, the system can create filters that discard the non-relevant information. In the second system[18], there is a small set of questions, that need to be answered when the user starts the application for the first time. Although this is not too troublesome, this approach shows that the interaction with the user is a key aspect for the correct behavior of the system. A Aloqa application tries to overcome these advantages by creating a default profile. However, the profile is not accurate for all the users, and needs to be altered. To do that, the user has to interact with the system, giving his own interests and desires.

In the end, it is impossible to create a system completely proactive. There is always some interaction with the user. However, the approach made by Aloqa enables a reasonable balance between push type and pull type services. The most proactive system that can be created is a system that, knowing the interests provided by the user, is able to display and update only the relevant information.

2.3 Mobile Augmented Reality

In order to realize what are the main challenges of creating a mobile AR application it is important to revisit its definition. According to Papagiannakis et al.[4] mobile AR systems can be defined as systems that:

- Combine real and virtual objects in a real environment
- Run in real-time and mobile mode
- Register(aligns) real and virtual objects with each other
- The virtual augmentation is based on dynamic, 3D objects (e.g. interactive, deformable virtual characters

After analyzing the given definition it becomes clear that there are some challenges to overcome in order to create an efficient mobile AR system. The biggest challenge lies in accurate tracking. It is important to determine the user's position and orientation with sufficient accuracy to avoid significant registration errors. The remainder of this section addresses this topic and presents some of the most common techniques.

2.3.1 Tracking and Registration of Virtual Elements

One of the problems when trying to track and register objects in mobile AR is due to the fact that there is no control of the outdoor environment. The fact that it is an unprepared environment makes the

[1] http://www.aloqa.com/

developing of AR systems even more difficult. However, in outdoor Augmented Reality a few lack of precision is acceptable in comparison with indoor. For instance, a doctor performing a needle biopsy requires that the virtual incision marker be accurate within a millimeter, but for a hiker walking around, perhaps even one degree of angular error is acceptable. The subject of tracking and registration for mobile AR has been one of the major research subjects in the area. Here are some of the approaches that have been studied:

- **GPS Tracking:** The Global Positioning System (GPS) is a natural candidate for outdoor tracking[1, 4, 19]. GPS is a time measurement based system that uses the signal of at least 4 satellites to measure the user's 3D position. The accuracy of the localization can vary between 3 and 10 meters, which makes it sufficient for tracking distant objects. However, the differential accuracy makes GPS an inefficient tracking technique regarding nearby objects. Nowadays, GPS receivers are less expensive and are introduced in mass-market devices such as PDAs and mobile phones.

- **Visual Marker-based tracking:** A still common approach for tracking and registration of virtual objects is to make use of easily recognizable landmarks, such as printed markers, in known positions around the environment[1, 4]. When an artificial marker appears on the camera's field of view the application will present the user with the virtual object. This kind of approach requires a previously prepared environment.

- **Visual Markerless tracking:** Another approach is using image processing to provide realistic real-time camera tracking[4, 19]. Nowadays, there are several visual tracking algorithms that recognize the environment in order to track the user's position. However, they require large amount of processing power posing difficulties on the additional AR rendering tasks.

- **Sensor based tracking:** Sensor based tracking is one of the most popular approaches[1, 4, 19]. There are several types of sensor that can detect the position and orientation changes. Accelerometers and pedometers are some examples of sensors that can register position changes with. These two examples provide that kind of information in two different ways; accelerometers provide the distance drift over time periods and pedometers provide the number of steps given in a time period. For orientation changes there are two kinds of sensors: gyroscopes and electronic compass. The gyroscope uses fixed points to register the user rotation and the electronic compass gives this information regarding the magnetic field of the Earth. Sensors can also be used to track objects by setting up transmitters and receivers (using magnetic, optical or ultrasonic technologies). The tracked objects emit a signal and then the sensor matches this signal to the display of a virtual object. This kind of sensors have the disadvantage of requiring extra infrastructures.

- **Wireless-LAN tracking:** Another position tracking system applicable for wide-area mobile AR involves calculating a user's location from signal quality measurement of wireless networking[1, 4]. This approach requires the deployment of equipment in the environment, such as wifi access points. However, wireless networking is the communication of choice for mobile AR systems and so no extra sensor infrastructure is required. The main disadvantage of this wireless tracking is the lack of accuracy. Buildings and other infrastructures can magnify or lower the wireless signal which can give a wrong impression of the user's location.

- **Hybrid tracking:** None of the technologies analyzed before provides a complete solution for outdoor tracking. In the near term, no single technology is foreseen to provide the performance required to efficiently track and register virtual objects. Therefore, combining several tracking technologies, or hybrid tracking[1, 4, 19], is the only feasible approach. Hybrid approaches increase system complexity and cost but provide the most robust results; they allow the weaknesses in each individual technology

to be covered by some other sensor, resulting in an overall system that behaves more robustly than with each sensor applied individually.

Although accurate tracking outdoors is a difficult problem, it is a worthwhile challenge because developing new technologies will lead to improvements for AR and Virtual Environment systems overall. Research in this direction will help push AR systems away from the highly specialized, difficult-to-build prototype systems of today to more flexible, portable systems that can be deployed anywhere.

2.4 Data Consistency

When creating an Augmented Reality system the notion of consistency is very important. According to [20] there are two different types of consistency:

- **Cognitive Consistency:** the system ensures that the user will correctly interpret perceived information and that the perceived information is correct with regards to the internal state of the system. This kind of consistency is most commonly used in the traditional distributed systems.

- **Perceptive Consistency:** the system ensures that the location of a virtual object and a real object that convey data is the same, and ensures that the user is able to sense the different information without losing some of it.

Regarding Location Based Services it is very important to ensure cognitive consistency. In this type of service, the information of the real world is stored in a database, and therefore, attempting to assure cognitive consistency is not only towards to what the user sees but also to what information is registered in the database. A case of inconsistency occurs when, for example, the information of a building retrieved by the LBS and the information of the same building in the database does not match. Nevertheless, it also important to guarantee the tracking and registration of the information, as well as perceptive consistency.

2.4.1 Locality-awareness

Locality-awareness is a form of interest management, detecting users' interest based on their locality. It is commonly used in massive multiplayer games, where consistency is most desired around the player position. Middleware such as Matrix(Adaptive Middleware for Distributed Multiplayer Games)[21], VFC for Ad-hoc Gaming[22] and Unifying Divergence Bounding and Locality Awareness in Replicated Systems with VFC[23] can track players position and according to it, they can strengthen consistency guarantees around the player position and weaken it as the distance increases.

Location Based Services base their functionality on the location of the user, so the notion of locality-awareness is thereby implicit. When a user wants to know information about his surroundings, normally he is most interested in the places that are close to him than the ones that are far. Guaranteeing consistency nearby the user is then more necessary in order to provide an efficient and well adjusted service.

2.4.2 Vector-Field Consistency

Vector-Field Consistency[22, 23] is an efficient and adaptable consistency model that uses locality-awareness techniques to selectively and dynamically strengthen or weaken the consistency of replicas. The idea behind VFC is that objects that are far from the user's range of view need to be less consistent then closer objects. So, it uses the locality-awareness techniques to identify different zones, and each zone has its own degree of consistency.

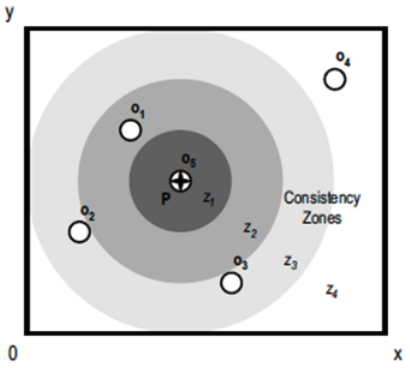

Figure 2.2: Consistency zones centered on a pivot.

To identify the different zones, VFC uses the concept of *pivots* and they represent the user's current position. Figure 2.2 illustrates an example with four consistency zones around the *pivot*(P) that are limited by a concentric circle. The VFC algorithm enforces the consistency within Z_1, and then this consistency level will decrease in throughout in the other three zones, from inside out.

To ensure different degrees of consistency, VFC provides 3-dimensional vectors called *consistency vectors*, $\kappa = [\theta, \sigma, \nu]$. Each dimension vector is a numerical scalar defining the maximum divergence of the orthogonal constraints time (θ), sequence (σ), and value (ν), respectively.

- **Time:** Specifies the maximum time that a replica can be without being refreshed with its latest value. Consider that $\theta(o)$ provides the time passed from the last update, relatively to object o. The time constraint κ_θ enforces that, at any time, $\theta(o) < \kappa_\theta$. This scalar (not necessarily integer) quantity measures time in seconds.

- **Sequence:** Specifies the maximum number of lost updates. The sequence constraint κ_σ enforces that, at any time, $\sigma(o) < \kappa_\sigma$. The unit is the number of lost updates.

- **Value:** Specifies the maximum relative difference between the current content and other updates. Consider that $\nu(o)$ provides this difference. The value constraint κ_ν enforces that, at any time, $\nu(o) < \kappa_\nu$. The unit of variation is a percentage.

For example, consider a consistency vector $\kappa = [0.1, 6, 20]$, that describes the divergence bounds of each constraint (time, sequence, value). This means that the view can only be outdated for 0.1 seconds or 6 lost updates or with a 20% variation in the view content.

Through a selective form of increasing and decreasing consistency enforcement, VFC is able to ensure critical updates to be immediately sent and less critical to be postponed. Thereby, it makes an efficient resource usage, reducing the network bandwidth usage and masquerading latency.

2.5 Location Based Services with Augmented Reality

Embedding Location Based Services in Augmented Reality systems is becoming a growing research issue in the past years. However, these kinds of systems not only have to address to challenges behind any LBS, but also the issues regarding the implementation of AR environments. In the remainder of this section we present some attempts made throughout the years to accomplish an efficient LBS that uses AR.

Figure 2.3: Head Mounted Display used in the Touring Machine

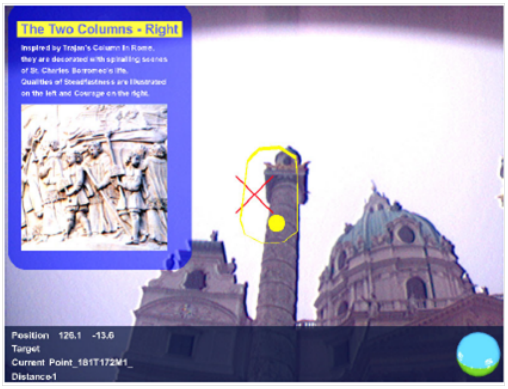

Figure 2.4: Example of the system proposed by Reitmayr et al.

2.5.1 Head Mounted Display systems

In the early years of mobile AR, the use of head mounted display (HMD) to generate virtual objects was very common. Feiner et al.[24] were the first developers to produce a touring machine that assists the user who is interested in a university campus.

As a user looks around the campus, his see-through HMD (see Figure 2.3) overlays textual labels on the campus. The system can display information about the campus, the user's current location, a list of departments and a list of buildings. After selecting a building within the list, the application provides a small compass pointer that will guide the user to the building location. Another feature presented by this system is the ability to display more specific information about the buildings. To do that, the user has to select the building and then an informative page will be displayed.

This was the first system that combined LBS with AR, and although it uses a small database and a HMD, it was a good starting point for more complex applications.

Reitmayr et al.[6, 25] created a tourist guide application that helps the user to travel through the city of Vienna. In comparison to the Touring Machine of Feiner, this system uses a larger database, presents more complex functionalities and supports collaborative work. This application provides three types of functionalities: navigation, information browsing and annotations. In navigation mode the user selects a specific target address or a desired target location of a certain type and then the system computes the shortest path in a known network of possible routes. The information of the route is displayed as a series of waypoints that are visualized as virtual cylinders standing in the environment. If two or more users are present, a number of collaborative interactions are possible. The user can follow or serve as a guide to another user or the users can meet halfway from each other. The information browsing mode presents the user with location based information. This information is displayed through virtual icons that can be selected to present more specific information about the building, as shown on Figure 2.4. In annotation mode the user can place virtual icons in the buildings that can be shared with different users. In this mode the user can also attach information regarding the placed icons.

The use of a small database allows this system to offer the user useful functionalities for city traveling.However, this system uses a small, static database and the issue of the freshness of the information is not

Figure 2.5: MapLens using the mobile phone and the paper map

Figure 2.6: The application proposed by Takacs et al. displaying information about a building

address. With a large, modifiable database, this system does not guarantee the consistency of the results. Another disadvantage of these two applications is the fact that they use a HMD. Users normally do no walk around carrying such devices.

2.5.2 Mobile Phone applications

The implementation of LBSs using AR in mobile phones enables the user to walk with such applications in their pockets. Some systems, such as MapLens[26] were created in order to fulfill this necessity.

MapLens[26] was created by a group of researchers from the Nokia Research Center. This application uses paper maps that are captured with the mobile phone camera to present virtual data information. The system analyses and identifies the GPS coordinates of the map area visible on the phone screen and based on these coordinates, location based media (photos and their metadata) is fetched. Users access media by selecting and clicking an icon, which displays a thumbnail photo on top of the map image on the display. MapLens uses image processing algorithms to compare the visible map area with predetermined map files. It uses this map files to determine the GPS coordinates and then retrieve the media icons related to that location.

The use of such system enables the awareness of points of interest in a wide area; however, this solution requires the use of a paper map. This can become troublesome for the user and can bring several disadvantages. One of them is that the tracking algorithm requires stability and therefore the user has to stand still while managing the mobile device. Another disadvantage is the need of holding the map as background surface. This means that the user has to perform two tasks: holding the map with one hand and pointing the mobile phone with another. Finally, the operation is constrained within proximity range of the paper map, which means that the mobile device and the map have to be placed close by.

Takacs et al.[27] created a system that rule out this disadvantages by discarding paper map and only using the mobile phone. The proposed system captures the image retrieved by the mobile phone screen and then matches it to a set of images stored in a database. After this, the information related to image is then passed to the client and a virtual tag is displayed on the building. The images are searched using an algorithm that only queries nearby images. Along with this, the image recognition algorithm only uses part a small part of the image to make the match. These two feature guarantee the efficiency of the system.

By using image recognition, this solution guarantees that the information display is accurate. However, a system like this, only gets information of one building at a time. If the user wants to understand his

Figure 2.7: The Mara AR application.

surrounding he has to query for all the buildings. Also, the information passed to the client is rather small, only passing a text tag is not really giving the user much information. Therefore the freshness issue is not a real concern.

MARA[28] is a system that provides points of interest in the range of view of the user without the use of any kind of external feature. It was developed by the Nokia Research Center and it uses the sensors of the mobile phone to get the current position and heading of the user. The user points the device's camera and then if there are any available annotation the information about the surrounding objects will be displayed. It is also possible to "click" on a selected object and be redirected to the hyperlink that it is associated.

One disadvantage of this system is that it does not filter any information passed through the network. All the annotations that are available for the image in the device's screen will be presented, possibly cluttering the display with non-relevant information for the user. Not only that, the amount of data passed through the network can be very high, depending on the number of annotations available on the database.

2.5.3 New generation of Augmented Reality Browsers

The appearance of operating systems such as Android and Windows Mobile made the implementation of application for mobile devices more portable. Since then a new set of LBS applications that use AR were created.

One example of such systems is the Layar Reality Browser[2]. It is a mobile augmented reality application that allows the user to discover new information by looking around the world. With the aid of cameras, GPS, compasses and accelerometers, Layar provides digital information superimposed onto reality.

The concept behind Layar is the use of layers that anyone can access or create. All the information about the world is stored on these layers. They can provide information of the whereabouts of restaurants, coffees, subway stations, etc (Figure 2.8). For a user to gain access to these layers, he has to subscribe to the one that has the information he desires. The user can also define the radius of his search and therefore limit or wide the search results.

The biggest disadvantage of Layar is the fact that it is impossible to overlap layers. A user is restricted

[2]http://www.layar.com

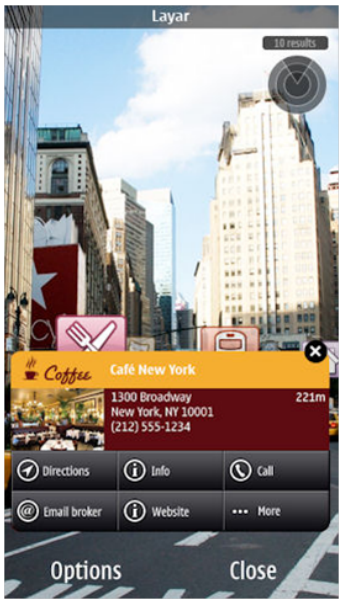

Figure 2.8: Layar Reality Browser

Figure 2.9: The "worlds" of Wikitude

to the subscribed layer and if he wants to subscribe another he has to unsubscribe the previous one. Other disadvantage is the difficulty of creating layers. They are basically a PHP server that supports JSON with a MySQL database[29]. For the average user this is not trivial and restricts the creation of layers to experienced developers. Finally, Layar does not guarantee the freshness of the displayed results. Nothing assures that, if the layer is altered, the information will be passed to the user, guaranteeing the freshness of the results.

In 2008 an application similar to Layar was created: Wikitude[3]. It was developed by Wikitude GmbH (formerly Mobilizy GmbH) and is a mobile augmented reality system that scans the user's surrounding for geo-references using the camera and the device's sensors. In Wikitude each point of interest belongs to a provider, a so called "world", see Figure 2.9. The functionality of this "world" is similar to the layers in Layar, but it has a difference: they can aggregate different types of places and not just only one. Another difference is the possibility to access different "worlds" at the same time and get information about all of them.

One disadvantage of Wikitude is the number of points of interest displayed when there is no customization of the preferences. Wikitude gathers all the nearby information of all the "worlds" and presents the results to the user, which can make the display cluttered by non-relevant information. This also means that a large amount of data is passed to the network and with high latency it can make the performance of the application decrease.

The GeoPointer[5] solution tries to decrease the information passed to the user. It was developed by Wolfgang Beer as client-server application with a lightweight client and server component. The client is only responsible for getting the coordinates of the user (latitude and longitude), as well as the orientation of the device. In the end the client will also be responsible for representing the information on the mobile device screen. After receiving the direction and position of the user, the server defines his area of interest. First, it will calculate a point, called waypoint, which represents the maximum coordinate to present results. This waypoint is calculated by using basic trigonometry relating the current user's position, his direction and a maximum distance. Then a rectangular area is calculated by using the current position, the distant waypoint, the direction of the user and a threshold that represents the size of the rectangle. In

[3]http://www.wikitude.com/en

Figure 2.10: The rectangle where GeoPointer is going to display the results

Figure 2.11: MobiAR displayng information about a monument

the end the user will have results from this rectangle with the waypoint in its center, as shown on Figure 2.10.

GeoPointer seems to be a really good way to present only the information that the user is interested, but it has a disadvantage. By changing the direction of the mobile device a new request has to be made and therefore the server has to perform new calculations. A user that wants to know about the information of a city will turn his device to every direction, this means that requests are made very often and if there is high latency, the results can be inconsistent with the image displayed in the screen, decreasing the performance of the application.

MobiAR[30] is an Android service platform for tourist information based on AR, which allows users to browse information and multimedia content about a city through their own mobile devices. This application not only handles location-based information and user preferences, but also uses image recognition to filter the resource the user is interested in.

MobiAR is a client-server application where the client sends the GPS coordinates of the user, the orientation of the mobile device and a reference to the image displayed on the screen. When the server receives the request it will match the image to a set of referenced images stored in a database. To narrow the search the server will only try to match the images that are close to the GPS coordinate received previously.

Using image recognition makes MobiAR a very precise augmented reality browser, however, image recognition is not trivial and may be computationally heavy. This can lead low performance and make the user wait for an answer. And, like the system proposed by Takacs et al.[27], it only gets results one building at a time. Regarding the freshness of the information, there are no guarantees that, if the information is altered, it will be passed to the client.

2.6 Summary

Table 2.1 presents a summary of the above mentioned systems regarding the type of LBS, the AR tracking technique and the consistency algorithm used. It is important to note that every system is a self-referencing, multitarget, central and outdoor kind of LBS, therefore the difference between them will be if they are pull

System	LBS type	Tracking Technique	Freshness guarantees
Touring Machine[24]	Push	Hybrid: GPS and Sensor tracking	None
Reitmayr et al.[6, 25]	Push	Hybrid: GPS and Sensor tracking	None
MapLens[26]	Pull	Hybrid: GPS and Visual Markless tracking	None
Takacs et al.[27]	Push	Hybrid: GPS and Visual Markless tracking	None
MARA[28]	Push	Hybrid: GPS and Sensor tracking	None
Layar	Pull	Hybrid: GPS and Sensor tracking	None
Wikitude	Pull	Hybrid: GPS and Sensor tracking	None
GeoPointer[5]	Push	Hybrid: GPS and Sensor tracking	None
MobiAR[30]	Push	Hybrid: GPS and Visual Markless tracking	None

Table 2.1: Comparison between the studied systems.

or push type LBSs.

None of the systems have any mechanisms to adapt consistency guarantees according to the needs/resources. Typically they do not address this issue. In most of the Augmented Reality applications, the freshness of the information is assured by a total consistency model. In cases like this, any update made will be propagated to the user.

Further, most systems are push type Location Based Services. The advantages of these kind of systems is that they do not trouble the user. However, most of them only give information of a point of interest at a time. We considered these kind of system a push type because the user does not need to select his interests, he only has to point the camera. The push systems that provide information about the street, do not take in consideration the interests of the user, and they possibly clutter the screen with non relevant information. The advantage of the pull type is that they present only the information the user desires.

Regarding tracking techniques all the systems make use of GPS tracking to get the location of the user. Most of the systems use the sensors of the device to get the orientation of the user and understand what he is seeing. Nevertheless some systems use visual markless tracking to infer the objects projected in the screen.

Chapter 3

Architecture

This section presents the architecture of the proposed system, named Idroid. This system provides to users the search for points of interest in their surrounding area. To do this, a user only needs to pick up his mobile device, define the search criteria and point the camera to any target. The results will then be efficiently presented using virtual elements drawn on the mobile device's screen. Additionally, Idroid uses a relaxed consistency model that takes into account the interests of users over certain points of interest, in order to create multiple consistency levels and assure the freshness of the information presented to the user.

3.1 Baseline Architecture

Idroid is based on a client-server architecture. Figure 3.1 illustrates the process where clients submit their location and interests to the server (for simplicity of the presentation, we considered only one server, capable of responding to all the users). These interests represent the search criteria for points of interest, such as the places the user wants to find, the maximum distance he wants to search. Taking these interests in account, the server is then able to give the user the information he desires, enforce multiple consistency degrees over multiple points of interest, in order to maintain this information the most up to date.

Thus, the process is based on the exchange of updates between clients and server, and on the submission of client's interests to the server. These interests are then used to make decisions of which updates have to be immediately propagated and which can be stored for a while and only later be sent.

Users may manifest their interests through an interface. In this interface users may select their search criteria to find different points of interest. They can be divided in four different steps. Firstly, the user selects what he is interested in. He can choose from a set of predefined categories that are divided in seven different main groups: *Food and Drink*, *Health and Welfare*, *Transportation*, *Religion*, *Entertainment*, *Local Services* and *Other*(covering non categorized interests such as *atm*, *laundry*, *gas station*, etc.). The second step is to define his traveling mode. A user can be walking or driving his car. Thirdly, a user must define if he wants the distance to the points of interest in minutes or in meters. Finally, the user has to specify the maximum radius of search.

The server has the responsibility of identifying what are the points of interest in the user's surroundings, and guarantee that the user has his information the most fresh as possible. The points of interest presented by Idroid are stored within a databases at the server side, and the information associated to them can be altered by anyone who has access to it. For example, the rating of a restaurant can be modified by anyone who decides to rate it, or, the office hours of the restaurant can be changed by its owners, etc. Every time a change occurs, the system verifies the consistency of the points of interest. To do this, Idroid uses a consistency model, named VFCdroid, that is based on the Vector-Field Consistency(VFC) model, providing fresh

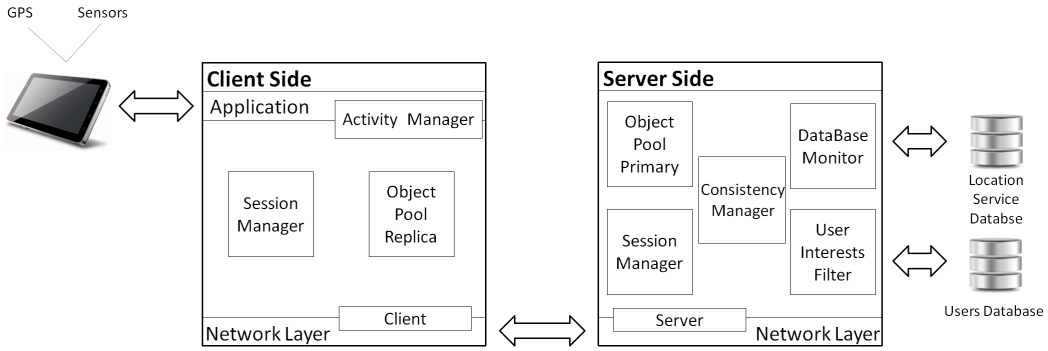

Figure 3.1: Idroid global architecture.

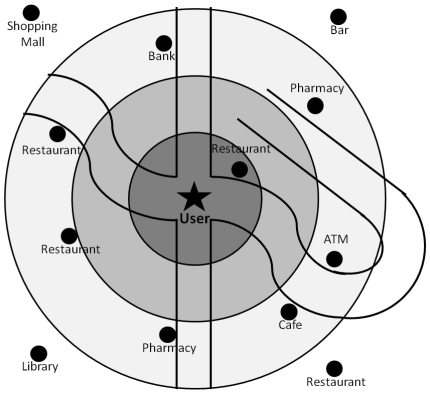

Figure 3.2: The VFC model applied over the digital map.

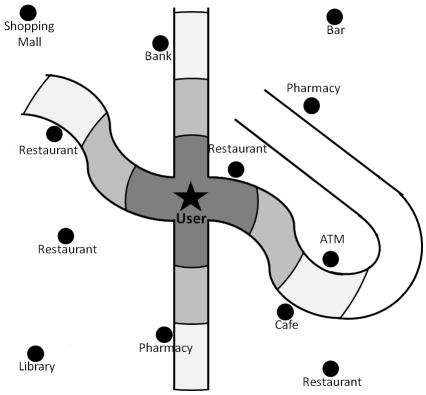

Figure 3.3: The VFCdroid consistency tubes applied over a digital map.

results using the least network resources possible. Idroid also takes into account the interests of the user, updating more rapidly the information closer to him and discarding non-relevant information. Whenever an object needs to be refreshed a Database Update is sent.

The main idea for the VFCdroid model is the creation of consistency tubes that provide the same consistency guarantees as the VFC consistency zones. Figure 3.2 shows how the VFC model can be applied to several points of interest. In Idroid, these zones are replaced by tubes, in order to adapt the VFC model to the real world constraints, see Figure 3.3. The detailed explanation of this concept is presented in the the next Section.

The users profiles are also stored within a database. Idroid stores only critical information about the user, like his interests and his location. The information stored in the database is changed every time a user alters his search parameters or his location. When any of this changes occurs a Client Update is sent. This updates are divided in two groups: Interests Updates and GPS Updates. The Interests Updates occur when one of the search criteria changes. For example, if the travel mode or the interests of the the user changes, an Interests Update is sent. The GPS Updates are sent when the the user is moving with the device and changing the place he is in.

With this architecture, Idroid gives the user the relevant points of interest on his surroundings, using the least network bandwidth available. The system also provides the most fresh information to the user, regarding his interests and his location, updating the points of interest that are closer to him, and postponing the updates of the ones that are further. Although Idroid is reducing the consistency in the users, this does not really represent an inferior usability feature. Since the user pays more attention to the points of interest that are closer to him, the information of the further ones is less critical and can be updated later.

3.2 VFCdroid Consistency Model

In this section we discuss the consistency model implemented in Idroid. First, we describe the basics of this model and understand its behavior, and then we explain how it works in Idroid. Further we present how the VFCdroid is able to remove non-relevant information for the user.

Idroid is an application that is able to present the users, only the relevant points of interest in their surroundings and maintains the information of those points the most fresh as possible. We do this by using the least network resources, improving the system scalability. In order to do this, we had to think of a consistency model that was capable of providing such features.

In order to understand the user's interest, our solution uses a push propagation model in the client side. This delegates to the clients the duty of providing the server with the interest of each user.

Regarding the propagation of information about the points of interest, Idroid uses a pull propagation model. This way the client does not have to worry about new updates, they are pulled by the server whenever it is necessary. This model avoids the overhead of having peaks of clients requests, saving network bandwidth.

The points of interest that the user wishes to see are scattered around him. So, VFCdroid only has to be concerned with such points of interest. The ones that are far, are not in the interest of the user. Therefore they can be ignored.

This led us to the topic of locality-awareness (or interest-awareness), where the knowledge of user's interests can be taken into account to specify multiple consistency guarantees. As such, we use as consistency model that takes into account the interest of users over the points of interest to enforce different consistency guarantees.

We identified the VFC model as a natural fitting model to this environment, where its locality-awareness techniques can be applied from the most to the less important user's data. By this, VFC can impose strong consistency guarantees to points of interest that are of extreme interest to users, or relax some consistency guarantees to the points of interest of less interest to users.

In the end, to accomplish multiple consistency requirements, we base our consistency model, which we named VFCdroid, on the Vector-Field Consistency(VFC) model. Therefore, VFCdroid is based on three main concepts: pivots, consistency zones and consistency degrees. Pivots represent the location of the users of the application. A pivot serves to calculate the distance between the location of the user and the points of interest.

Consistency zones are formed around pivots. In Idroid there are three different zones. Each zone has its own limits, a maximum range. They have variable sizes and each has it's own area. The first zone represents the points of interest that are closer to the user. This zone area begins in the user position and ends in Sits limit. The second zone starts its area where the first zone finished and ends in its limited range. Finally, the third zone begins it's area where the second zone ended and goes as far as the maximum distance of search,

defined by the user. A point of interest is in a particular zone if the distance between the user and the point of interest is within the zone limits.

Each consistency zone has a consistency degree associated. Consistency degrees vary with the distance, so that zones closer to the pivot have higher consistency degrees. The degree of each zone is defined by a 3-dimensional vector that specify the consistency deviation limits for objects within a zone. The vector parameters are: time(θ), sequence(σ) and value(ν). Time specifies the maximum time an object can be without being refreshed with the most recent value. Sequence specifies the maximum updates an object can receive without being refreshed. Value specifies the maximum difference an object can be from the last refresh. The values of these parameters can also be customized by the user.

In order to adapt these concepts to our system we take in consideration three different aspects:

i) user's interests

ii) distance of search

iii) real world

The first aspect is related to what the user wants to search, whether it's pharmacies, restaurants or museums. It is important to understand what are the points of interest the user is looking for.

In Figure 3.4 we present a user, in the center of the street, with several points of interest in his surrounding. For explanation purposes, we assume that the user has two different interests: restaurants and pharmacies. Therefore, all the other points of interest that are not part of these categories must be excluded from the results. After this removal the user will only get the points of interest he is interested in, as shown on Figure 3.5.

The second aspect is related to the range of search. Users do not want all the points of interest in the world, so, he defines an area of search that will be applied over the digital map. This maximum distance can be given in minutes or in meters, it is up to the user to decide which one he wants.

Figure 3.6 shows the area, selected by the user, being applied on the previous points of interest. As we can see there is a restaurant that is out of the range. This means that it is not of the user desire that this restaurant appears on the results. After this step, we filtered even more the points of interest of this user, as shown on Figure 3.7.

Finally, the third aspect is related to the real world constraints. To understand more accurately the points of interest in the user's reach, it is important to take in consideration the obstacles of the real world. For example if there is no route possible to reach a certain point of interest, there is no interest on showing that result to the user. Therefore, we modified the normal search area into tubes, in order to cover the streets surrounding the user. This way, the inaccessible buildings and the ones that are over the radius of the user are not taken in consideration. Then we divided these tubes in consistency zones. The size of each zone is defined by the distance a user can travel through a street. For example, if one zone is limited to 100 meters, it is limited, to every possible route, by the same distance, irregardless of the amount of curves the street has (see Figure 3.8).

Analyzing Figure 3.8 we can see that there is a restaurant that is not accessible via the presented streets. This object is not in the interest of the user because he cannot reach it. It can be considered as non-relevant information. We can also see that there is a pharmacy that, through moving around the street, is reachable by the user. However, according to the definition of the consistency tubes given before, this object is too distant of the user. Therefore, the user cannot reach it and it must not be presented to the user as a point of interest. Figure 3.9 presents the final result for the search of points of interest for this specific user.

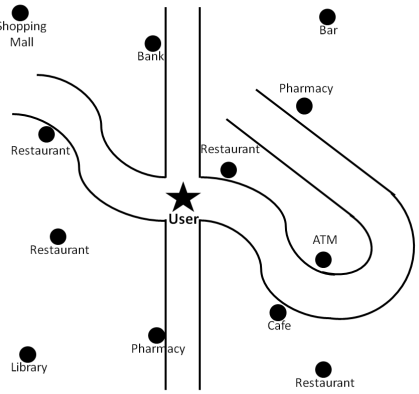

Figure 3.4: A user in the middle of the street with several points of interest around him.

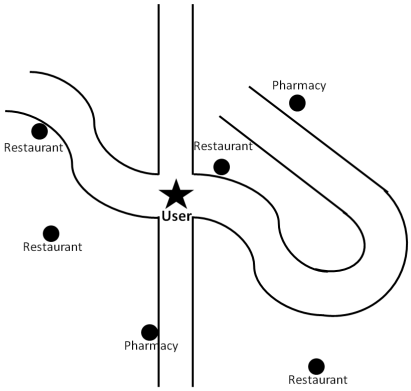

Figure 3.5: The points of interest that are part of the interests selected by the user.

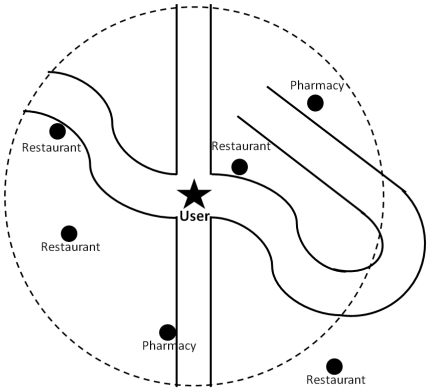

Figure 3.6: The area of search applied to the previous points of interest.

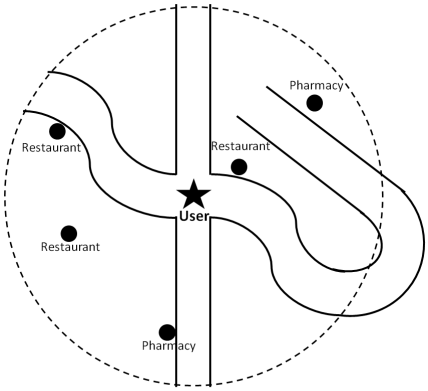

Figure 3.7: The points of interest that are in the interests and in the range of the user.

The decision of creating these tubes of consistency allows us to give the user the accurate distance to a certain object. More importantly, it also eliminates all non-relevant information, providing the user only with the points of interest he desires and the ones he can reach.

In the end, by using VFCdroid, we intend to reduce the bandwidth usage. We intend to do this not only by postponing non critical updates, but also by reducing the amount of points of interest to the necessary, removing the non-relevant ones.

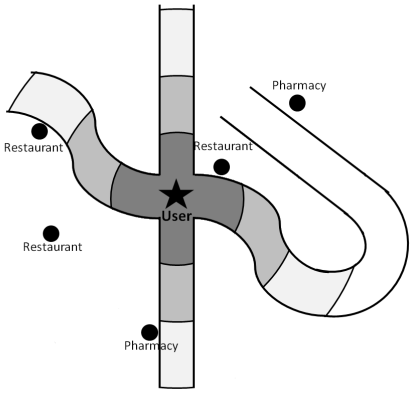

Figure 3.8: Idroid consistency tubes applied to the streets surrounding the user.

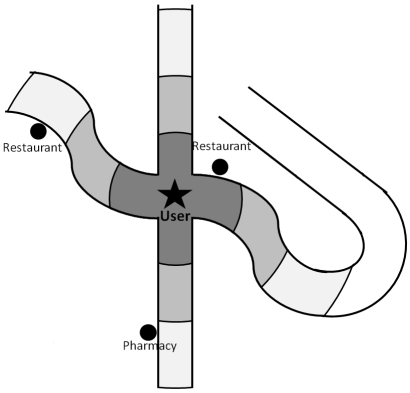

Figure 3.9: Final version of the points of interest presented to the user.

3.3 Idroid Main Components

In this section we describe an overview of Idroid main architecture, namely the client and the server nodes.

Client nodes correspond to a user-level application and environment that allows users to give their interests to the system. Idroid client is responsible for getting the inputs of the user, meaning his interests and search criteria, and the inputs received from the device's sensors, as the orientation and coordinates of the user. Client nodes also have the responsibility of sending this information to the server. Finally, after receiving the points of interest from the server, the client node is responsible for drawing the objects, that are on the device's screen range, in the right position.

The Server node corresponds to the central node of the system. All updates are either received or sent by the server, being this node responsible to ensure data consistency through client nodes. This component is thus responsible for receiving updates and propagate them according to users interests.

3.3.1 Client Side

The client has the responsibility of getting the user's interests and position and to draw the points of interest in his surroundings. The client node architecture, represented on Figure 3.10, is composed by four main modules: *Activity Manager*, *Session Manager*, *Object Pool Replica* and *Client*. On the following, we describe each module in more detail:

- **Activity Manager:** responsible for both inputs and outputs of the application. The two functions work at the same time. While listening to the inputs of the device, the Activity Manager is also drawing objects in the screen. The Activity Manager listens to the sensors of the mobile device to get the location of the user. This information, plus the search criteria, form the possible inputs of the application. The outputs are the objects to be drawn on screen. Every time the Activity Manager needs to draw something, it checks the heading of the device using its sensors. Then, it fetches the objects stored in the Object Pool Replica and sees if they are in the device's angle of view.
- **Session Manager:** is responsible for processing the messages sent between client and server. Every time an update from the server is received, the Session Manager is responsible for processing this update and manage the information stored in the Object Pool Replica. It is responsible for adding new points of interest or delete the ones that are not in the user's surroundings.

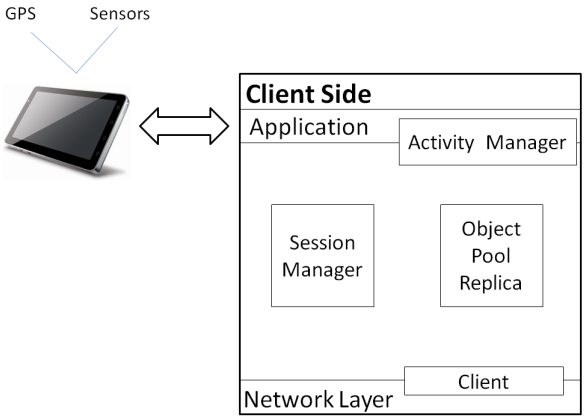

Figure 3.10: Client node architecture

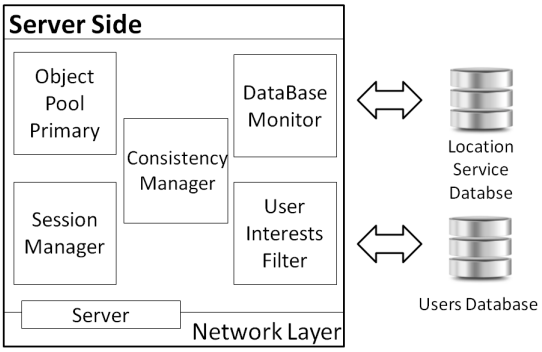

Figure 3.11: Server node architecture

- **Object Pool Replica:** stores all the objects that are in the user surroundings. These objects are available to be drawn by the Activity Manager, depending on the the user's heading.
- **Client:** is responsible to maintain a connection between the server and the client. It is responsible to send the information regarding the user and to receive the updates sent by the server. This information will be processed by the Session Manager.

3.3.2 Server Side

The server has the responsibility of understanding what are the points of interest on each user's reach and to maintain their information fresh. The Server node architecture, as seen on Figure 3.11 is mainly composed by six modules: *Session Manager*, *Consistency Manager*, *Object Pool Primary*, *Database Monitor*, *User Interests Filter* and *Server*. On the following, we describe each module in more detail:

- **Session Manager:** processes the messages sent from the client and the messages to be sent to the user. It is responsible to process the messages received by the clients and dispatch them to the Consistency Manager. It is also responsible for managing the massages sent to each client.

- **Consistency Manager:** this model is responsible for enforcing the VFCdroid consistency model. It ensures critical updates to be immediately propagated to clients and less critical to be postponed.
- **Object Pool Primary:** stores all the points of interest that are part of the users results. It manages the objects, removing the ones that are no longer associated to any user.
- **Database Monitor:** this model has the responsibility of checking the new versions of the points of interest stored in the Object Pool Primary. The Database Monitor periodically monitors the Location Service Database, in order to infer if there is any alterations. If any occurs, it stores the new data in the Object Pool Primary and informs the Consistency Manager about the alterations.
- **User Interests Filter:** is responsible to store the information about the users that are requesting services from Idroid. It reads the information from the Users Database in order to understand their interests. The User Interests Filter is also responsible for managing the users, removing the ones that are not using the application.
- **Server:** responsible to send and receive information from and to the client. The Server is capable of responding to several clients at the same time. By using a pool of threads, the server is able to assign each request to a new thread.

3.4 Updates propagation

In this section we describe how the the updates are propagated from the client to the server and from the server to the client.

As said before, there are two types of updates: Database Updates and the Client Updates. The Database Updates occur in the server side, and are triggered by alterations on the information stored on the Location Service Database. The Client Updates are, as the name suggests, sent by the client and are triggered by a change of interests or a change of location.

3.4.1 Database Updates

The Database Updates are a very important aspect in the correct function of the system. These updates guarantee that the objects presented on the users are the most fresh. These updates are initiated on the server side and are triggered every time an alteration happens in Location Service Database. The propagation of each update is controlled by the VFCdroid model, which determines if it can be delayed or if it has to be immediately propagated to a certain client. The VFCdroid model is able to perform this selective selection and propagation of updates according to each user's interests. The process of updating the information of the users starts once the information on a user becomes outdated.

Figure 3.12 illustrates the process of updating a point of interest in the client. The process starts on the server side and ends on the client, when the object is updated with its most fresh information.

The Database Updates process is composed by nine main steps, namely 1) *Checking updates*, 2) *Informing the Consistency Manager*, 3) *Verifying consistency*, 4) *Creating responses*, 5) *Sending updates*, 6) *Processing updates*, 7) *Updating information*, 8) *Informing Activity Manager* and 9) *Drawing objects*. We now describe each step and each involved module of the process:

- **1) Checking updates**
  Periodically, the Database Monitor checks the Location Service Database for new updates. For every object present on the Object Pool Primary the Database Monitor compares the information with the information stored on the Location Service Database. If the information differs, then the object is marked as outdated.

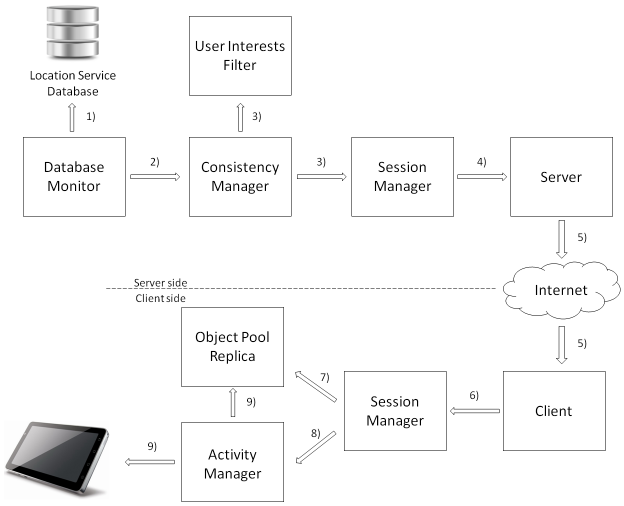

Figure 3.12: Database update process. The figure shows how the changes in the objects are propagated to the client.

- **2) Informing the Consistency Manager**
  When the Database Monitor checks all the objects, it alters the ones marked as outdated, replacing the information stored on the Object Pool Primary. Then, the Database Monitor informs the Consistency Manager of the objects that were modified.

- **3) Verifying consistency**
  After receiving the information from the Database Monitor, the Consistency Manager, will then verify the consistency of the modified objects. For each object, the Consistency Manager will check which users have a replica of it. Then, for each user, it will verify the consistency limits and understand who are the users that consider the object outdated. The process of verifying consistency and enforcing the VFCdroid will be described in more detail in the Section 3.6.

- **4) Creating responses**
  After verifying the consistency of each object for each user, the Consistency Manager gathers the users and the objects that need to be refreshed. Then, it creates response messages containing only the updated objects for each user. This information is passed to the Session Manager, in order to be dispatch to every user.

- **5) Sending updates**
  Then, the Server sends the objects to be updated to the specified users. This information is passed through network, directly to the Client module of the user.

- **6) Processing updates**
  When the Client receives a new message, it informs the Session Manager that new information has been received. The Session Manager will the process this information.

- **7) Updating information**
  The updates received are, then stored in the Object Pool Replica. The new information replaces the one that was stored previously.

- **8) Informing Activity Manager**
  After this, the Session Manager informs the Activity Manager that there was an update made in the Object Pool Replica. The information displayed on screen becomes invalid and needs to be refreshed.

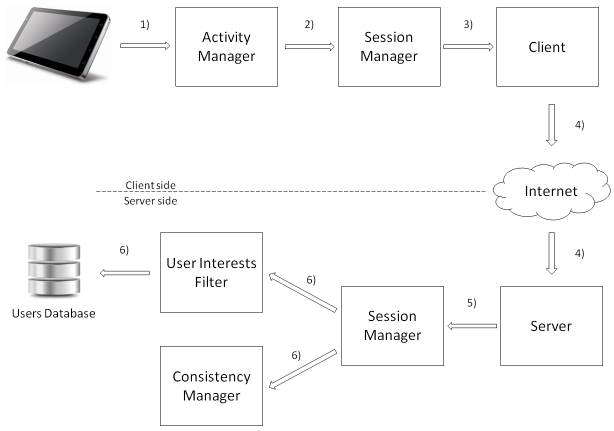

Figure 3.13: Client update process. The figure shows how the client update is propagated to the server.

- **9) Drawing objects**
  Finally, the Activity Manager checks the new information stored in the Object Pool Replica. If the points of interest are in the device's range of view, the Activity Manager will draw them on the screen, otherwise, nothing will be drawn.

3.4.2 Client Updates

The Client Updates serve to give the system the knowledge of the user's preferences and locations. As mentioned before, there are two types of Client Updates: Interests Updates an GPS Updates. The Interests Updates are sent every time the user changes his preferences. A user can alter the traveling mode, the metrics of the distance (minutes or meters), the radius of the search and the categories he is looking for. The GPS Updates are sent every time a user moves ten meters from his current position. These updates only send the new coordinates of the user. Despite the differences between the two types, the way they are propagated is the same. The Client Updates are initiated in the client side and are triggered by the alteration made directly by the user, or by the application itself.

Figure 3.13 illustrates the all process of updating the information of the user in the server. The process starts on the client side and ends on the server, when the information of the user is updated.

The Database Updates process is composed by six main steps, namely 1)*Listening inputs*, 2)*Informing the Session Manager*, 3)*Creating a new request*, 4)*Sending requests*, 5)*Processing new request* and 6)*Storing the new information*. We now describe each step and each involved module of the process:

- **1) Listening inputs**
  The process of sending Client Updates starts in the Activity Manager. This module listens the inputs that come from the mobile device. There are two ways of triggering a Client Update. The first one is by monitoring the sensors of the mobile device, namely the GPS sensor. The second method is to receive direct inputs from the user, which are also passed through the mobile device. The Activity Manager monitors the activity made by the user and, when a change occurs, a Client Update is triggered.

- **2) Informing the Session Manager**
  After realizing the new information about the user, the Activity Manager informs the Session Manager that a new message has to be created.

- **3) Creating a new request**
  The Session Manager will then construct the message to be sent to the server. If the user changed his preferences and defined a new search criteria, then an Interests Update is created with all the new specifications. If the user is moving around, and only his coordinates are changing, then a new GPS Update is created, containing the new position of the user.

- **4) Sending requests**
  The Session manager will then dispatch the message to be sent to the Client module. The message is sent, through the network, to the Server.

- **5) Processing new request**
  After receiving a new request, the Server will create a new thread that will process the message. After realizing if the message received is an Interests Update or a GPS Update, the Server dispatches it to the Session Manager. Here, the update will be processed accordingly to its type.

- **6) Storing the new information**
  The Session Manager will inform the User Interests Filter of the changes made on the user. The Session Manager also requests the Consistency Manager to check if any object of the user is outdated. If the received update is an Interests Update, then the Consistency Manager will verify, with the new specifications, what are the points of interest in the user's surroundings. The new objects will be sent to client as a new Database Update. If the received update is a GPS Update, then the Consistency Manager will check if there are any new points of interest and if the previous ones are still in the consistency limits. More details about the consistency enforcing will be given in Section 3.6. If there is new information to be passed to the user, the server will send a new message to the client. Finally, the User Interests Filter stores the alterations of the user in the Users Database.

3.5 Data Management

In this section we will discuss the management of the data in the system. We will present the representation of the user in system, how it is stored and how Idroid manages the information about him. Then we will discuss the information of each point of interest. We will see what is passed from the server to the client and how does the system manages this information.

3.5.1 Users management

The users are the main focus of this system. It is important to understand their interest in order to provide the information they desire. The understanding of the users interest is also important when applying the VFCdroid consistency model. The system has to know which are the points of interest that are important to the user and which are less important.

As said before, there are two types of Client Updates: Interests Updates and GPS Updates. Only the Interests Updates provide the system with information regarding the user's preferences. The other type of updates, only send the new location of the user. Knowing this, it was important to store the information of each user in the server side. Then, we decided to create a database that stored all the information regarding the user, and named it Users Database.

In Idroid, each user is defined by twelve different parameters. On the following we describe in more detail the structure of each user:

- ***id*:** a unique identifier for each user. This *id* is an integer that is assigned to every user and in each assignment it takes a different value. A user that never used the application starts with *id* "0". After the first request, the server assigns a new *id* to the user and sends it back to the client, piggybacked on the response message. In the client side, the new *id* is stored in mobile device memory card. Therefore,

we can say that the *id* is attributed to the mobile device, rather than the user itself. When the client sends a new Client Update, it will use the *id* stored in the mobile device's memory.

- ***username*:** the name the user wants to be known. It can be modified whenever the user wants to, and has no constrains on the choice.
- ***radius*:** it represents the maximum radius of search. This value is used to filter the points of interest that are far from the user. If the distance between an object and the user is bigger than this value, then the object is discarded. The *radius* can be given in minutes or in meters.
- ***limitMetrics*:** this parameter defines if the user wants the distance from the objects to be calculated in minutes or in meters.
- ***mode*:** defines the traveling mode of the user. There are two types: walking or driving. This is an important value when presenting the points of interest to the user. The difference lies in the limit metrics used. A user that wants the distance in minutes can have different points of interest, depending on the mode he is traveling. For example, a restaurant that is located 100 meters of two users. Both users want this distance in minutes, however, the first user is walking and the second is driving. For the user who is walking, this distance, in minutes, will be bigger than the distance for the user who is driving. This is due to the velocity the users can move in the both modes.
- ***interests*:** the list of categories the user is interest in. This parameter specifies the type of the points of interest he desires. The interests are divided in seven different main groups: *Food and Drink*, *Health and Welfare*, *Transportation*, *Religion*, *Entertainment*, *Local Services* and *Other*(covering non categorized interests such as *atm*, *laundry*, *gas station*, etc.). The user chooses his interest from the categories inside this groups.
- ***zone1*:** A tuple representing the first consistency zone limits. It includes the maximum distance of the zone, the time that the objects in this zone can be without being refreshed, the number of updates that can be lost and not propagated, and the difference an object in this zone can be from the last refresh.
- ***zone2*:** A tuple representing the second consistency zone limits. It includes the maximum distance of the zone, the time that the objects in this zone can be without being refreshed, the number of updates that can be lost and not propagated, and the difference an object in this zone can be from the last refresh.
- ***zone3*:** A tuple representing the third consistency zone limits. It includes the maximum distance of the zone, the time that the objects in this zone can be without being refreshed, the number of updates that can be lost and not propagated, and the difference an object in this zone can be from the last refresh.
- ***latitude*:** represents the last known latitude, in GPS coordinates, of the user.
- ***longitude*:** represents the last known longitude, in GPS coordinates, of the user.
- ***locations*:** A set of points of interest that are in the user's surroundings. The objects present in this field are a copy of the objects stored in the Object Pool Primary. Attached to the information of the point of interest, there is the distance of the user to the object, the date of the last time the object was sent to the user, the number of updates received since then and the percentage of changes the object suffered since the last refresh sent to the user.

With these fields, the server can understand who the user is, and what are his desires. By having a unique id to every client, the information about the user's preferences does not need be sent in every Client Update. If the user is only moving around, the client sends the new coordinates of the user and his id. With this information, the server can connect the id to a user, and understand his interests and what are the

points of interest that are in his surroundings.

As said before, the information about the users is stored in the Users Database. However, in order to optimize the accesses to the database, the users that are currently active and using the application are stored in the memory of the server.

By defining which users are active, we restricted the amount of updates sent by the server to the clients. In other words, whenever a Database Update occurs, the Consistency Manager will only check if the object needs to be refreshed in the active users. This decision reduces the amount of users that need to be verified, reducing the accesses to the database.

To achieve such efficiency, we decided to add a *time to live* on each user. If a user makes a request and he is sill not stored in memory, then it is fetched from the database and stored in the server memory. Then, a *time to live*, with the duration of ten minutes, is added to the user. For that period of time, the server will access its memory whenever information of that user is required. When the server receives an update from the client, the user's *time to live* is extended for more ten minutes. If the user does not send another update in that period of time, then his validity expires and he is removed from the server's memory.

The module responsible for managing the users is the User Interests Filter. Periodically, in every minute, this module checks if there is any user in which the *time to live* as expired. If so, the User Interests Filter removes the user from the list of active users. Until a new update from the client is received, this user will not get the new updates made in his points of interest. Also, when the user is removed, he stops being associated to his points of interest, in the next section this feature will be explained in more detail.

It is important to understand that a user who uses Idroid for the first time, gets his information stored in the database for an indefinite period of time. When a user is removed from the server memory, it does not mean that he is also removed from the database. The information of each user is never lost. When an user decides to use the application one more time, his information will be stored in the database.

3.5.2 Points of interest management

Presenting points of interest to the user is the main goal of Idroid. The only objective of the user, who intends to use this system, is the presentation of the points of interest.

As mentioned before, the points of interest are stored in a Location Service Database. This database can access to any location service, such as Google Maps, Bing Maps, Yahoo Maps, etc. It is important to define the points of interest and present the user the same information, regardless of the location service used.

In Idroid, we defined a point of interest as an object with eleven different parameters. On the following we describe in more detail the structure of each point of interest:

- ***reference*:** a unique identifier for each point of interest. The *reference* is a set of characters that are different for every object. Ultimately, this is what differs the points of interest from each other.
- ***name*:** the business name of the point of interest. This *name* is the common name of the place. For example, if there a point of interest named "Goldman's Pharmacy", then this is the *name* associated to it.
- ***latitude*:** represents the latitude, in GPS coordinates, where the point of interest is located.
- ***longitude*:** represents the longitude, in GPS coordinates, where the point of interest is located.
- ***address*:** contains the physical address of the place. This *address* is equivalent to the "postal address", which sometimes differs from country to country.

- ***phoneNumber*:** contains the point of interest phone number in the international format. This format includes the country code and is prefixed by the plus (+) signal. For example, every phone number in Portugal starts with "+351"

- ***imgUrl*:** contains the url of the icon representing the place. This image is the object that will be displayed on the mobile device's screen.

- ***linkUrl*:** contains the url page of the point of interest. This url is linked to the official site of the point of interest.

- ***rating*:** contains the object's rating. A point of interest can be rated from 0.0 to 5.0, based on the users reviews.

- ***type*:** represents the categories the point of interest is associated to. Each object can be associated to one or more categories. For example a *cafe* can also be categorized as a *restaurant* or a *bar*. The three categories are part of the types of the point of interest.

- ***clients*:** a list with the identifiers of the users that are associated to this point of interest. If the list is empty, it means that no user has this point of interest in his results.

After defining what is a point of interest, it is important to define how the server manage that information. Another important task is to define which of this fields are taken in consideration for consistency purpose. It is also important to define what is the information passed to the client and how he manages that information. Finally it is important to clarify what pieces of information are presented to the user.

We begin by defining how the server manages the points of interest. After retrieving the objects from the Location Service Database, the server stores all the results in the Object Pool Primary. Each point of interest stored in the server memory has a list of users that are associated to him. In order to optimize the memory usage, we decided that, if an object has no user associated to him, then it must be deleted.

The removal of the points of interest, in the server side, is managed by the Object Pool Primary. Periodically, in every minute, this module checks if there is any point of interest with no association to any user. If so, it means that the object is not in the interest of the users and the changes that can be made to him, will not be propagated. Therefore, the Object Pool Primary proceeds to the removal of the object from the memory. When the object becomes part of the results of a user, then it will be fetched again from the Location Service Database.

The points of interest stored in the Object Pool Primary are the ones the users are interested. Therefore, Idroid only needs to check the consistency of these objects. However, not all fields are relevant for the consistency purpose. For example, the *reference* of the object won't change through time, so it does not need be checked for changes. So, for consistency purposes we decided that the changeable fields are: *name*, *latitude*, *longitude*, *address*, *phoneNumber*, *imgUrl*, *linkUrl* and *rating*. So, the Database Monitor will check, for every object in the Object Pool Primary, if there is any change regarding this fields in the Location Service Database. Another field that is considered for consistency is the distance between the user and the point of interest. However, this field is different for every user. The consistency issues will be explained in more detail in Section 3.6.

These eight fields, that are important to check the consistency of the object, together with the *reference* are the ones that are sent to the client. This was also one of the reasons for making such decision. When the server needs to send the points of interest to the client, it creates a response message containing the objects with the information associated to them. If these information changes, than the server must update the user, regarding his consistency limits.

As said before, when the client receives the updates, it stores them in the Object Pool Replica. The server also sends, piggybacked, a list of the locations that were removed from the user's results. With this information, the client can delete the points of interest that are not in the user's surroundings.

For each point of interest, not all information is presented to the user. The client uses the *latitude* and *longitude* to understand where to draw the object, and the *imgUrl* is used to draw the object itself. The *reference* is used only to identify the points of interest. The remaining information, *name*, *address*, *phoneNumber*, *linkUrl* and *rating*, along with the distance to the object, are the only information available for the real user. The way the system draws the objects and how the information is displayed on the screen will be explained on Section 4.

3.6 VFCdroid Enforcement

In this section of the document, we describe how the VFCdroid model is enforced over the multiple updates. We will present how the consistency zones are created and how we assigned different consistency degrees to each one. Additionally, details about the checking of VFCdroid limits and how the systems bounds those limits to the points of interest are presented in this section.

After selecting a point of interest for the users, the server is responsible for enforcing the VFCdroid model. The Consistency Manager is in charge of determining the consistency zone of point of interest, according to the users interests. Then, it is assigned to set of users to whom this point of interest may concern. Each user will have the point of interest assigned with a consistency level and a set of consistency limits.

After assigning the points of interest to the consistency zones of each user, the system will enforce the VFCdroid. Periodically, the Consistency Manager iterates the users and their points of interest in order to update the time parameter and verify if any object has already exceeded the time limit to be propagated to a certain client. For instance, from time to time, we check if the time for the next update is equal or superior than the current time. When the time parameter of any point of interest exceeds the specified time limit to that consistency level zone, the object is elected to be propagated.

Identically, when a new update is received, the sequence parameter and the value parameter of the affected point of interest, also changes. The sequence value is incremented in one unit, and the value is modified depending on the number of changes made to the object. Then, the both parameters are compared to their limit values specified to that zone. If they exceed these limits, the object is elected to be propagated.

3.6.1 VFCdroid consistency zones

After receiving a Client Update, the server needs to assign the points of interest of the user, to each consistency zone. As already mention, there are three different consistency zones.

The distance limits of each zone can be customized by the user. However, to provide a user friendly interaction, some default values were given to those limits. In order to reduce the objects that need higher level of consistency and, therefore, decrease the network usage, we tried to place more objects in the farther zones. Thereby, the first zone is the smaller one, the second zone is the second smaller and the third zone is the bigger one. To calculate the values we proceed like this: the user defines a maximum distance of search. For the first zone, the limit is calculated by multiplying this value for 1/6. The second zone is calculated by multiplying the distance for 1/2. And the third zone is limited by the maximum distance.

After defining the distance limits of each zone, the server needs to assign the points of interest to each zone. To do this, we calculate the distance from the user to the points of interest. Firstly, we check if the calculated distance is smaller than the first zone limit, if not, then we check if it is smaller than the second

Distance Metrics / Zone	Minutes	Meters
Zone 1	[0, 1, ∞]	[0, 1, ∞]
Zone 2	[zone 1 limit, 5, 25%]	[zone 1 limit * 36/40, 5, 25%]
Zone 3	[zone 2 limit, 10, 50%]	[zone 2 limit * 36/40, 10, 50%]

Table 3.1: Consistency values for the user that is walking.

Distance Metrics / Zone	Minutes	Meters
Zone 1	[0, 1, ∞]	[0, 1, ∞]
Zone 2	[zone 1 limit, 5, 25%]	[zone 1 limit * 36/500, 5, 25%]
Zone 3	[zone 2 limit, 10, 50%]	[zone 2 limit * 36/500, 10, 50%]

Table 3.2: Consistency values for the user that is driving.

zone limit. If the value is still bigger, we check if the calculated distance is smaller than third, and higher consistency zone limit. If the point of interest is not inside any of this zones, then it is discarded an not considered an interest for the specified user.

$$zone1\_limit <= distance(point\_of\_interest; User\_X)$$
$$zone2\_limit <= distance(point\_of\_interest; User\_X)$$
$$zone3\_limit <= distance(point\_of\_interest; User\_X)$$

3.6.2 VFCdroid consistency degrees

In order to maintain the points of interest fresh in each client, a set of limits of the VFCdroid are imposed.

The values of the consistency degrees vectors can be customized by the user, however, to keep a user friendly interface, some default values were given.

In order to understand the user's consistency needs it is necessary to understand if he is walking or driving his car. This aspect is very important because it will define the velocity the user is moving, changing the parameters of the consistency vector. For example, the distance between the user and a point of interest can be traveled more rapidly if a user is driving his car, than if he is walking. This means that this point of interest has to be refreshed more often when the user is driving than when he is walking. Also, the distance between the user and the point of interest can be given in minutes or meters. For each profile we have given values to the three main divergence criteria of the VFCdroid, namely time(θ), sequence(σ) and value(ν).

Table 3.1 and table 3.2 represent the consistency degrees of a user that is walking and driving, respectively. In each case, any object in the zone 1 will be refreshed every time it gets updated. The values calculated for the time limit correspond to the minutes a user takes to reach the current zone. We estimated that the average speed for a person walking and for a person driving is 4 km/h and 50 km/h, respectively. The rest of the calculus are simple math.

3.6.3 Checking VFCdroid Limits

Time

The time limit of the VFCdroid consistency vector defines the maximum time an object can be without being refreshed with its most recent value. To enforce the time limits, we set a mechanism that periodically

verifies, for each point of interest, if the time limit defined for the object has already been exceeded.

To create such mechanism, we created a thread in the Consistency Manager that runs in periods of one second. This function checks the current time and, for every object, compares it to the time limit of the object. This comparison is made in order to understand if the time limit has been exceeded.

$$Current\_Time > Time[distance(point\_of\_interest; User\_X)]$$

Sequence

The sequence limit is defined as the maximum number of updates an object can receive without being refreshed. The sequence limits are checked in consequence of either the arrival of a GPS Update or a Database Update. When a GPS Update arrives, the distance from the user to the points of interest changes. Therefore, the Consistency Manager needs to verify if the objects are still in the same zone and change the distance parameter. Every time a Database occurs, there are alterations made to the points of interests. In both cases, the sequence parameter is incremented in one unit.

This limit is checked simply by comparing the number of undelivered updates of a certain point of interest with the maximum sequence limit defined by the according sequence divergence limit.

$$\#Updates[point\_of\_interest] > Sequence[distance(point\_of\_interest; User\_X)]$$

Value

The value limit is defined as the maximum difference an object can be from the last refresh. As the sequence limits, the value limits are checked in consequence of either the arrival of a GPS Update or a Database Update. The behavior in the arrival of each update is the same as when checking the sequence limits. However, instead of incrementing one unit at a time, the value parameter is incremented depending on the number of changes made in the object.

This limit is checked by comparing the ratio between the changes made to a point of interest and the total of fields that can be changed, with the maximum value limit defined by the according value divergence limit.

$$\frac{\sum_{i=1}^{\#changes} update[point\_of\_interest[i]]}{\#changeable\_fields} > Value[distance(point\_of\_interest; User\_X)]$$

Chapter 4

Implementation

In this chapter we present the most relevant details of the implementation of our solution.

In this work we implemented a prototype, called Idroid, that is capable of presenting to the user points of interest in his surroundings. Idroid takes in consideration the user's interests displaying only relevant information. As said before, the system is based on a client-server architecture. The Client is responsible for getting the interest of the user and display the results on the mobile device's screen. The Server side is responsible for understanding which are the points of interest the user is interested in, and to maintain the information about them up to date.

4.1 Client

In this section we overview the client node implementation and describe its most interesting features.

The client of this system was implemented in Android, version 4.0.3. The advantages of using Android are several, including:

- one of the most popular mobile operational systems;
- similar programing language to Java;
- wide set of tools to deal with positioning and heading;
- wide set of tools for drawing images on the screen;
- building GUI using predefined Android layouts.

The main objectives of the client is receiving the interests of the user and drawing the points of interest that appear on the mobile device's screen.

4.1.1 Detecting User's Position and Heading

In order to understand what are the points of interest to draw, it is important to realize where the user is and what is in his range of view.

To find the user's coordinates we use Android's *location* library. This library accesses the mobile device's GPS sensor. It also allows us to periodically check the movements of the user. In order to receive new points of interest on the move, we defined a periodicity for the changing of coordinates. If the user changes his locations in ten meters, than the client will inform the server of the user's new position.

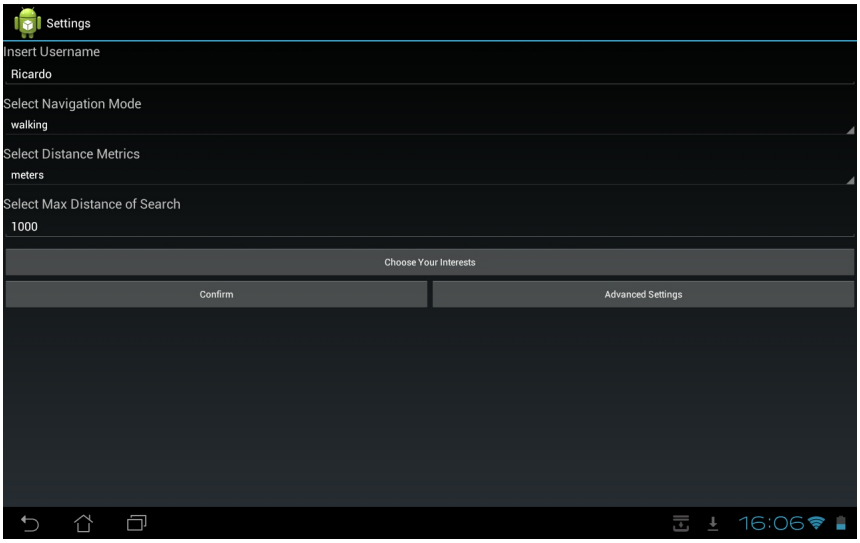

Figure 4.1: The "Settings" screen where the user defines his search criteria.

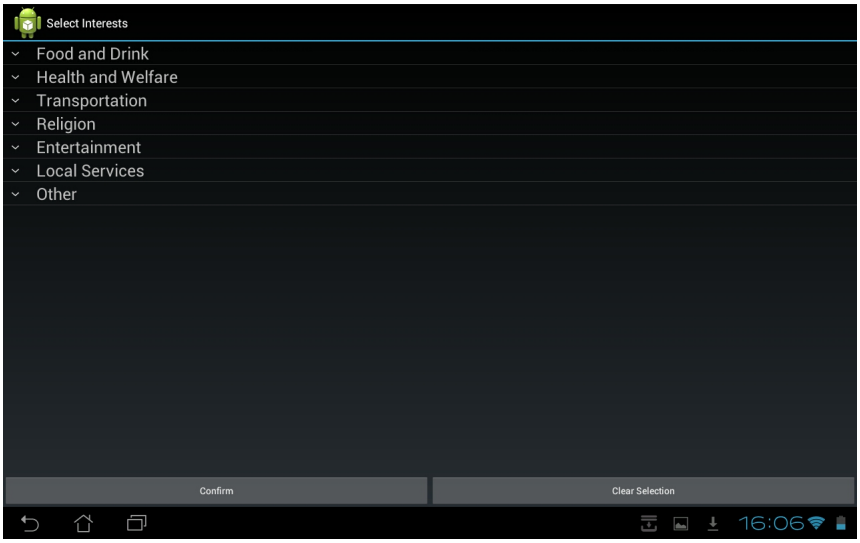

Figure 4.2: The "Select Interests" screen where the user can select his interests.

Determining the heading of the user also required access to the hardware of the mobile device. In this case we used the Android's *hardware* library. This library allowed us the access the accelerometer and the magnetometer sensors. It also provided a *Sensor Manager* that periodically checks the values of the two sensors. Every time their values changed, we needed to check if the displayed points of interest were still valid. However, the accelerometer and magnetometer of the mobile devices tend to be very inconsistent. Therefore, we applied a low pass filter in the received values, in order to eliminate small variations. It is important to clarify that this inconsistency on the sensors is only due to the hardware itself. The only thing we can do is try to correct it, implementing filters.

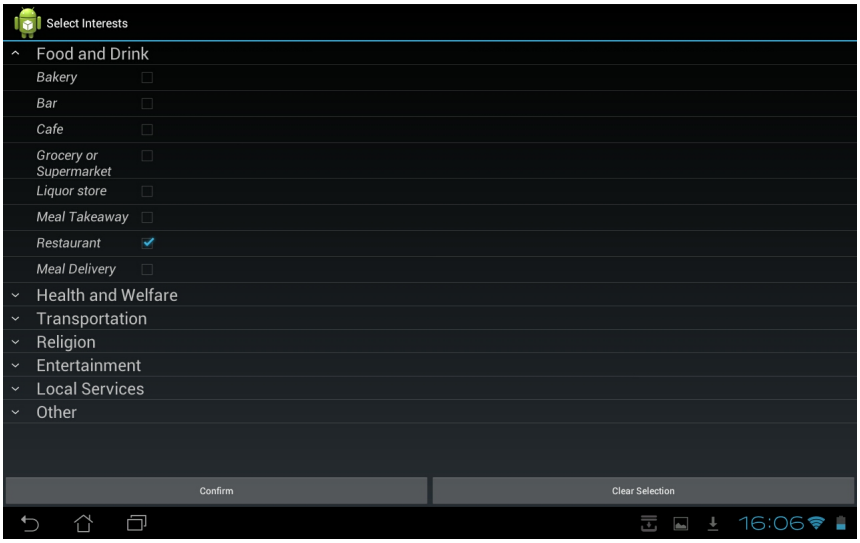

Figure 4.3: An example of an interest selected.

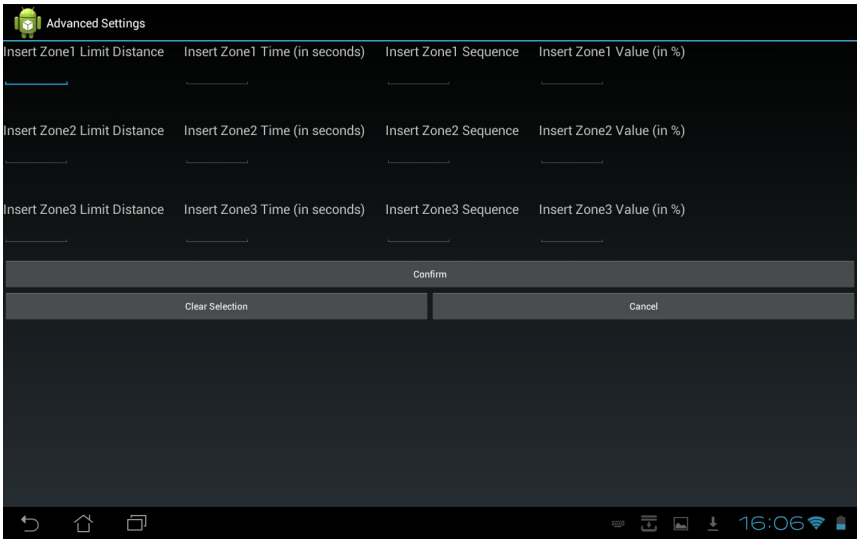

Figure 4.4: The "Advanced Settings" screen where the user can define the VFCdroid model consistency limits.

4.1.2 Interests Upload Interface

To understand the user's interests we implemented three different interfaces. One is for the user to define his search criteria. The second is for selecting the categories he is interested in. Finally, the third one, is an optional interface. It serves to define the consistency limits for each zone of the VFCdroid consistency model.

Figure 4.1 shows the screen where the user selects his search criteria. This screen is known as "Settings". As we can see, the user can modify the values of four fields. The first one is the name the user wants to be known. The second field is to define what is the traveling mode of the user. There are only two traveling modes available: walking or driving. The third one is to define the distance metrics the user wants. Again, there are only two choices: meters or minutes. Finally, the user needs to provide the maximum distance of

Figure 4.5: Presentation of the point of interest in the mobile device's camera.

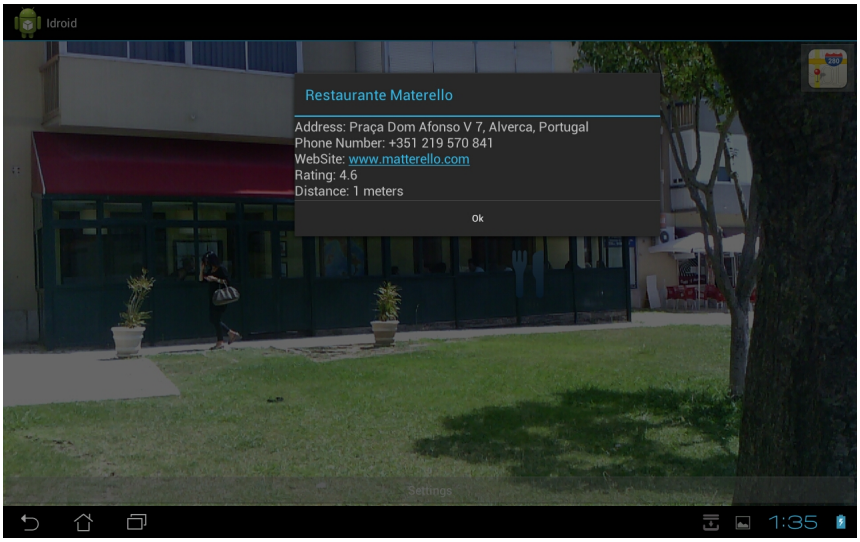

Figure 4.6: The pop-up box with the information regarding the point of interest.

search. After this the user can choose is interests by clicking the "Choose Your Interests" button.

The interests screen of Idroid is presented in Figure 4.2. As we can see, there are seven different types of interests. If the user taps on one of them, a set of interests appears. For a user to select an interest, it only has to tap the category he desires. This process is shown on Figure 4.3. After selecting his interests, the user can click on the "Confirm" button.

The third screen is the "Advanced Settings" screen. This is an optional feature where users can customize the consistency model limits. Figure 4.4 shows this screen. As we can see, for the three zones, a user can specify the range of the zone and its time, sequence and value limits. If the user does not understand the concepts he is choosing, he can always cancel the operation and return to the "Settings" screen.

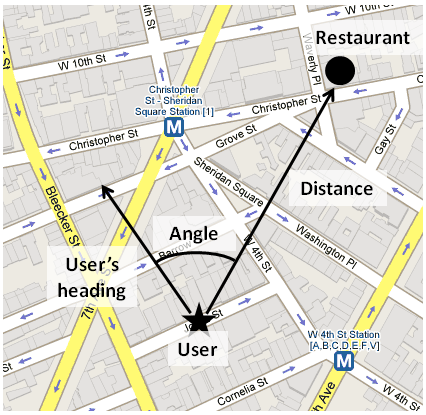

Figure 4.7: Distance and angle between the user and a point of interest, according to his heading.

Only after all the fields of the "Settings" screen are filled and the interests are selected, the user can click on the "Confirm" button. Then the points of interest will be shown on the mobile device's screen.

4.1.3 Displaying Points of Interest

The objective of Idroid is to present the points of interest in the mobile device's screen. One of the challenges of doing this is to understand where to draw the objects.

Figure 4.5 shows a point of interest overlapped in the camera of the mobile device. In order to understand where to draw it, we used simple trigonometry. First, we calculated the angle between the user's heading and the point of interest, see Figure 4.7. Then, using this angle and distance to the object in a straight line, we projected the position of the point of interest, using a projection formula[1]. In the end, if the object is on the camera's range of view, then it is drawn.

Idroid also allows the user to see information about a certain point of interest. To do this, he only needs to tap the object. Then a pop-up box will be displayed over the selected point of interest. Figure 4.6 exemplifies this process. As we can see, the information displayed consists on the place's name, its address, the phone number, the link to its official web page and the rating of the place. An additional feature available is the redirection to the web page of the point of interest. To do this the user only needs to click the url link.

4.2 Server

In this section we overview the server node implementation and describe some of its most interesting details. We will understand how it gets the points of interest and he calculates the distance between the user and those points of interest.

The server of Idroid runs as a daemon on the server machine and it is implemented in Java, version 6.0. It can receive requests from multiple users simultaneous. We support this by using a pool of threads where each thread is assigned to a new request. Once the request is processed, the thread is destroyed.

In order to store the users and their interests, the User Interests Filter accesses to a MySQL database. As mentioned before, this database is named Users Database and stores the interests of every user that made a request to the system. The information regarding the users are stored as an object. To identify the user, an unique identifier is used. So, in the database, every user is represented by a MySQL data type named long

[1] http://membres.multimania.fr/amycoders/tutorials/3dbasics.html

BLOB and it is identified with an unique integer. The queries made to the database are done in two steps. First, it is necessary to connect Idroid to the MySQL Server. This is done through the Connector/J[2] driver. Then, the User Interests Filter uses the JDBC API[3] to make the requests. The queries made are done using only the unique identifier of every user. When a change occurs in his interests, the object representing the user is replaced by a new one, with new information and the same identifier.

To provide the points of interest to each user the server makes use of the Google Places API[4]. This api allows to search points of interest around a certain location. It also allows to search the points of interest by type. So, for each user, we only get the places that are in his surroundings and the ones he is interested in.

To check the real distance between the user and a point of interest, we accessed the Google Distance Matrix API[5]. This api takes into account the possible routes for the user, and calculates the minimum distance he needs to travel, to reach a certain point of interest. The values of the distance are given in both seconds and meters. The api also allows to get the distance when the user is walking or when he is driving. It takes into account the direction of the streets. The Google Distance Matrix API calculates the distance only by selecting the possible streets the user can pass. For example, if a car is not allowed in a certain road, than, that road is not part of the calculated distance.

[2]driverhttp://dev.mysql.com/doc/refman/5.1/en/connector-j.html
[3]http://www.oracle.com/technetwork/java/overview-141217.html
[4]https://developers.google.com/places/documentation/
[5]https://developers.google.com/maps/documentation/distancematrix/?hl=pt-br

Chapter 5

Evaluation

In this chapter we present the evaluation, both qualitative and quantitative, of our prototype.

First, we will present the evaluation made by real users, in qualitative terms, when comparing Idroid to Layar.

Then, we present the results of the quantitative analysis regarding bandwidth usage gains and savings in the presentation of non-relevant information, when using Idroid with VFCdroid and without VFCdroid.

5.1 Qualitative Evaluation

In this section we are going to present the results of the qualitative evaluation of Idroid.

Idroid and Layar are two applications that work in similar way. They produce the same output: presenting points of interest in the user's surroundings. Also in both applications, users submit their interests and the results are displayed in the mobile device's screen.

Since the systems are similar, we decided to qualitatively compare our prototype to Layar. To do this, we handed both systems to a group of real users. After performing a set of tasks with both applications, the users answered a questionnaire were they evaluated the systems.

The main objective of this evaluation is to provide an analysis over the user's experience regarding the search for points of interests, the response time and the presented results. Additionally, we also want to analyze the distances to the points of interest calculated by both applications. Finally we want the see if the presented results are relevant or not to the user.

The questionnaire made to the users can be found in Appendix A.

5.1.1 Test Group

For this test we selected a group of 26 subjects. These subjects used the application in two different cities. Half of the users used it Lisbon and the other half performed the test in Alverca.

Before starting the test, we asked each user some questions, in order to understand the group of users. The results of these questions can be found in Appendix B.

The subjects were composed mostly by users with ages around 21 and 35 years old. Most of them, around 88%, were college students, or had already graduated. In terms of gender, the group was almost equally

divided, there were 14 females and 12 males.

Regarding the use of mobile devices, almost half of the group, around 42%, possesses a smartphone or a tablet. However, all the users have already worked with such device. Only 23% rarely uses a smartphone or a tablet.

Despite this knowledge about mobile devices, only one user has admitted that he already used a Location Based Service.

5.1.2 Performed Tasks

The test presented to the users was divided in three different tasks. Each task was performed equally in Layar and Idroid. In the end of each task the users answered three questions in order to evaluate the systems.

For each question the users had a scale where they marked the number they desired, from 0 to 5. In this scale, 0 meant "I completely disagree" and 5 meant "I totally agree".

The first task was about finding points of interest. The task stated:

"After launching the application, search for a restaurant in a radius of 500 meters."

With this task, we wanted to find out if the users had any problem working with the applications. We also wanted to understand if the response time was acceptable and if the displayed results were graphically appellative.

In the second task we focused on the distance to a certain point of interest. The suggested task was:

"Regarding the results of the first task, select a point of interest and check its distance. With the application turned on, go to that point of interest."

In this task, we compared the distances calculated by Idroid and Layar. We asked the user if, in his way to the point of interest, the distance was updated. Another question made was if the distance presented corresponded to the distance the user traveled. This question is a little relative. Users do not know the exact distance they traveled. We wanted to perform this task in minutes, instead of meters, however, Layar only provides the searches to be made in meters. We also asked the users if new points of interest appeared on the screen.

In the third task we wanted to see if there was any result that was not relevant to the user. The third task suggested:

"Go back to the starting point. From the displayed results, select the one that is farther away. Check the distance presented. Now, go to the selected point of interest."

In this third task, we firstly asked them if the distance traveled was similar to the distance presented by both applications. Then we asked if the distance traveled was smaller or equal to the radius of the search. Finally, we asked the users if that particular point of interest should be in the displayed results.

5.1.3 Result Analyses

After the users answered the questionnaire, we processed their answers and compared the two systems. To do this we added all the values each user gave to each question. For example if ten users marked the

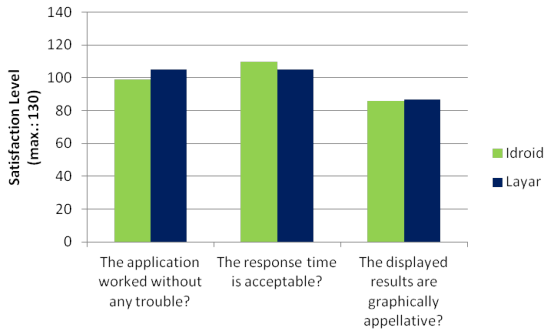

Figure 5.1: Results of the evaluation of the users after completing the first task

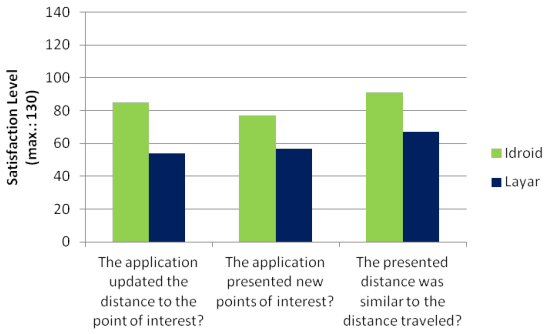

Figure 5.2: Results of the evaluation of the users after completing the second task

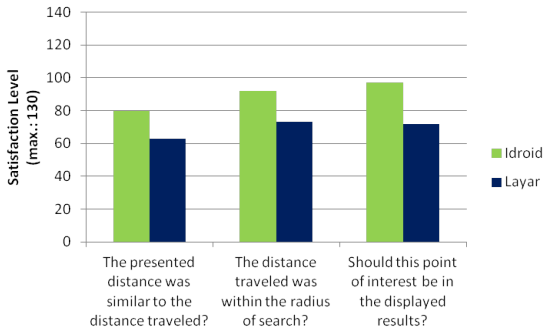

Figure 5.3: Results of the evaluation of the users after completing the third task

number five in a certain question, for Idroid, than, in overall the system has an evaluation of 50. So, for each question the maximum result a system can have is 130. The tables with the results for each task and for each question can be found in Appendix B.

Task 1

Analyzing the results of performing this task, see Figure 5.1, we can see that, for the users, the two systems are very similar.

For the first question we can see that both applications worked properly, without any problems. The difference of the values given for both applications for this question is too small to allow us to make distinction between them.

Regarding the second question, we can see that the users are satisfied with the response time of both applications. Also, as in the first question, it is not possible to make difference between Layar and Idroid. Both have similar acceptance values.

Finally, we can see that users think that the objects drawn are graphically appellative. The level of acceptance is not as higher as in the other two questions, but it is still a positive result. Again, there is no difference between the two systems.

Task 2

Figure 5.2 shows the evaluation given by the users after performing the second task.

Regarding the updates of the distance to a certain point of interest, users preferred Idroid to Layar. In Idroid, users stated that, through their journey to the point of interest, the distance was being periodically updated. Most of the users stated that the same thing did not happen in Layar. The same thing happen in the displaying of new points of interest. In this case, users also stated that Idroid presented more often new points of interest rather than Layar.

Finally, analyzing responses given, users believe that the distance presented by Idroid is more accurate than the distance presented by Layar.

Task 3

The results of the evaluation of the third task, presented on Figure 5.3, shows us some differences between the two systems.

Regarding the distance traveled, we can see that users believe that the distance presented by Idroid is more accurate than the one presented by Layar. Also, regarding Layar, we see that for almost half of the subjects, the distance traveled was higher than the search radius. The same does not happen with Idroid. In this case, the users believe that the distance traveled is within the search radius.

Finally, the level of acceptance of the point of interest in the displayed results is higher in Idroid than in Layar. In Layar, the users believe that the point of interest is too distant, and, therefore does not belong to the results.

5.1.4 Summary

In this section we analyzed and compared, qualitatively, our system, Idroid, with another similar application, Layar. After the analysis of the evaluations of the users, we saw some advantages of Idroid regarding Layar. The following conclusions can me made:

- **Idroid response time is acceptable.** Idroid and Layar presented very similar levels of acceptance of response time. Also, these levels have high values, so, we can conclude that Idroid has an acceptable response time.
- **Idroid updates the distance to points of interest and provides new points of interest on the move.** In our prototype, the distances to the points of interest are updated periodically. Also, as the users move, new points of interest are presented. As users stated, the same does not always happens in Layar.
- **Idroid calculates a more accurate distance to the points of interest.** By taking into account the obstacles of the real world, Idroid provides a more accurate distance to the points of interest. In Layar, the distance is calculated in a straight line, regardless of the obstacles of the real world.
- **Idroid displays less non relevant information.** By calculating a more accurate distance, Idroid removes the points of interest that exceed the radius of search. This way, our system reduces the amount of non relevant information displayed to the users.

5.2 Quantitative Evaluation

In this section we present the results of some conducted tests. These tests had the objective of determining the savings in the using of network resources and the savings in non-relevant information passed to the users.

5.2.1 Experimental Settings

In order to fully evaluate the system, we simulated several users performing tasks with the application. The users sent their request to the server, which was in charge of responding the requests.

We also simulated a set of changes made to the points of interest information. Due to the limitations of the Google Places API[1] and the Google Distance Matrix API[2], we needed to simulate these location services. We created two databases, one for each api, that were filled with information retrieved from those two apis. Doing this, also allowed us to make the updates of the information we desired. The server had the responsibility of checking these updates and propagate them to the proper users.

To perform these tests, two dedicated machines where required. The first one, was used to simulate the users. The other machine stored the Idroid server and the database containing the points of interest. The network and the number of points of interest measures, were always performed in this machine. Both machines were running Windows 7 Premium (64-bit) on a Intel Core I5 - 3210M, 2.5GHz with 4GB of RAM. Finally, a 100mbps full-duplex Ethernet LAN was used for the tests.

5.2.2 Compared Solutions

The solutions used in these tests where the Idroid, fully implemented, and the Idroid without the VFC-droid. We used both system to compare results, and determine the gains of the fully implemented prototype.

We made this decision because it represents the exact differences of using VFCdroid. The comparison of the two systems permitted to compare two main issues:

- **Relevant information** - Compare the amount of relevant information passed through the network in both systems.
- **Total consistency vs VFCdroid** - Compare the results of the VFCdroid model against the total consistency model.

5.2.3 Workload Description

To exercise the system, and to make the tests as real as possible, we created three different groups of users, each one with different interests. The first group, is a group of college students and they start using their application in one of the entrances of Instituto Superior Técnico. The second group, is a group of employees that work in a building near Picoas. The last group, is a group of tourists that are visiting Lisbon and they start their tour in an hotel, near Avenida da Liberdade.

As said before, the groups have different interests. We summed the interests of the average college students and select the following categories: *university*, for students who want to know were the other universities are; *library*, to study and get the material for their exams; *book store*, to by the books for each course; *cafe*, to take their breakfast or simply to drink coffee, *bar*, to hang out with their friends; *subway station*, the most common public transport to move around the city.

The second group, the employees, are people that go to work every day, and have some needs. We summed their interests in: *cafe*, for the coffee break; *restaurant*, to lunch or dinner; *bus station*, for users who need to catch the bus to get home; *subway station*, for users who need the subway to go home; *train station*, for users who use the train; *parking*, for users who go to work with their car; *grocery or supermarket*, to buy supplements for the house.

[1] https://developers.google.com/places/documentation/
[2] https://developers.google.com/maps/documentation/distancematrix/?hl=pt-br

Zone	**Time** (θ)	**Sequence** (σ)	**Value** (ν)
Zone 1	0 minutes	1 update	∞
Zone 2	2 minutes	5 updates	25%
Zone 3	6 minutes	10 updates	50%

Table 5.1: Consistency values for User 1 that is walking and wants the distance in minutes.

Zone	**Time** (θ)	**Sequence** (σ)	**Value** (ν)
Zone 1	0 minutes	1 update	∞
Zone 2	3 minutes	5 updates	25%
Zone 3	9 minutes	10 updates	50%

Table 5.2: Consistency values for User 2 that is driving and wants the distance in meters.

Zone	**Time** (θ)	**Sequence** (σ)	**Value** (ν)
Zone 1	0 minutes	1 update	∞
Zone 2	2 minutes	5 updates	25%
Zone 3	6 minutes	10 updates	50%

Table 5.3: Consistency values for User 3 that is driving and wants the distance in minutes.

Zone	**Time** (θ)	**Sequence** (σ)	**Value** (ν)
Zone 1	0 minutes	1 update	∞
Zone 2	1 minute	5 updates	25%
Zone 3	3 minutes	10 updates	50%

Table 5.4: Consistency values for User 4 that is driving and wants the distance in meters.

Finally the third group, the tourists. This group has interest in the main attractions of the city. We categorized them in: *restaurant*, to find traditional restaurants, in order to try local food; *art gallery*, to see art exhibitions; *museum*, to visit local museums; *zoo*, to take a trip to the zoo; *shopping mall*, to buy gifts, souvenirs or other things; *embassy*, to understand were their embassy is.

After specifying the interests of the tested users, we filled one of the created databases (see section 5.2.1) with 1250 points of interest. The other database was filled with information regarding the distance of each user to all the points of interest.

Finally, in each test we increased the number of users. We started with 4 user, with 4 different consistency zones, and in the end we had 1000 times the initial users, which means 4000 user. The detailed results of each test can be found in Appendix C.

5.2.4 VFCdroid Limits

To evaluate the system, we used four types of user, one for each profile. The first two users, User 1 and User 2, are walking and their search radius is 12 minutes and 1200 meters, respectively. The third and the forth users, User 3 and User4, are driving and their search radius is 12 minutes and 5000 meters, respectively. Tables 5.1, 5.2, 5.3 and 5.4 represent the configuration values of the VFCdroid mode, for each user:

5.2.5 Information Download Test

We start by discussing the results obtained for the basic test of downloading all the points of interest. In this situation, the users start the application for the first time and then, Idroid downloads all the information

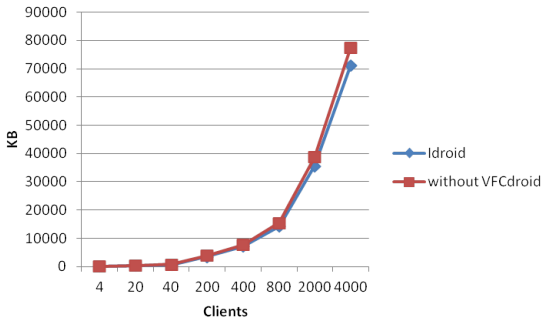

Figure 5.4: Bandwidth usage (KB) for sending the points of interest to the students group.

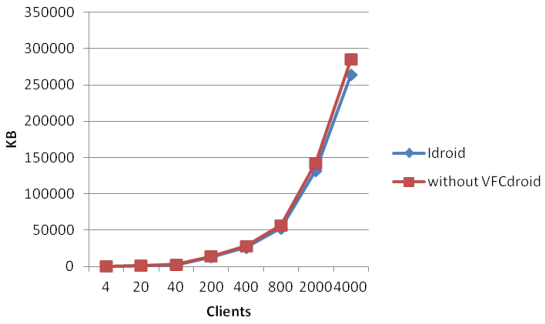

Figure 5.5: Bandwidth usage (KB) for sending the points of interest to the employees group.

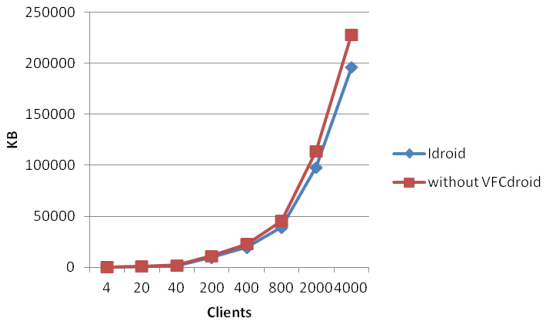

Figure 5.6: Bandwidth usage (KB) for sending the points of interest to the tourists group.

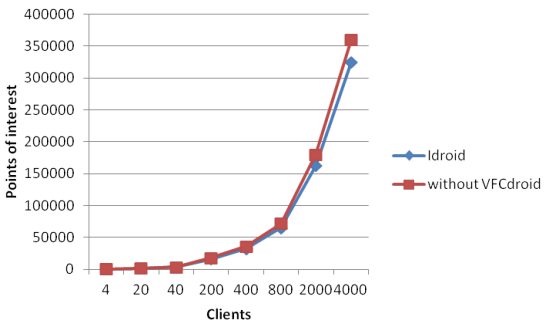

Figure 5.7: Total number of points of interest sent to every user of the students group.

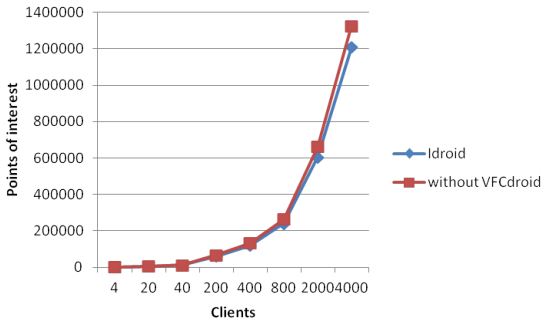

Figure 5.8: Total number of points of interest sent to every user of the employees group.

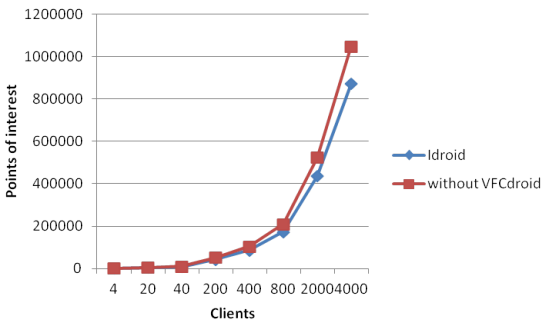

Figure 5.9: Total number of points of interest sent to every user of the tourists group.

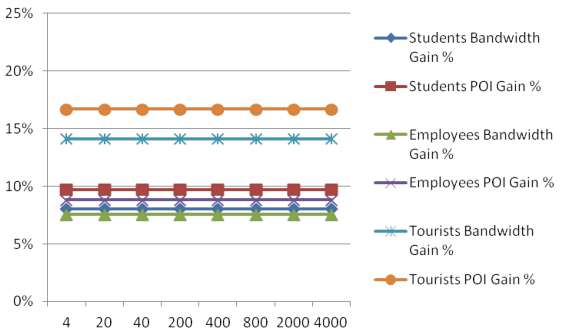

Figure 5.10: Percentage of gains of Idroid regarding the points of interest sent to the three testing groups.

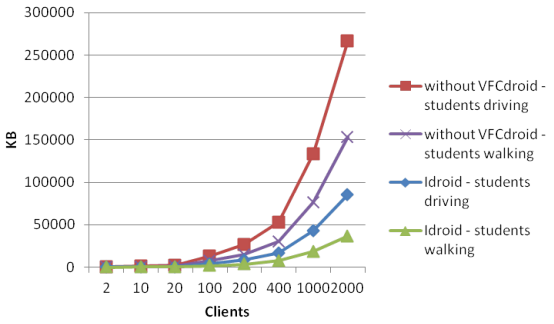

Figure 5.11: Bandwidth usage (KB) for sending the points of interest to the students group after traveling a small route.

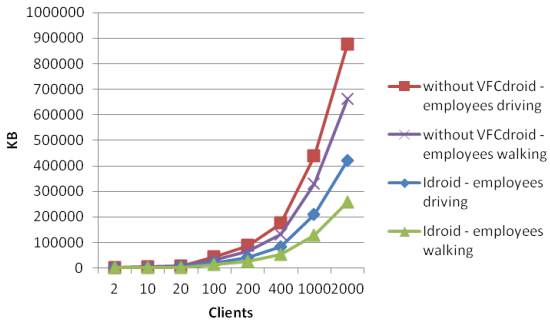

Figure 5.12: Bandwidth usage (KB) for sending the points of interest to the employees group after traveling a small route.

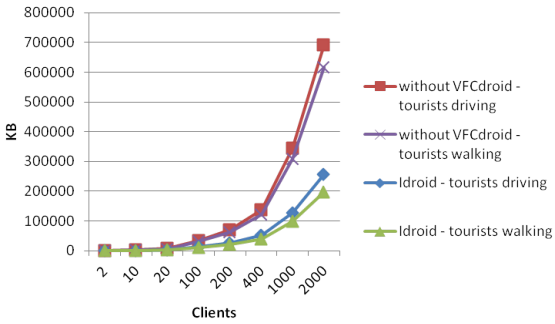

Figure 5.13: Bandwidth usage (KB) for sending the points of interest to the tourists group after traveling a small route.

to the user, regarding his interests. The results of the bandwidth usage, for the students, employees and tourists, are illustrated in Figures 5.4, 5.5 and 5.6, respectively. The number of points of interest sent in each system are shown on Figures 5.7, 5.8, 5.9.

The presented graphs show that for all the groups, the bandwidth used to download all points of interest is approximately the same (Figure 5.4, 5.5 and 5.6). This is due to the identical behavior of both systems. The information to download is the same, therefore the bandwidth usage will be almost equal.

The difference lies in the number of points of interest passed to the user. Idroid, with VFCdroid, sends less points of interest than the system without the consistency model (Figure 5.7, 5.8 and 5.9). This fact can be explained by the VFCdroid consistency tubes. They take into account the obstacles of the real world, removing the unaccessible places. Without using this consistency model the system will get all the points of interest that are in the users radius. Therefore, the excess sent by the system without VFCdroid is considered non-relevant information for the user.

Finally, with this test we can see that the performance of Idroid does not decay with the increasing number of users, regarding a system without VFCdroid (Figure 5.10). Both systems increase the number of points of interest and the bandwidth usage, however there is always a slight difference (from 7% to 17%)

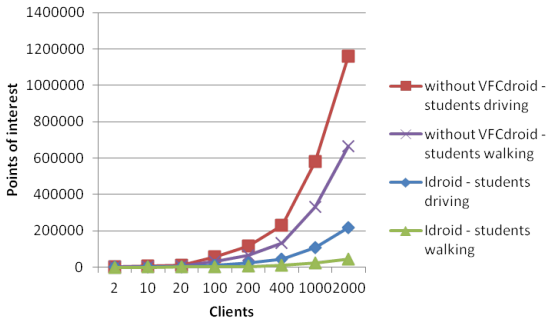

Figure 5.14: Total number of points of interest sent to every user of the students group after traveling a small route.

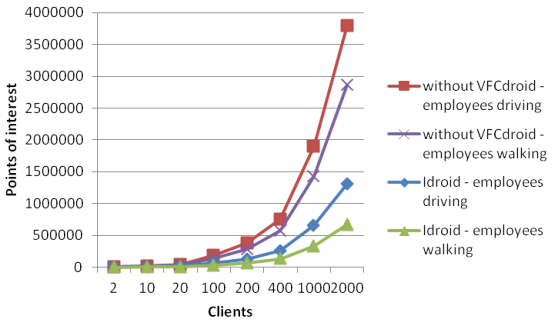

Figure 5.15: Total number of points of interest sent to every user of the employees group after traveling a small route.

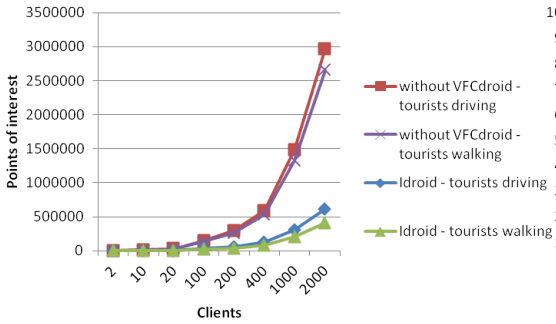

Figure 5.16: Total number of points of interest sent to every user of the tourists group after traveling a small route.

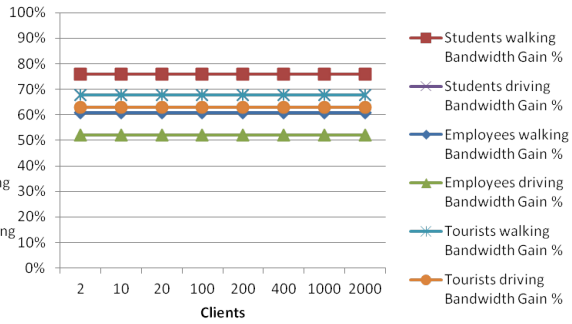

Figure 5.17: Percentage of gains of Idroid regarding the bandwidth usage, after the three groups traveled a small route.

between the two systems, in favor of VFCdroid.

5.2.6 GPS Updates Test

In this test we intended to get the results for users that are on the move. We created a short route for the three groups. Each route is composed by five different places. In each place, a user will send a new request to the server, only giving his new coordinates. After completing the course, we measured the total bandwidth usage and the number of points of interest sent.

Figures 5.11, 5.12 and 5.13 show the results of bandwidth usage by the students, employees and tourists, respectively. Figures 5.14, 5.15 and 5.16 show the total amount of points of interests that were sent to the users after the completion of the route.

The results show that, Idroid performance is always better than the system without VFCdroid. There are two reasons for this fact. Firstly, as seen on the previous test (Section 5.2.5), the amount of points of interest is different in both systems. Idroid has less points of interest than the system without VFCdroid. Therefore, if the user moves, the number of new points of interest to be sent is lower in Idroid. The second reason is due to the presence of a consistency model. Every time a GPS Update occurs, the distance between the user and his points of interest changes. Idroid only updates this information if it is critical, regarding the values of the consistency model. In a system where there is no consistency model, the new distances will

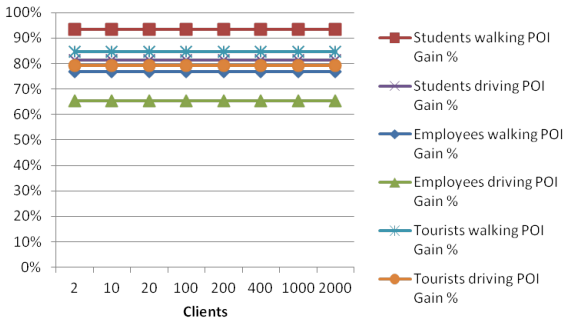

Figure 5.18: Percentage of gains of Idroid regarding the points of interest sent, after the three groups traveled a small route.

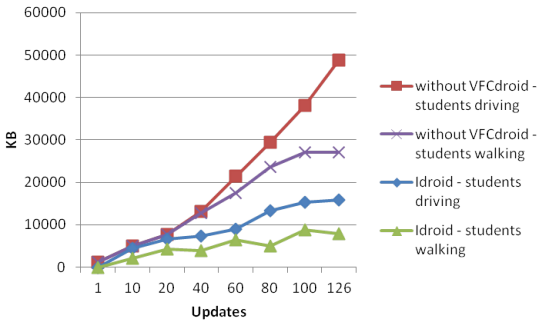

Figure 5.19: Bandwidth usage (KB) for updating the points of interest of the students group.

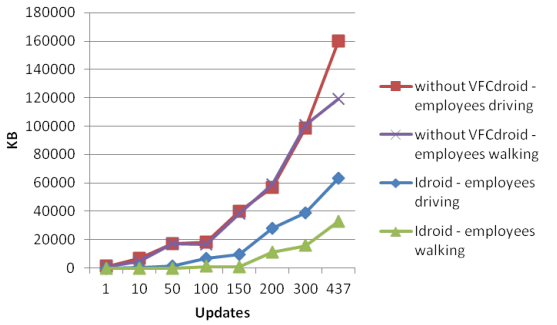

Figure 5.20: Bandwidth usage (KB) for updating the points of interest of the employees group.

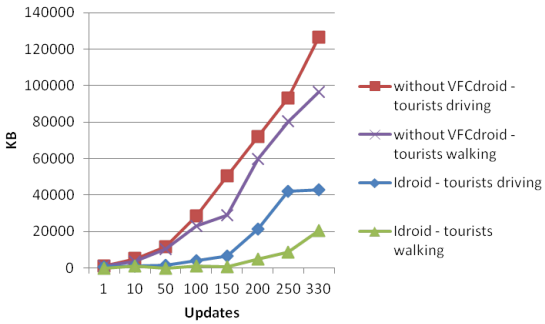

Figure 5.21: Bandwidth usage (KB) for updating the points of interest of the tourists group.

always be propagated to the users.

In comparison to a system without VFCdroid, Idroid shows a total saving between 52% and 72% in terms of bandwidth usage(Figure 5.17). Regarding the total number of points of interest passed to the users, Idroid results show a gain between 65% and 93%(Figure 5.18).

5.2.7 Database Updates Test

The objective of this test is to understand the behavior of the system when several updates are made to the database. In the end of each test we measured the bandwidth usage and the number of points of interest sent to the client.

To accomplish this, we made modifications to the information that was in the interest of the users. This means that for students we updated the points of interest categorized as being *university*, *library*, *book store*, *cafe*, *bar* and *subway station*. For employees we only updated the ones categorized as *cafe*, *restaurant*, *bus station*, *subway station*, *train station*, *parking* and *grocery or supermarket*. Finally, for the tourists we updated every place categorized as *restaurant*, *art gallery*, *museum*, *zoo*, *shopping mall* and *embassy*. The number of points of interest available for students, employees and tourists were 126, 437 and 330, respectively. It is

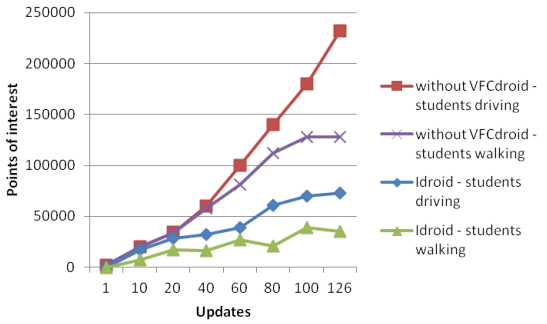

Figure 5.22: Total number of points of interest that were updated for the students group.

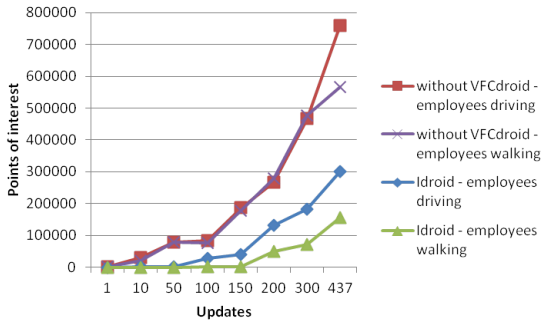

Figure 5.23: Total number of points of interest that were updated for the employees group.

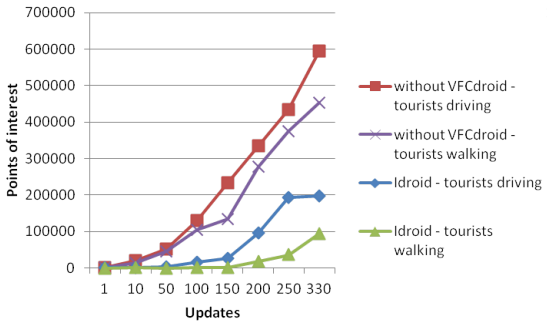

Figure 5.24: Total number of points of interest that were updated for the tourists group.

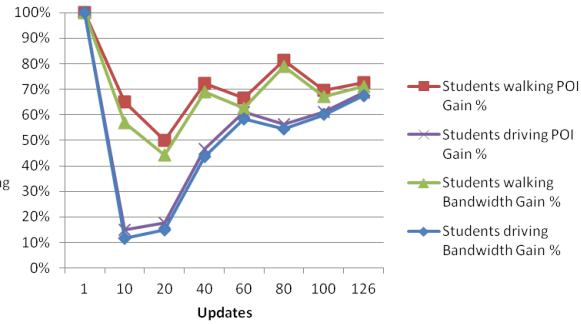

Figure 5.25: Percentage of gains of Idroid regarding the updates of the students group.

important to clarify that not all the points of interest are part of the results of the users, however we did not exclude them. The only restriction made to the updates, was that the first one had to be done in a point of interest that was part of the results of every user. Then, updates were made randomly. In each update, we altered the rating of the place. It is also important to clarify that, for this test, we simulated 4000 clients using both systems.

The results of the bandwidth usage, for the students, employees and tourists, are presented on Figures 5.19, 5.20 and 5.21, respectively. In the same order, Figures 5.22, 5.23 and 5.24, represent the total number of points of interest that were updated, for each group. Also by the same order, Figures 5.25, 5.26 and 5.27 show the percentage of gains of Idroid regarding bandwidth usage and the number of places updated, for each group.

In comparison with the system without the consistency model, we can see that Idroid uses less bandwidth and updates less points of interest. Analyzing the bandwidth usage (Figures 5.19, 5.20 and 5.21), we can see that, with the growing number of updates, the lines of Idroid grow much slower that to the system without VFCdroid. This means that, with the increasing number of updates, the difference of bandwidth between the two systems will grow. The same fact can be seen regarding the number of points of interest updated (Figures 5.22, 5.23 and 5.24). This differences can be explained by the presence of the VFCdroid

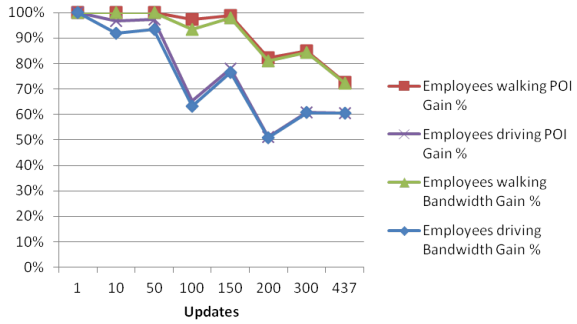

Figure 5.26: Percentage of gains of Idroid regarding the updates of the employees group.

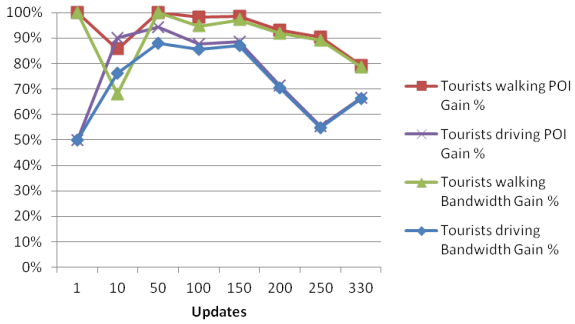

Figure 5.27: Percentage of gains of Idroid regarding the updates of the tourists group.

consistency model. This model postpones non critical updates and, therefore, not all the points of interest will be immediately. Without VFCdroid, each update to the database will be propagated to the respective users.

Analyzing the percentage of gains of Idroid, we can see that, for students (Figure 5.25), the gains for the first 20 updates are not very high (between 11% and 18%). This can be explained by the number of points of interest located near the user. In this case the consistency limits of the first two zones (see section 5.2.4) are exceeded, therefore the updates must be propagated. However, with the increasing of the number of updates, we can see that the gains also increase, reaching a maximum of 69%. Analyzing the other two groups (Figures 5.26 and 5.27), we may reinforce this explanation. In both groups we can see that the gains sometimes get higher, and sometimes get lower, never getting below the 50%. Agian, this is due to the users consistency limits, and when they are exceeded, the updates must be sent.

5.2.8 Summary

In this section we summarize the results of each test and discuss some conclusions we can make of the performance of both systems.

In comparison to a system without VFCdroid, we can see that Idroid presents less points of interest to each user, reducing the amount of data to be sent to each one.

Regarding moving users, we also noticed that Idroid sends less updates to the users, and therefore, reducing the bandwidth usage.

Finally, we also noticed that, when updates are made to the database, the number of points of interest updated is higher in the system without a consistency model, than in Idroid. This means that our solution spends less bandwidth when propagating this updates.

Summarizing, Idroid is a system with much better performance than a system without any consistency model. In overall, Idroid overcomes the capability of reducing bandwidth resources. Through an efficient and usable consistency model, our system is capable of heavily reduce the use of network resources.

After the analysis of the obtained values, the following conclusions can me made:

- **Idroid reduces up to 17% the amount of non-relevant information sent to the user.** Our analysis shows that the use of the consistency tubes can decrease the amount of non-relevant information in the user side. Realizing the obstacles of the real world allows Idroid to remove unaccessible,

therefore non-relevant, points of interest. This fact enables us to achieve one of the goals of this project: display only relevant information to the user.

- **Idroid is able of reducing up to 90% of the used bandwidth in comparison with a solution that does not uses a consistency model** Using the VFCdroid model, our prototype is capable of postponing non critical updates. In order to maintain the freshness of the results, the VFCdroid consistency model schedules the updates by the distance they are to the user. The points of interest that are closer, require to receive more updates than the far ones. By doing this, the system uses less bandwidth, and can present the users with more accurate information.

Chapter 6

Conclusion

The fast growing on the technologies of mobile devices leaded to an increasing development of mobile applications. Amongst them are Location Based Services and Augmented Reality applications. Furthermore, the interest in combining these two technologies has motivated the design and implementation of systems capable of providing points of interest using virtual objects. All this, enhanced the perspective of combining the real world with a virtual one. One of the main concerns in augmented reality applications is the responsiveness of the system. Allied with the inconstant mobile network signal, the need for reducing the use of network resources became a premise. This document presented Idroid, a Location Based Service capable of efficiently presenting points of interest in the user's surroundings, displaying virtual objects in the mobile device's screen. It also guarantees the freshness of the displayed information, maintaining the user updated regarding changes in the point's of interest information.

The system achieves a better efficiency and guarantees the freshness of the information by using a consistency model, where distant objects request less freshness than closer ones. We named the model as VFCdroid. This consistency model searches for the points of interests, regarding the obstacles of the real world. VFCdroid is applied over a digital map and takes into account the streets the user can pass. This way, the inaccessible points of interest and the ones that are over the radius of the user are not taken in consideration, lowering the amount of data passed through the network. By using VFCdroid, this system is also capable of preforming selective scheduling of updates based on the importance, determined through user specification. Therefore, multiple consistency degrees are applied to different points of interest, giving the system the possibility of only sending critical updates and postponing the others. These postponed updates end to be overlapped by new updates, avoiding the need of synchronizing them which saves network resources.

A prototype was built for running on mobile devices, specifically in Android platforms. The prototype consists in a multi-user client-server application that allows to respond the requests of each user. Additionally, the client side application is equipped with an interface in which users may specify their interests and then, see the points of interest drawn on the mobile device's screen.

The Idroid prototype was evaluated using several users with different interests. The results of the evaluation were then compared with a different solution, where a total consistency model was applied. From this evaluation, we concluded that Idroid reduces the amount of non relevant information sent to the user. Also, the VFCdroid model proved to bring huge benefits w.r.t. the total used network bandwidth. In our evaluation, we proved that using the VFC model against the total consistency model, the system could save up to 90% of the used bandwidth. Therefore, the system proved to use very low bandwidth compared to the other solution, which makes it suitable for Augmented Reality systems and Location Based Services.

In conclusion, Idroid is an efficient Location Based Service capable of presenting results through Augmented Reality using low bandwidth to maintain the freshness of the information regarding the points of

interest in the user's surroundings. It makes use of a consistency model that makes an intelligent schedule of updates according to its importance to each user.

6.1 Future Work

In this section we present modifications or improvements that may be done in the future:

- **Present other users.** The current implementation of Idroid only allows the presentation of points of interest. This concept could be expanded to also show the other users that are using the application. If a user is also interested in finding other people, the system could provide him with the users that are close to him.
- **Provide security mechanisms.** In Location Based Services the security concern is one of the most studied issues. These types of systems can be hacked and several of attacks can be done. For example, a user can impersonate another user. However, the most troublesome attack can be the stocking attack. A intruder can hack the system and follow another user, knowing is interests and the places where he usually is. This aspect was not in Idroid's project scope. However, it can be an additional feature. Some of the security mechanism that can be added are: authentication of each user, assure confidentiality and data integrity of the information passed between client and server, guarantee the user's privacy and introduce policies in order to manage the information passed through the network.
- **Provide directions.** The system could provide directions to the points of interest. After presenting the objects in the screen, the user could select one of them. Then, the system could guide the user to the point of interest, providing the directions he should follow. This functionality would be similar to the what the GPS devices provide.
- **Detailed search.** Instead of only searching for types of points of interest, the system could provide a more detailed search criteria. For example, a user, who is interested in restaurants, can be interested only in the ones with higher rating. This feature would decrease amount of non relevant information to the user, decreasing the usage of network resources.

Bibliography

[1] Höllerer, T.H., Feine, S.K.: Mobile augmented reality. In: Telegeoinformatics: Location-Based Computing and Services. (2004)

[2] Junglas, I.A., Watson, R.T.: Location-based services. Commun. ACM **51** (March 2008) 65–69

[3] Bellavista, P., Kupper, A., Helal, S.: Location-based services: Back to the future. IEEE_M_ PVC **7** (2008) 85–89

[4] Papagiannakis, G., Singh, G., Magnenat-Thalmann, N.: A survey of mobile and wireless technologies for augmented reality systems. Comput. Animat. Virtual Worlds **19** (February 2008) 3–22

[5] Beer, W.: Geopointer: approaching tangible augmentation of the real world. In: Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia. MoMM '10, New York, NY, USA, ACM (2010) 221–225

[6] Reitmayr, G., Schmalstieg, D.: Data management strategies for mobile augmented reality. In: International Workshop on Software Technology for Augmented Reality Systems (STARS 2003), Tokio, Japan (October 2003) 47–52

[7] Jiang, B., Yao, X.: Location based services and gis in perspective. In Gartner, G., Cartwright, W., Peterson, M.P., Cartwright, W., Gartner, G., Meng, L., Peterson, M.P., eds.: Location Based Services and TeleCartography. Lecture Notes in Geoinformation and Cartography. Springer Berlin Heidelberg (2007) 27–45 10.1007/978-3-540-36728-4_3.

[8] Cupper, A., Treu, G., Linnhoff-Popien, C.: Trax: a device-centric middleware framework for location-based services. Communications Magazine, IEEE **44**(9) (sept. 2006) 114 –120

[9] Wang, L., Qi, M., Lin, C.: User preference awareness in city traveler helper system based on naïve bayes classification. In: Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on. Volume 1. (aug. 2008) 618 –621

[10] Küpper, A., Treu, G.: Efficient proximity and separation detection among mobile targets for supporting location-based community services. SIGMOBILE Mob. Comput. Commun. Rev. **10** (July 2006) 1–12

[11] Maximilian Zündt, Girija Deo, M.N.D.M.L.: Realizing peer-to-peer location-based services in mobile networks. In: Proceedings of the 2nd workshop on positioning, navigation and communication & 1st Ultra-wideband expert talk. (aug. 2005) 175 –182

[12] Asthana, A., Cravatts, M., Krzyzanowski, P.: An indoor wireless system for personalized shopping assistance. In: Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on. (dec 1994) 69 –74

[13] Adusei, I., Kyamakya, K., Erbas, F.: Location-based services: advances and challenges. In: Electrical and Computer Engineering, 2004. Canadian Conference on. Volume 1. (may 2004) 1 – 7 Vol.1

[14] Burcea, I., Jacobsen, H.A.: L-topss - push-oriented location-based services. In Benatallah, B., Shan, M.C., eds.: Technologies for E-Services. Volume 2819 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2003) 131–142 10.1007/978-3-540-39406-8_11.

[15] Koeppel, I.: What are location services? - from a gis perspective. In: ESRI white paper. (2000)

[16] Kumar, S., Qadeer, M., Gupta, A.: Location based services using android (lbsoid). In: Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on. (dec. 2009) 1 –5

[17] Pashtan, A., Blattler, R., Andi, Heusser, A., Scheuermann, P.: Catis: A context-aware tourist information system. In: Proceedings of the 4th International Workshop of Mobile Computing, Rostock, Germany (2003)

[18] Xiaofang, Z., Jin, L., Qi, L.: Personalized information recommendation service system based on gis. In: Natural Computation, 2007. ICNC 2007. Third International Conference on. Volume 5. (aug. 2007) 805 –808

[19] Azuma, R.T.: The challenge of making augmented reality work outdoors. In: In Mixed Reality: Merging Real and Virtual, Springer-Verlag (1999) 379–390

[20] Dubois, E., Nigay, L., Troccaz, J.: Consistency in augmented reality systems. In Little, M., Nigay, L., eds.: Engineering for Human-Computer Interaction. Volume 2254 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2001) 111–122 10.1007/3-540-45348-2_13.

[21] Balan, R., Ebling, M., Castro, P., Misra, A.: Matrix: Adaptive middleware for distributed multiplayer games. In Alonso, G., ed.: Middleware 2005. Volume 3790 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 390–400 10.1007/11587552_20.

[22] Veiga, L., Negrão, A., Santos, N., Ferreira, P.: Unifying divergence bounding and locality awareness in replicated systems with vector-field consistency. Journal of Internet Services and Applications **1** (2010) 95–115 10.1007/s13174-010-0011-x.

[23] Santos, N., Veiga, L., Ferreira, P.: Vector-field consistency for ad-hoc gaming. In: Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware. Middleware '07, New York, NY, USA, Springer-Verlag New York, Inc. (2007) 80–100

[24] Feiner, S., MacIntyre, B., Höllerer, T., Webster, A.: A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. Personal Technologies **1** (1997) 208–217 10.1007/BF01682023.

[25] Reitmayr, G., Schmalstieg, D.: Collaborative augmented reality for outdoor navigation and information browsing. Science **66** (2004) 31–41

[26] Morrison, A., Oulasvirta, A., Peltonen, P., Lemmela, S., Jacucci, G., Reitmayr, G., Näsänen, J., Juustila, A.: Like bees around the hive: a comparative study of a mobile augmented reality map. In: Proceedings of the 27th international conference on Human factors in computing systems. CHI '09, New York, NY, USA, ACM (2009) 1889–1898

[27] Takacs, G., Chandrasekhar, V., Gelfand, N., Xiong, Y., Chen, W.C., Bismpigiannis, T., Grzeszczuk, R., Pulli, K., Girod, B.: Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In: Proceedings of the 1st ACM international conference on Multimedia information retrieval. MIR '08, New York, NY, USA, ACM (2008) 427–434

[28] Kähäri, M., Murphy, D.: Mara - sensor based augmented reality system for mobile imaging. In: Proceedings of the Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR06). (October 2006)

[29] Arusoaie, A., Cristei, A., Chircu, C., Livadariu, M., Manea, V., Iftene, A.: Augmented reality. In: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2010 12th International Symposium on. (sept. 2010) 502 –509

[30] Marimon, D., Sarasua, C., Carrasco, P., Álvarez, R., Montesa, J., Adamek, T., Romero, I., Ortega, M., Gascó, P.: Mobiar: Tourist experiences through mobile augmented reality. In: Proceedings of 2010 NEM Summit, Barcelona (October 2010)

Appendix A

Questionnaire

Questionário para avaliação de aplicações móveis

Ano Lectivo 2011/2012

Introdução:

Este questionário tem como objectivo avaliar duas aplicações móveis. Esta aplicações são denominadas Serviços Baseados em Localização. Têm como objectivo apresentar pontos de interesse em redor do utilizador. Em particular, estas duas aplicações, apresentam os seus resultados desenhando objectos virtuais sobre a vista que a câmera do dispositivo móvel. As aplicações a serem usadas têm como nome Layar e Idroid.

Ao utilizador serão dadas as duas aplicações e um conjunto de tarefas que terá de realizar. De seguida preencherá este questionário.

Dados do utilizador:

Sexo: M ☐ F ☐

Nível de Ensino:

Sem ensino ☐

Ensino Básico ☐

Ensino Secundário ☐

Frequência do Ensino Superior ☐

Licenciado ☐

Idade:

< 21 ☐

21 – 25 ☐

26 – 35 ☐

36 – 45 ☐

46 – 55 ☐

56 – 65 ☐

> 65 ☐

Com que frequência interage com Smartphones ou Tablets?

Nunca utilizei ☐

Raramente ☐

Ocasionalmente ☐

Diariamente ☐

Exaustivamente ☐

Possui algum Smartphone ou Tablet?

Sim ☐

Não ☐

Alguma vez utilizou um serviço para encontrar pontos de interesse?

Sim ☐

Não ☐

Questionário

À medida que realiza as tarefas, responda ao questionário em baixo. Responda às questões marcando com um X a resposta numérica onde 0 significa "Discordo Totalmente" e 5 significa "Concordo Totalmente".

Tarefa 1:

"Após iniciar a aplicação realize uma procura onde possa encontrar restaurantes a uma distância máxima de 500 metros."

Idroid	0	1	2	3	4	5
A aplicação não apresentou problemas de funcionamento?						
O tempo de espera pelos resultados é aceitável?						
Os resultados apresentados são graficamente apelativos?						

Layar	0	1	2	3	4	5
A aplicação não apresentou problemas de funcionamento?						
O tempo de espera pelos resultados é aceitável?						
Os resultados apresentados são graficamente apelativos?						

Tarefa 2:

"Com os mesmos resultados apresentados, escolha um ponto de interesse próximo de si. Verifique a distância indicada pela aplicação até esse ponto de interesse. Agora, com a aplicação ligada, dirija-se até ao local."

Idroid	0	1	2	3	4	5
A aplicação foi actualizando a distância ao local?						
A aplicação apresentou-me novos pontos de interesse?						
A distância apresentada foi similar à distância que percorri?						

Layar	0	1	2	3	4	5
A aplicação foi actualizando a distância ao local?						
A aplicação apresentou-me novos pontos de interesse?						
A distância apresentada foi similar à distância que percorri?						

Tarefa 3:

"Volte ao local onde começou este teste. Dos resultados apresentados, escolha o ponto de interesse com maior distância. Verifique o valor apresentado. Agora, dirija-se até ao seu local."

Idroid	0	1	2	3	4	5
A distância apresentada foi similar à distância que percorri?						
A distância que percorri foi inferior ou igual ao raio de procura?						
Este ponto de interesse deveria estar nos resultados apresentados?						

Layar	0	1	2	3	4	5
A distância apresentada foi similar à distância que percorri?						
A distância que percorri foi inferior ou igual ao raio de procura?						
Este ponto de interesse deveria estar nos resultados apresentados?						

O questionário terminou!

Muito Obrigado

Appendix B

Questionnaire Results

Resultados dos Questionários

Dados do Utilizador

Grau de Ensino	
Sem ensino	0
Ensino Básico	1
Ensino Secundário	2
Frequência de Ensino Superior	11
Licenciado	12

Idade	
< 21	3
21 - 25	16
26 - 35	5
36 - 45	0
46 - 55	2
56 - 65	0
> 65	0

Alguma vez utilizou um serviço para encontrar pontos de interesse?	
Sim	1
Não	25

Com que frequência interage com Smartphones ou Tablets?	
Nunca utilizei	0
Raramente	6
Ocasionalmente	10
Diariamente	5
Exaustivamente	4

Sexo	
Masculino	12
Feminino	14

Possui um smartphone ou um tablet?	
Sim	11
Não	14

Tarefas

Tarefa 1

A aplicação não apresentou problemas de funcionamento?

	Idroid	Layar
0	0	0
1	0	0
2	0	0
3	0	0
4	10	7
5	16	19
Total	120	123

O tempo de espera pelos resultados é aceitável?

	Idroid	Layar
0	0	0
1	0	0
2	0	0
3	4	6
4	12	13
5	10	7
Total	110	105

Os resultados apresentados são graficamente apelativos?

	Idroid	Layar
0	0	0
1	0	0
2	3	2
3	13	15
4	9	7
5	1	2
Total	86	87

Tarefa 2

A aplicação foi actualizando a distância ao local?

	Idroid	Layar
0	0	3
1	2	5
2	2	7
3	10	9
4	11	2
5	1	0
Total	85	54

A aplicação apresentou-me novos pontos de interesse?

	Idroid	Layar
0	0	2
1	3	4
2	5	10
3	9	7
4	8	3
5	1	0
Total	77	57

A distância apresentada foi similar à distância que percorri?

	Idroid	Layar
0	0	0
1	1	5
2	2	6
3	9	11
4	11	3
5	3	1
Total	91	67

Tarefa 3

A distância apresentada foi similar à distância que percorri?

	Idroid	Layar
0	0	0
1	1	5
2	3	8
3	15	10
4	7	3
5	0	0
Total	80	63

A distância que percorri foi inferior ou igual ao raio de procura?

	Idroid	Layar
0	0	0
1	0	1
2	1	9
3	13	11
4	9	4
5	3	1
Total	92	73

Este ponto de interesse deveria estar nos resultados apresentados?

	Idroid	Layar
0	0	0
1	0	2
2	0	8
3	10	11
4	13	4
5	3	1
Total	97	72

Appendix C

Quantitative Evaluation Results

Resultados do Carregamento de Pontos de Interesse

Estudantes (tamanho dos dados KB)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	71,339	77,593	8,06%
20	356,695	387,965	8,06%
40	713,39	775,93	8,06%
200	3566,95	3879,65	8,06%
400	7133,9	7759,3	8,06%
800	14267,8	15518,6	8,06%
2000	35669,5	38796,5	8,06%
4000	71339	77593	8,06%

Estudantes (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	325	360	9,72%
20	1625	1800	9,72%
40	3250	3600	9,72%
200	16250	18000	9,72%
400	32500	36000	9,72%
800	65000	72000	9,72%
2000	162500	180000	9,72%
4000	325000	360000	9,72%

Trabalhadores (tamanho dos dados KB)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	264,391	286	7,56%
20	1321,955	1430	7,56%
40	2643,91	2860	7,56%
200	13219,55	14300	7,56%
400	26439,1	28600	7,56%
800	52878,2	57200	7,56%
2000	132195,5	143000	7,56%
4000	264391	286000	7,56%

Trabalhadores (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	1209	1326	8,82%
20	6045	6630	8,82%
40	12090	13260	8,82%
200	60450	66300	8,82%
400	120900	132600	8,82%
800	241800	265200	8,82%
2000	604500	663000	8,82%
4000	1209000	1326000	8,82%

Turistas (tamanho dos dados KB)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	196,135	228,346	14,11%
20	980,675	1141,73	14,11%
40	1961,35	2283,46	14,11%
200	9806,75	11417,3	14,11%
400	19613,5	22834,6	14,11%
800	39227	45669,2	14,11%
2000	98067,5	114173	14,11%
4000	196135	228346	14,11%

Turistas (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
4	873	1048	16,70%
20	4365	5240	16,70%
40	8730	10480	16,70%
200	43650	52400	16,70%
400	87300	104800	16,70%
800	174600	209600	16,70%
2000	436500	524000	16,70%
4000	873000	1048000	16,70%

Resultados das Actualizações de GPS

Carro

Estudantes (tamanho dos dados)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	85,89	266,805	67,81%
10	429,45	1334,025	67,81%
20	858,9	2668,05	67,81%
100	4294,5	13340,25	67,81%
200	8589	26680,5	67,81%
400	17178	53361	67,81%
1000	42945	133402,5	67,81%
2000	85890	266805	67,81%

Estudantes (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	217	1160	81,29%
10	1085	5800	81,29%
20	2170	11600	81,29%
100	10850	58000	81,29%
200	21700	116000	81,29%
400	43400	232000	81,29%
1000	108500	580000	81,29%
2000	217000	1160000	81,29%

Trabalhadores (tamanho dos dados)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	419,838	876,095	52,08%
10	2099,19	4380,475	52,08%
20	4198,38	8760,95	52,08%
100	20991,9	43804,75	52,08%
200	41983,8	87609,5	52,08%
400	83967,6	175219	52,08%
1000	209919	438047,5	52,08%
2000	419838	876095	52,08%

Trabalhadores (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	1314	3800	65,42%
10	6570	19000	65,42%
20	13140	38000	65,42%
100	65700	190000	65,42%
200	131400	380000	65,42%
400	262800	760000	65,42%
1000	657000	1900000	65,42%
2000	1314000	3800000	65,42%

Turistas (tamanho dos dados)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	255,796	691,165	62,99%
10	1278,98	3455,825	62,99%
20	2557,96	6911,65	62,99%
100	12789,8	34558,25	62,99%
200	25579,6	69116,5	62,99%
400	51159,2	138233	62,99%
1000	127898	345582,5	62,99%
2000	255796	691165	62,99%

Turistas (nº de poi)

Nº de Clientes	VFCdroid	sem VFCdroid	Gain %
2	618	2970	79,19%
10	3090	14850	79,19%
20	6180	29700	79,19%
100	30900	148500	79,19%
200	61800	297000	79,19%
400	123600	594000	79,19%
1000	309000	1485000	79,19%
2000	618000	2970000	79,19%

Resultados das Actualizações da Base de Dados

Carro

Estudantes (tamanho dos dados)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	0	1121	100,00%
10	4406	4981	11,54%
20	6622	7787	14,96%
40	7406	13167	43,75%
60	8904	21377	58,35%
80	13374	29471	54,62%
100	15237	38149	60,06%
126	15833	48721	67,50%

Estudantes (nº de poi)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	0	2000	100,00%
10	17000	20000	15,00%
20	28000	34000	17,65%
40	32000	60000	46,67%
60	39000	100000	61,00%
80	61000	140000	56,43%
100	70000	180000	61,11%
126	73000	232000	68,53%

Trabalhadores (tamanho dos dados)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	0	1121	100,00%
10	561	6941	91,92%
50	1132	17159	93,40%
100	6764	18429	63,30%
150	9403	39873	76,42%
200	27980	56961	50,88%
300	38775	98593	60,67%
437	63391	160019	60,39%

Trabalhadores(nº de poi)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	0	2000	100,00%
10	1000	30000	96,67%
50	2000	78000	97,44%
100	29000	84000	65,48%
150	41000	186000	77,96%
200	131000	268000	51,12%
300	183000	466000	60,73%
437	300000	760000	60,53%

Turistas (tamanho dos dados)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	604	1209	50,04%
10	1209	5099	76,29%
50	1424	11749	87,88%
100	4079	28413	85,64%
150	6461	50375	87,17%
200	21168	71963	70,58%
250	42097	93033	54,75%
330	42720	126353	66,19%

Turistas (nº de poi)

Nº de Updates	VFCdroid	sem VFCdroid	Gain %
1	1000	2000	50,00%
10	2000	20000	90,00%
50	3000	52000	94,23%
100	16000	130000	87,69%
150	27000	234000	88,46%
200	96000	334000	71,26%
250	194000	434000	55,30%
330	198000	594000	66,67%