A LOCATION-BASED MOBILE APPLICATION FOR THE GREATER MANCHESTER POLICE ON THE BEAT

A dissertation submitted to The University of Manchester for the degree of Master of Science in the Faculty of Engineering and Physical Sciences.

2013

RAFAEL I. LARIOS RESTREPO

SCHOOL OF COMPUTER SCIENCE

List of Contents

Final word count: 30791

List of Figures

List of Tables

Abstract

The Greater Manchester Police (GMP) has a wide variety of information about victims, suspects, crimes, among others, resulting from their day to day activities. That information is correlated with the location where it was generated. As of today, this location-specific information is not available to the police officer that is patrolling the streets.

To introduce a system that can give up to date information, relevant to the current location of the police officers that use it, can improve the efficiency and effectiveness of the activities performed by them.

The aim of this project is to develop a mobile application that uses location based data provided by the GMP, to display on a mobile device, using user friendly interfaces and using the device's sensors and features; this enhances the information received by the officer on the beat. To achieve this goal, an analysis of the needs and requirements of the GMP is made. Also, a review of current approaches for location based applications for a variety of entities is constructed.

According to the needs of the GMP, an application with several interactive interfaces has been designed and implemented. Technologies like GPS, Map interfaces and Augmented Reality displays are described, designed and used on the implemented application. A series of technical and functional evaluations have also been performed to test the system functionalities and performance on several environments and scenarios. These evaluations reflect a successful implementation of the selected technologies and the user's requirements. With this information a conclusion of the work is presented including guidelines for further work which can improve and enhance the application in the future.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the dissertation, for example graphs and tables (“Reproductions”), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/display.aspx?DocID=487), in any relevant Dissertation restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s Guidance for the Presentation of Dissertations.

1. Introduction

In recent years, mobile technology has permeated all aspects of human interaction with the world. This interaction ranges from consumer applications, including web-browsing, social media and games, to work-related applications, including financial and systems monitoring, and a wide range of instant communication. The availability of the internet on any location, thanks to data plans on mobile devices, from laptops, tablets and smart phones, has given all consumers the possibility to reach any information they desire from any location.

With the increase of these devices, new ways to increment the productivity and efficiency of workers in all fields has been introduced [1]. From simple text messaging communication (chat, e-mail) to applications catered specifically to a given field of work. These include project scheduling software, remote monitoring applications, and Enterprise Resource Management (ERP) systems among others.

1.1 Current needs from the Greater Manchester Police

The Greater Manchester Police (GMP) is looking for innovative ways to implement technological solutions involving mobile technologies. These solutions aim to increase the efficiency of the work done by police officers during patrol. They are also aimed to complement their current technological infrastructure, and improve the efficiency of their activities on a day to day basis, without incurring in a considerable investment.

GMP expects its officers to use a technological solution to complement their activities, confident that the availability of current, detailed and precise information will increase their effectiveness. The information queried by police officers regarding victims, suspects, arrests, properties, etc, needs to be marked with the precise location of occurrence, in order to be shown to the user in the location where the user is located at any time. This information is managed by internal information systems within GMP; however there is not an actual system that allows its querying on real time by police officers on the beat. It is expected that a mobile application that uses readily available services and features and low cost of acquisition and maintenance, could help achieve the above needs.

1.2 Aims and objectives

This project will develop a mobile solution for the GMP that allows their police officers patrolling the streets, to query and receive relevant information based on their current location. This information contains data regarding suspects, victims, properties, etc and their specific geographical position. To allow a greater functionality, the application must capture images taken from the device's camera, and enhance them with information queried from the GMP's database.

1.3 Dissertation Outline

To accomplish the project aims and objectives is important to know the current environment and tools that are being used by the police forces. This report contains a study that will help understand the police needs and constraints regarding their field of work.

With that in mind, a review of the mobile technology evolution is made including its characteristics and main functions. It will serve as a starting point to understand the main focus of the dissertation. In the same vein, a review of Augmented Reality (AR) technologies will take place. These technologies, serve to enhance the interaction of the user with the world by embedding computer generated images into the user interface.

A history of the technologies available and its characteristics, and describing the technologies available for implementation is made. This will present a frame of reference that will help shape the final implementation of the mobile application required by the GMP needs.

Once the background review is made, this report will focus on the processes and methodologies that are being followed to develop the mobile application. It describes the software development cycle followed in the implementation, including a description of the tools required for its evaluation.

Further, the report describes all the design decisions made to develop the application prototype, including an overview of the meetings that have taken place with the GMP and the list of requirements gathered from them. A section covering the process of

analysis and selection of the development technologies is also included. This section includes all the design diagrams and features defined for the implementation.

The implementation chapter shows all the steps incurred to develop each feature as described in the design chapter. It is focused in the most interesting aspects of the application implementation, including the different modes and features developed for it. In some cases, diagrams and excerpts of application code are displayed and explained further.

The next chapter shows all the methods and processes defined for the project evaluation and testing of the application prototype. It defines how the application has been tested during and after development and includes technical and functional perspectives..

Finally, the dissertation outlines a summary of the work that has been executed, conclusions over its main results and provides guidelines to improve it further. These guidelines include in detail the main aspects that can be used to enhance the system features and technical capabilities in order to increase the system appeal and usefulness in the future.

2. Background

2.1 Police environment

Government entities around the world are looking for solutions to increase their effectiveness while reducing their costs described by Hauner and Kyobe [1]. During the last economic crisis, the budget of government in the UK has been reduced [2], and it has affected all its departments. While it seems that recession continues to affect budgets worldwide; innovative solutions have been looked upon to tackle this environment.

While is the case that private companies are leading the way in terms of technological innovations, it has been seen that government entities are using the same incentives to improve their bottom line [3]. An increase in productiveness and effectiveness for every area of public service is needed to overcome their budget cuts.

Among the departments affected by budget reduction are the police departments in the UK. According to the Her Majesty's Inspectorate of Constabulary (HMIC) [4], the organism in charge to inspect the police forces, budget cuts have affected the number of police men and women patrolling the streets. It has been shown by McCarty and Ren [5] that the overall security of an area is diminished with the reduction of police force, directly affecting the citizens located in the area.

A police officer has a wide variety of duties, ranging from patrolling the streets in case of suspect behaviour to administrative duties to support the policing organization. Other duties concern arresting individuals who have broken the law. While performing said duties, a police officer can conduct investigations, keep records of his/her activities, provide first aid in case of emergency, and testify in court regarding a specific crime. It is expected that a police officer must enforce the letter of the law in an ethical manner while carrying out the orders given by their supervisor [6].

A police officer in the UK can belong to a given number of departments. The law enforcement departments in the UK are diversified according to their function [7]:

- **Territorial Police Force**: Carry out most police duties in the UK. Their duty is to enforce the law, provide security services for all residents of a specific area;

prevent crime and damages to property and to improve the quality of life for all. The police forces associated with this description are deployed within England, Wales, Scotland and Northern Ireland.

- **Special Police Forces**: These are national forces that do not have a specific area or jurisdiction. They are in charge of specific duties, like policing regular and rail transport, non military nuclear facilities, and drug enforcement, among others.
- **Miscellaneous Police Forces**: Have the responsibility to police certain specific areas, like ports or parks.

In order to perform their duties, police officers have several tools at their disposition. These tools can include cuffs, extendable batons, and incapacitating devices like tasers and incapacitant sprays. Most of the police officers do not carry firearms, but there are some officers authorized to carry them during special duties and on special patrols [8]. while most of the tools have suffered some changes during their usage in the police duties, this changes seem to be rather evolutionary than revolutionary in nature, and most of the procedures and day to day activities have been the same during this time.

While the information age has permeated all structures of private and public industries, helping data analysis, information retrieval and productivity, most police officers during their regular patrol still rely in the same tools of communication and law enforcement as their peers 20 years ago as specified by UK Legislature [7] [8].

It is expected that with a reduction of the number of police officers patrolling the streets, each officer must cover a wider range of area reducing their effectiveness and increasing the probability of criminal activity. To overcome this situation, it is necessary to envision new tools that can be developed and given to each individual officer so he can maintain, or even increase the productivity and efficiency of each patrol duty.

In order to create such tool, it is imperative to know in detail the regular activities of a police officer on the beat. With such description it will be possible to find the most pressing needs of the police department in this area and design a tool that can be used effectively by the officers.

The main activities done by a police officer encompass the following as described by [9]:

- Search of individuals or vehicles for suspect activities and for keeping record of said searches.
- Arrest, detention and treatment of individuals during criminal activity. This activity can be done with or without a warrant depending on the situation.
- Stop vehicles and review drivers' documents like drivers licence and insurance. Request a breath test if the driver seems to have been drinking or the driving pattern seems erratic.
- Search premises and seize property found on them.

During the execution of these activities most police officers patrol the streets on foot. In order to effectively execute those activities, it is necessary for them to hold the most relevant information related to their current location. If the police officer has enough current information regarding individuals and premises, it will be possible for the police officer to prioritize the most pressing actions at a given time and location.

While location based technologies have been introduced to the police force in recent years [10], they have mostly been introduced in police vehicles used during patrol (Cars, Helicopters). Assistance for police officers on the beat, offering functionality related to the current location has not been implemented on a worldwide basis.

A device that can recognise a police officer's location, and offer information regarding updated relevant information regarding criminal offences, suspects, vulnerable individuals, etc. could help officers to engage in their activities with a greater degree of accuracy, saving time and effort for each member and increasing security for the city population. The ability to query other information regarding nearby locations is also needed to increase the efficiency of the police on the beat.

With the continuing advances in mobile technology and the affordability of these devices for the regular population, it would be possible to implement a prototype to display relevant information for the officer on a simple display without a huge increase in costs for the government.

2.2 Evolution of Technology on Mobile devices

It is possible to categorize the handheld transceiver (personal radio transceiver or "walkie-talkie") as the first mobile device used for communication [11]. The possibility 2 persons to communicate over a large area increased the possibility to coordinate and implement actions without visual contact. Its development dates from the Second World War using batteries and vacuum tubes which made the devices a considerable size. After the war and with the introduction of the transistor, the handheld transceiver reduced its size and was introduced safely and as a consumer device for recreation or other activities.

During the 50's new ways for communication were introduced with the pager. The pager was introduced as a device to receive phone messages from a limited range, and it was used first by physicians in New York on a subscription basis as stated in [12]. The technology of the pager also evolved to acknowledge, and replay received messages. Although by the mid 90's the availability of cellular networks displaced the popularity of the pager, this technology is still used in locations were the simplicity, and cost of the device still offer an advantage over more advanced technologies, as seen in [13] [14].

Recently, the technology of mobile phones has also evolved. The first mobile calls were performed on a device placed on a car in St Luis, USA during the first half of the 40's decade [15]. By the 70's mobile phone technology started to grain ground based on the advances made by Motorola Research Laboratories [16] [17], with the first commercially available mobile phone sold in the first half of the decade. While the phone had severe disadvantages (high price, size, weight battery duration and charging times), it had a very good consumer reception for the time, with orders numbering in the thousands [18].

During the 80's the technology started to be commercialized in the rest of the world. Countries like Japan and the Nordic countries increased their investment in infrastructure to support the cell networks needed to support mobile phones [19]. During this time new features were introduced by the phone manufacturers to increase consumer appeal. Technologies to reduce the batteries' size (reducing the overall phone size) and the introduction of text messages by the short message service

(SMS) were counted among these advancements. By the middle of the 1990s, mobile phones were starting to gain ground in the common population. The call and device prices, while still high, started to be accessible by the majority of the people [20]. With the introduction of higher processing power and features like games and applications, mobile phone stopped being a luxury item and started to become tools for efficient communication and collaboration among their users.

By the 2003, hardware resources where advanced enough to start increasing additional features on phones. The launch of devices like Blackberry started the era of smartphones, which increased mobile phone features, by adding E-mail messaging, internet browsing, and a greater number of business and consumer applications. By the first half of the decade, it was possible for consumers to acquire smart phone devices with an extend array of sensors and communication technologies (Bluetooth, Wi-Fi) and high quality touch screens. By this time, advances in new Operating Systems (OS) like iOS and Android, made it possible for the general population to start creating applications, taking advantages of the phone's capabilities, and to commercialize them through vendor-supported marketplaces.

2.3 Overview of available technologies for Implementation

On a current hand held device, like a high end smart phone or tablet, it is possible to find a variety of sensors and technologies that can help the development of a wide range of applications. Global Positioning System (GPS) sensors, network access, gravity and light sensors, accelerometers, gyroscopes, orientation and rotation vectors, high definition screens among others, can be found on most devices. These are the most relevant systems available to create a rich user experience on a mobile display:

- **Global Positioning System (GPS):** System that is composed of an array of over 30 satellites in low earth orbit that provide signals that can be can be detected by a receiver on the earth's surface [21]. The system allows any device that has a GPS sensor, and can receive a clear signal from at least 4 satellites, to determine an accurate location of the user, in real time, all over the world. With the use of the satellites' signal, the receiver can triangulate its location accurately. The system uses the standard coordinate system (longitude and latitude) that can be used on any application or physical map. The ability of any

GPS device to infer the user's location, allows any software to provide any information relevant to it. The system is developed and administered by the Department of Defence of the United States of America.

- **Touch Screens**: Current hand held devices have displays that can be used to interact with the user by haptic sensitive screens. Screens that can perceive the human touch and capacitive devices like stylus pens enable the users to interact with the information displayed in them [22]. These types of screens can be attached to a wide array of devices, ranging from computers and video game consoles to smart phones, helping the development of a great set of consumer and business applications [23]. Touch screens use different technologies, include capacitive sensing (senses electrical conductivity elements like human fingers and conductive pens) [24], resistive sensing (layered conductive surfaces that react to the position of the interaction) [25] and acoustic recognition (sound wave sensors that digitalize sounds produced by screen touch) [26] are some of the available technologies used to implement interactive screens on current devices.
- **Digital Cameras:** Devices that can sense light sources to produce photographs or video on a digital apparatus using an electronic sensor [27]. These cameras provide the user with a digital view of the world around the person. Using several methods, like single-shot, multi-shot and image scanning, the light sources can be manipulated to display images to the user using different resolutions and filters [28]. While most digital cameras started as single devices, today the largest number of digital cameras can be found in mobile phones [29], thanks to the advancements of electrical manipulation and miniaturization, and improvement in lens technologies.
- **Network Access:** Most hand held devices today have access to a data connection in most places around the world. Hand held devices can use several technologies to interchange data. This extended connectivity can be obtained using Bluetooth, GMS, GPRS, EDGE and LTE networks, Wi-Fi (802.11 IEEE standards), among others. With this capability, any modern hand held device and software can retrieve and upload information from anywhere as long it has a connection available. A data connection improves the amount of interaction that a user can make with the applications and their surrounding environment,

due to the possibility to generate and display dynamically any information and to connect to any service reachable on a data connection.

- **Accelerometer:** An accelerometer is a sensor that can measure the acceleration along a given axis on any device that holds it. It is based on the gravity experienced by the weight of a test mass over a point of reference as described in [30]. The accelerometer can use piezoelectric elements (which use electrical charge generated by mechanical stress) and capacitive elements (which use electrical capacitance) to convert the mechanical force produced by gravity into electrical signals that can be read on a device, and perform calculations based on them. Using the accelerator provides a way to measure the direction and pitch of a device, having gravity as a point of reference. Using this sensor, an application can calculate the approximate orientation of a device and display information accordingly.
- **Compass:** The compass is an instrument that can measure the earth's magnetic fields surrounding it, to display the direction and orientation within a frame of reference. Compasses on mobile devices help software to use those measurements to provide and display to the user, the direction of the device within the cardinal points of a location and to display it to the user. The compass on a mobile device is implemented via a magnetometer [31], which measures the strength and direction of the magnetic field.
- **Gyroscope:** A device to measure the orientation of the device. A gyroscope uses angular momentum to calculate the rotation and pitch. Several applications can use gyroscopes to provide accurate information to the user, and increase the precision when used with other sensors like accelerometers [32]. Among these applications it is possible to find video gaming, naval navigation, and stabilization of flying vehicles.

Using these available technologies it is possible to create a compelling tool for almost any requirement from a consumer or business application that needs robust location and orientation capabilities. Among the applications that can take advantage of these sensors and tools, are applications that provide a direct view of the world and enhance their view with computer generated information. These kinds of applications are described as Augmented Reality applications.

2.4 Augmented Reality overview

Augmented Reality (AR) as described in [33], is the inclusion of computer generated information and widgets over a display of real world images. In this context, the real world view is "augmented" by all the information generated by the computer interface between the view and the user. The information displayed the computer interface can be generated from a variety of inputs, based on the type of application and the device used.

There is a wide range of displays that can provide an AR perspective to the user [34] [35] [36] [37]. Here is a description for some of these devices:

- **Head mounted displays**: Special helmets located on the user's head that provide an information display to the user. These devices have special sensors that provide a wide range of degrees of freedom and adjust the information according to head movements. An example of this technology can be seen in Air force pilot's helmets.
- **Eyeglasses:** An evolution from the head mounted display; eyeglasses can provide an augmented display of the real world, using small camera and a range of sensors, displayed over the lenses of the screen. Lately, Google has been working on the development of this technology in the Project Glass.
- **Handheld devices:** The evolution of smart phones and tablets has allowed the inclusion of different sensors, including cameras, GPS, accelerometers, etc. With the increase of processing power of handheld devices, a wide set of applications is being developed. Companies like Layar have surfaced and they develop applications and services that take advantage of these technologies [38].
- **Spatial Reality:** Instead of displaying information on specific screens, spatial reality devices project the desired information over physical objects. This separates the display from the user, allowing a wider range of users to collaborate on the same space. One of the drawbacks of this type of device is that since it needs a surface to project the desired information, so it cannot offer the same flexibility that user specific devices provide.

- **Next Generation Displays**: The latest generation of displays is being researched around the world, and are based on the direct display of information over user's retina or contact lenses. Other types of displays are based on the real time processing of light sources to provide an augmented view of a light source that the human eye cannot physically see, for example a clear view of a soldering process that cannot be seen directly by the human eye [39].

AR displays and software can already be found in several fields, from consumer to business applications. Some of the uses for this technology can be described as task support (allowing a performance increase in the execution of a complex task), to educational and recreational. A description of these applications and range of fields is followed in more detail:

- **Task Support**: applications that help the execution of a given task can be found on several fields [40] [41] [42] [43]. Using AR, an architect can help visualize information about construction projects while they are being built, and a consumer can view extended information of a product that the person may want to buy inside a store. AR displays can also be incorporated on assembly lines to improve production line processes. In a military perspective, extended information on the battlefield can help soldiers to assess risk regarding their environment and enable real time communication between the soldiers and the command centre.
- **Educational**: AR can help extend the interaction between the user and the subjects of study as noted by [44] [45] [46]. Textbooks and other related educational material can be scanned using AR displays to increase the information available to that object. Field trips can be enhanced by the use of this technology, by means of increasing the information available on a remote site, and providing simulations of past events on the site.
- **Recreational**: Tourism applications that present interactive information on important sites on different cities can be displayed using these devices [47] [48]. Fans of sports can improve their viewing experience by looking computerized images that extend information on the field of play, examples of these can be found on the "first down" line on American Football transmissions

[49], where the TV viewers can see increased information regarding the playing field that is not available for the people at the stadium.

Overall, any task that can be performed using an appropriate display and can be enhanced by availability of increased information regarding said task, is suitable for an AR implementation.

2.5 Location based and AR applications for consumer and business entities

With the evolution of mobile technologies, software development has taken advantage of the capabilities offered by those technologies. Application marketplaces have provided a simple way for both consumer and business users to download and use applications, and for developers to distribute them. To give an example, since the introduction of the app store by apple in 2007, more than 800,000 applications are available to download. Their success has been demonstrated by more than 40 billion downloads in the same time frame [50]. This mobile marketplace has led the way for other mobile vendors like Google and Microsoft to create their own platforms for content distribution on their own mobile environments.

The availability of marketplaces and the increasing resources and technologies offered by mobile devices has produced a great number of applications that have been created to cater the needs of a wide range of consumers. One subset of those applications is geared towards location based technologies, in which their use can enrich user interaction and provide reliable location based information for users. Likewise, the increase of the devices' capabilities to sense and measure their surroundings, has enabled the development of AR applications on a wide range of platforms.

From the consumer standpoint, applications like foursquare, let users check in into businesses and venues based on a GPS based location, are available. On the same social media topic, mobile apps for facebook and twitter can access the GPS sensors to tag user postings and create a map of the user locations for the user's friends to see.

Outside the social media environment, consumer applications that enhance the user experience can be found, including applications that let the user decide if a product will look good in his house before buying it [51]. Other consumer applications are

based on displaying finished views of boxed products, like Lego's augmented reality app [52]. Tourism has also been a good source for developing applications, where museums and government entities use mobile applications to display enhanced information on historical landmarks or to display an object in different periods of time.

From a business perspective, several entities have created applications that take advantage of the mobile technologies to create an enhanced user experience [53]. Fleet and sales management apps are being created to keep detailed locations of sales and fleets across a wide area. Other areas in which these technologies are applied are based on asset tracking, work validation and location based search.

The usage of location based applications has been in use on the military since the beginning, and in the case of AR technologies, it was first introduced by military research [54]. However, given the evolution and cost of the technologies, its uses have been restricted to few applications. Research has been done on mobile applications for military use based on detached devices like smart phones that provide a better presentation interface [41].

Regarding the subject of analysis, the GMP has provided an Application Programming Interface (API) that provides location based information for any application that wants to use it [55]. This data contains information about crime statistics at street and neighbourhood level, nearest police stations, among others. There are several applications that use this data in creative ways [56], alerting the users to rough neighbourhoods, and the possibility of locating specific crime in an area. The drawbacks of this public API is that the location information is not as accurate as the one used internally by the GMP and the information may not be up to date.

3. Methodology

3.1 Development Process

The project aims to analyze, design and implement an application prototype that meets the needs and requirements of the Greater Manchester Police (GMP) officers on the beat. In order to accomplish this objective it is imperative to define a process in which these requirements will be gathered and studied. This project will follow an iterative process of software development engineering as stated by Larman [57], in which the main phases of the process are described as follows:

- **Inception**: A small study is made to gather the most outstanding requirements from the client (in this case the GMP). This phase is made with the in collaboration with the GMP contact, the project's supervisor and the student.
- **Elaboration**: The main objectives are laid out and the most pressing needs are stated to give start to the phase. This phase is iterative, with requirements identified and addressed in each iteration. On this project, the client will not be able to accompany personally during all the iterations, but constant communication with the client is expected throughout the duration of it.
- **Construction**: This phase encompasses coding and testing of the requirements defined for each iteration. During an iteration, the construction phase has the majority of the effort. It is possible that during the course of each phase, the original requirements change, and those changes will be addressed on the following phase.
- **Transition**: Results from the construction phase are analyzed and deployed on the working prototype. Feedback from the client is not expected on each iteration. This phase will be executed by the project supervisor and the student. Once a transition phase is finalized a new elaboration phase begins.

It is expected that a prototype will be constructed during several iterations of the software engineering process. Throughout each cycle or iteration, several artefacts (deliverables of the process) will be constructed. The artefacts that will be delivered by each development cycle are:

- **Design diagrams:** These show the overall design of the applications, they could include the different abstraction layers of the implementation. This set of diagrams can specify static or dynamic processes of the application. They are intended to give a "bird's eye" view of the system structure and function.
- **Class Diagrams:** They display the model of the data manipulated by the application. They show the domain model and the class structure for the system. In these diagrams, attributes and operations of each class are shown. It is expected that the model is augmented and refined in each cycle.
- **Vision Document:** This single document specifies the overall goals and objectives of the resulting application. It defines the limits and scope of the effort to be executed on the project. This document is constructed in the initial phase of the project (Inception). In this dissertation the vision document is represented in the Design chapter.
- **Bugtracker:** This document defines all the test cases, and keeps track of the results of each test. It allows monitoring of the real progress of a cycle. The bugtracker specifies all the tests to be made on the application, including functional and non-functional requirements, integration and acceptance tests to be made by the client. The bugtracker is displayed on the chapter 6 (Evaluation) of the dissertation.

3.2 Design Patterns

In order to provide an acceptable result the application should be designed to conform to common design patterns in the code and user interface. As stated by the "gang of four" [58], software design patterns are useful to create reusable and scalable designs. Using design patterns helps the development of the application in an iterative cycle since each pattern allows the separation of function and implementation [59]. This allows the implementation of client requirements in different stages of application development. Design patterns can be categorized into several subsets based on their function and goals:

- **Creational Patterns:** They specify the methods in which some classes of the domain are created. Based on the needs of the model, separating the creation of the classes from their use allows a design to be extensible and reusable

within different subsets or layers of the application. Within this category the Abstract Factory, Builder, Factory Method, Singleton patterns among others can be found.

- **Structural Patterns**: They define the overall structure and design of the application, allowing the separation of different layers and views within the application. Adapter, Bridge, Composite, Facade and others are patterns found in this category.
- **Behavioural Patterns:** They define the behaviour of a subset of classes or layers and how they should interact. They allow the possibility of decoupling the implementation from actual usage, and how a specific class should be treated and managed during the course of a function within the system. Some of the patterns that are described in this category are Command, Iterator, Mediator and State.

For the user interface, it is possible to define some design patterns to be used. Taking into account that the goal of the application is to be hosted on a mobile device, it should follow a number of user design patterns adequate to the medium. The application should behave according to the expected behavioural patterns of the environment where is developed. Layouts, button placement, and user expectations should be taken into account in order to specify the design of the user interface.

For every mobile platform, including Apple's iOS, Google's Android, Microsoft's Windows Mobile and others, there are user interface standards to be applied. It is expected that the application should conform to the standards specified with the selected development environment for the project. At the very minimum it is expected that the application should make appropriate use of the touch screen interface, gestures and overall application lifecycle of the platform.

3.3 Deliverable of the project dissertation

As a result of the project dissertation, a mobile application will be developed. In order to support the required environment, a layered system using a centralized database and information retrieval middleware will also be constructed. This middleware is needed so the mobile application can use its services.

As part of the resulting dissertation, a section covering the evaluation and selection of the technologies to implement the application will be presented. Mobile environments, such as Apple's iOS, Google's Android and different mapping technologies will be studied and presented in this section. Technologies related to Augmented Reality, which aim to improve the spatial experiences of the user, will also be evaluated and implemented.

The project will produce a prototype/proof-of-concept rather than a production-level application, in order to show the capabilities of the selected technologies. The resulting application would be evaluated by GMP personnel associated with this project. The evaluation will be based on a survey on which each potential user will evaluate each requirement defined for the prototype. Additional factors including ease of use, responsiveness and performance are also evaluated.

3.4 Out of scope issues

The application is a proof of concept, so the project won't be designed to function within the actual expected nature of devices in the field, e.g. space age bespoke headset (similar to Google Glass project) which could be argued is a better solution than an 'off the shelf' tablet or smartphone.

Security constraints regarding the information available in the application/device will not be addressed, e.g. In-the-field security issues (e.g. loss of device, eavesdropping), suspects and victims, information, etc. Also, security issues around using off-the-shelf technologies for the implementation (e.g. trusting Google with your real data, possibilities of protection/loss via/despite encryption etc.) will not be addressed. Data mining opportunities around more sophisticated pre-processing of police/other data prior to display on devices will not form part of the development of the application prototype in the project dissertation.

3.5 How the software is going to be evaluated

Since the result of the project dissertation is a working piece of software, a working prototype with its characteristics described in previous sections; the result and functionalities should be evaluated by the final users of the application. For the final

software evaluation, two main tests are defined for this purpose; acceptance tests, and usability tests.

The acceptance tests of the project will be evaluated by the student based on the complete fulfilment of the main requirements for the application. The main requirements are defined by the GMP, and these are based on a subset of the use cases defined for the whole project. Other types of requirements (performance, reliability, supportability requirements on the FURPS model) should not be evaluated by the final users unless it is clearly specified by the GMP.

The evaluation method is a weighted average for the subset of functional requirements. It would be based on the following criteria:

Each evaluated requirement will have a defined weight in the overall score. If a requirement of the prototype is met by the standards defined for it, it will be given a score of 1 otherwise, its score will be 0. For a successful result in the acceptance test, the overall score of the full set of the user defined requirements must be above a certain percentage (also defined by the client).

To measure the acceptance tests the formula described as follow should be applied:

$$Result = \frac{\sum_{i=1}^{n}(Weight_i * Score_i)}{\sum_{i=1}^{n} Weight_i}$$

Where n = Total number of requirements on the acceptance test

To describe the above process, let assume that given software project has the flowing user evaluated requirements and results for an acceptance test, as described in Table 3-1.

Requirement	Weight	Score
Req1 - Acceptance Requirement 1	1	1
Req2 - Acceptance Requirement 2	3	1
Req1 - Acceptance Requirement 2	2	0
Req2 - Acceptance Requirement 3	1	1

Table 3-1: Example of user tests and results for each requirement.

Using the results in the table and applying the equation, the result would be equal to 83%. In this example, if the user defined that the minimum threshold that the project

requirements should be 90%, the project would fail the acceptance test, but if the threshold was defined as 80%, the acceptance test is deemed satisfactory. A detailed definition of the requirements for the prototype is specified on the Design section of this dissertation.

The final product should also be evaluated based on usability tests. Usability testing is the process on which the interaction between the software and the user is evaluated as described by [60]. The goal of usability testing is to receive a direct input of this interaction and to measure the capacity of a given piece of software to meet the user's expectations. In order to evaluate the final prototype of the application a System Usability Scale (SUS) is going to be used.

The System Usability Scale was developed by John Brooke [61], and aims to evaluate on a scale of agreement the interaction of the users and the software. The valuation is made on a series of questions given to the user after following a predefined interaction. The questions to be asked to the user are specified in Table 3-2:

Number	Question
1	I think that I would like to use this system frequently
2	I found the system unnecessarily complex
3	I thought the system was easy to use
4	I think that I would need the support of a technical person to be able to use this system
5	I found the various functions in this system were well integrated
6	I thought there was too much inconsistency in this system
7	I would imagine that most people would learn to use this system very quickly
8	I found the system very cumbersome to use
9	I felt very confident using the system
10	I needed to learn a lot of things before I could get going with this system

Table 3-2: Questions asked to the system evaluator on a SUS **[61]**.

Each of the questions is evaluated on a scale from 1 to 5, where 1 is "Strongly Disagree" and 5 is "Strongly Agree". The score from each question goes from 0 to 4. The SUS generates a single score representing a measure of the overall usability of the system. (Scores from individual questions are not meaningful on their own). To score the overall SUS score, even questions are graded 1 minus the scale position. Odd imbed questions are graded 5 minus the scale position. All grades are added and

multiplied by 2.5 to obtain the overall result. SUS scores have a range from 0 to 100. It is expected that an application that scores over 80 has a good usability. For all applications, the average of SUS evaluation is 68 [62].

The overall evaluation from the project clients will be based on both the requirements and usability tests, and it could show the overall effectiveness of the project result from the GMP perspective.

4. Design

The application development is based on a standard iterative software development cycle. This process includes requirements gathering, overall system design, implementation and testing. On this chapter, the requirements and the overall application design are discussed in each of the following subsections.

To gather the full extent of requirements, separate meetings were held to discuss the project and client needs. These meetings were coordinated with the project supervisor and the client (Greater Manchester Police). An overview of the 2 types of requirement gathering meetings with the project stake holders is described below.

4.1 Project's requirements

Since the project inception, the student met with the project supervisor to discuss the advancements and progress on a weekly basis. On each meeting there are several subjects that are addressed and at the end of each meeting new tasks and goal are made to be reviewed on the following week. The tasks range from simple activities like reading on a specific subjects or related to a given deliverable of the prototype. The main milestones defined were:

Technology review: Since the project inception, the project objective is to implement a mobile application using location based technologies. To ensure a successfully developed application, a formal review of available technologies was needed. A correct implementation needs to take into account readily available technologies based on a multitude of software APIs and Hardware providers. The result of this milestone is reviewed on a more detailed basis in the "*Selection of Technology*" section.

Overall application design: Based on the objectives dictated by the project description, an overall application design was needed to envision the full extent of the effort needed to complete it. Several exploratory designs were made to describe the overall application features and functionality and to describe the layers that will form the full environment. Overall, the layers that were defined for the application were the data layer (covered by a database) a middleware (data gathering, processing and

formatting) and mobile application (user facing component of the application; consumes the services provided by the middleware). The result of this milestone is reviewed in the "*Architecture Design*" section.

Prototype development: During the initial weeks of the project, efforts were done to start the implementation of the software application. The aim of the initial milestones was to make a proof of concept for several technologies related to the overall goals of the application. Progress was made on several fronts, from map related development, sensor analysis and an initial review of video overlays for an enhanced world view for the user. This development served as a "proof of concept" that was later introduced into the overall application development. The detail of the project implementation, including the technologies that were proofed is specified in the "Implementation" chapter.

4.2 Functional Requirements

The prototype that is being developed will need to serve current needs expressed by the GMP. A requirement gathering process was followed to formally retrieve those needs. To achieve this, 2 meetings were held to gather the main and secondary requirements from the GMP. The first meeting was with the main supervisor of the project from the GMP side, (Peter Langmead-Jones) in which the main requirements were laid out. A second meeting was made with another contact form the client (Duncan Stokes) in which the requirements were further developed for the project. As a result of the meetings, the requirements were gathered and categorized.

Application/Software Requirements

The project must provide a mobile application that should be able to display 'summary data', originating from a number of sources (GMP databases, public available services), in a visual display that correlates data to an approximately 'photo-realistic' image of its geographical source.

The artefact (software prototype) won't address security issues, such as legal/regulatory issues, data protection, anonymity, etc. The prototype should show an intelligent enhancement over the services provided by current geo-based services (crime-mapper/police.uk website). The application should behave in a "push"-based

interaction with the user, rather than "pull"-based (the application will notify the user rather than the user proactively looking for the information).

The nature of the data retrievable by the constructed system should include crime scenes, suspect and victim locations, vulnerable individuals, information regarding the nearby properties (Key-holders, issues, street furniture), and other relevant data that might help the users of the system to take appropriate and accurate actions. The data to be used on the development of the application is provided in part by the GMP (providing “dummy”, not real information).

Other information could also be queried from other public available services provided by the local government entities. An appropriate device for the development of the system, according to the technology evaluation, should be provided by GMP.

GMP provided a limited information set based on the Rusholme area (near the M14 4DJ post code) that matches the current structure of the data managed by GMP’s internal systems. With the data provided, a data management layer must be developed to analyze said data and provide that information to the police officers on the beat, using the most appropriate display like a map based interface and augmented reality display.

Table 4-1 shows a summary of the consolidated requirements gathered by the student during this process, including the project milestones and the needs of the GMP as a project client.

Requirement	Type	Description
1	Client	Application must Display Summary data of the GMP on a mobile display for use by the police on the beat.
2	Client	Application must show the data based on the geo-location of the information. The display must correlate to a 'photo-realistic' image of its geographical source.
3	Client	The prototype should show an intelligent enhancement over the services provided by current geo-based services (e.g police.co.uk).
4	Client	The application should behave in a "push"-based interaction with the user, rather than "pull"-based.
5	Client	The nature of the data retrievable by the constructed system should include crime scenes, suspect and victim locations, vulnerable individuals, and other relevant data.
6	Project	Review of available technology.
7	Project	Development of proof of concept designs.
8	Project	Development of application prototype that reflect the user requirements.

Table 4-1: Summary of client and project requirements.

4.3 Selection of Technology

To achieve the results specified on the requirements made by GMP, the first step is to select a technology that can appropriately support the selected features. In order to evaluate the technology, key characteristics were analyzed and compared between the current mobile vendors. The features that were taken into consideration for the evaluation are:

Map Support: The availability of a programmable map interface to compose a custom application with related map layers, markers and overlays.

Video Support: In order to present the location based information in a photo realistic environment to the user, video support is needed. The ability to programmatically manipulate video displays, add overlay images, and control the on board camera is imperative.

Sensor Information: The ability to detect and obtain sensor information to position the user on a realistic location in the world. Sensors like GPS, accelerometers, orientation, gyroscopic and light sensors are needed to correctly locate the user position and direction and present him with the relevant information related to his current location.

Programmability: The possibility to develop a multilayered system on the device, able to communicate with external systems, control notifications (in text, light and vibration forms), usage of services (separately running services from the main application). It should be possible to control the application lifecycle in order to reduce energy consumption and appropriate usage of sensor information. Ability to produce an application suited for mobile and tablets without additional code.

Price/Licensing: It should be possible to develop the main application without the need of external licenses, and should be cost free to develop and publish. Usage of free and open source software should be expected for the prototype.

For the evaluation of said features, the biggest mobile vendors were selected based on current market share. These vendors are Google with the Android Mobile Operating System (OS), Apple with iOS and iPhone, iPad mobile devices, and Microsoft Windows Mobile on Supported devices.

Table 4-2 shows the comparison of features between the different mobile vendors and the features analyzed.

	Vendors		
Feature	**Apple**	**Google**	**Microsoft**
Map Support	Introduced Apple Maps on iOS 6. Before that version used google maps API. No restrictions on use.	Offers Map Support through Google Maps API. Free to use if used less than 25000 times per day. Supported natively on all Android phone and Tablets.	Provides Windows MAP API that uses Bing Maps and Nokia services for location based applications. Restricted to use the API on navigational, asset tracking and business applications.
Video Support	Apple devices provide a wide range of camera devices that can capture the surrounding environment with an appropriate resolution.	Google devices provide a wide range of camera devices that can capture the surrounding environment with an appropriate resolution.	Nokia devices provide a wide range of camera devices that can capture the surrounding environment with an appropriate resolution.
Sensor Information	Latest iPhone and iPad devices are provided with the following sensors: GPS, Accelerometer, Gyroscope, light sensor, proximity sensor.	The Nexus range of devices provides the following sensors: GPS, Accelerometer, Gyroscope, light sensor, barometer, magnetometer, and proximity sensor.	Top of the line Nokia Lumia devices provide the following sensors: GPS, Accelerometer, Gyroscope, light sensor, magnetometer, proximity sensor.
Programmability	iOS Applications are developed with the Object-C programming language on the Xcode development environment. Development is restricted on Apple Mac devices.	Android apps are created on the Java language using Android development API. Development can be made on windows or Linux platforms using an eclipse IDE plug-in.	Windows 8 Apps are developed on .NET Languages (C#, Visual Basic .Net) and can only be developed on Windows OS.
Price/Licensing	To deploy an app on the marketplace an application has to be approved by an Internal apple team. A licensing fee of USD$ 99 is needed. It is not possible to legally distribute applications outside the market.	Free application distribution can be done via marketplace or by side-loading directly on the device. There are no restrictions on distribution of apps outside the marketplace.	To deploy an app on the marketplace an application has to be approved by an Internal Microsoft team. A licensing fee of USD$ 99 is needed. It is not possible to legally distribute applications outside the market.

Table 4-2: Overview of Mobile Vendors features and characteristics.

After reviewing the all the capabilities offered by the mobile vendors and taking into account the overall cost of implementation, the Android development environment was selected. It is expected that the nature and maturity of the services provided by Google will provide a strong environment to develop the desired application.

The selected tools for the implementation are:

- **Operating System (Development)**: Windows 7
- **Development Language**: Java SE 7.
- **Development Environment**: Eclipse Integrated Development Environment (IDE). Includes Android development Plug-in.
- **Mobile platform**: Android 4.1.1 and above (Jelly Bean).
- **Device**: Samsung Galaxy Nexus.

4.4 Overall system design

In order to meet all the GMP requirements, a design was envisioned to support all the needs of the client. The system and mobile applications designs have to take into account several restrictions from the data management standpoint and from the mobile environment in which it will perform.

From the meetings with the GMP it was noted that all the data information is stored in plain text files, without proper data normalization. Even if the application in the prototype stage will only manage a reduced set of data (centred on the M14 4DJ post code), in the actual running environment it could manage and / or query a much greater set of information. Since the application should perform in a push-based mode (the application should notify the user if there is an important set of locations around the current user position), it should behave like a background service, but using the least amount of power to reduce the energy consumption on the mobile device.

Taking into account those restrictions, a three-layered design was created. The three layers are composed of the Database (DB) Layer, System Middleware (MW) Layer and Mobile Application (MA) Layer.

Database Layer: Describes and stores the information provided by the GMP. It consists on a normalized central database that will store the information of the structure needed by the middleware and the mobile application that will consume its services.

Middleware Layer: A server side component that performs 3 basic functions. Provide services to the mobile app, query and manage the data stored in the data layer, and

perform the data transformation from the information provided by the GMP and related public available services in the internet.

Mobile Application Layer: This is the user-facing part of the application, provides context and location specific data to the client. Shows Map based and augmented reality information to the user. It will follow the application lifecycle of the selected environment, and will provide the necessary tools to query the information to be consumed by the services provided by the middleware layer.

Figure 4-1 shows the overall design of the layers and their interactions during the lifecycle of the application. While it provides an overview of the system functionality, each of the layers and their interactions are explained further below.

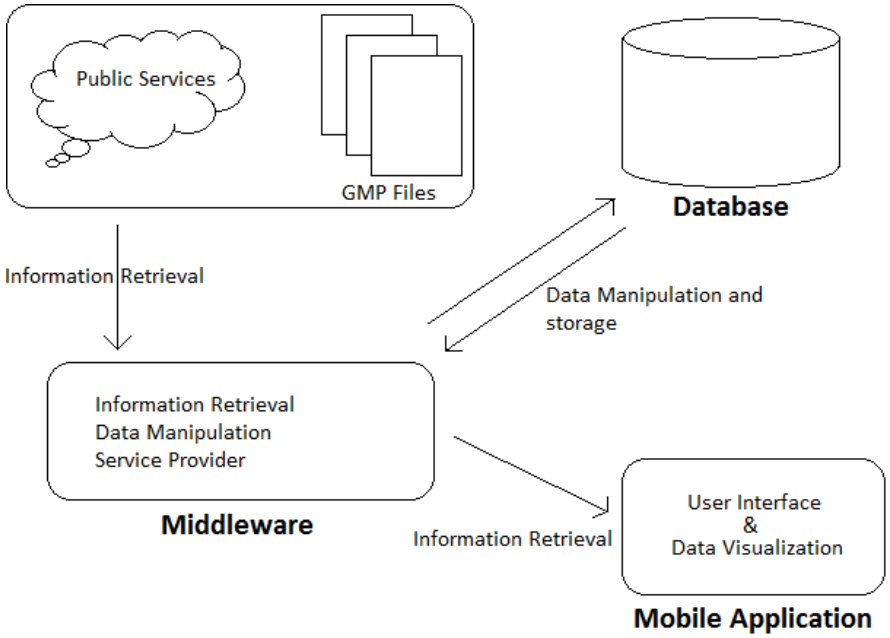

Figure 4-1: General system design of the application.

The DB layer stores all the information related to the GMP data, and also stores application specific data (configuration, data structures, etc when needed). The database design follows normalization rules to ensure reduction of data redundancy and dependencies. The data structures defined in the database are important, since they hold the appropriate definition of the data to be queried by the app. It also ensures that data from other sources, other than GMP files, is stored using a standard structure.

The DB layer can only be accessed directly by the Middleware (MW) layer. The MW layer performs queries and data operations (insertion, updates, deletes) over the

information on the database. The MW layer also has the function to capture, process and format the information from different sources (GMP files, public available services). After processing the data, the middleware stores the information in the appropriate structure on the database. This function ensures that data from different sources and formats will be available by the mobile app at run-time. The MW layer uses software design patterns to perform its operations in an extensible way, and uses them to ensure an adequate level of performance, by ensuring proper methods of data processing (e.g. caching, connection pools).

The mobile application (MA) layer consumes the services provided by the MW layer and services from the maps components. The application itself does not store much information on the device but will use the information from the sensors, to accurately calculate the devices position and orientation to present relevant data to the user. The middleware application has 3 modes of operation.

- Display of location data relevant to the user's position on a map interface. This mode will allow the user to use touch gestures to zoom specific locations and display relevant information.
- Display of an augmented reality display, using the device's camera and overlay markers on top of it, to display on the user's point of view. The information displayed is limited to their proximity to the user's view, which should be around 200 meters around the user's position.
- The mobile application also performs queries taking into account the user's location. This is done through a service to notify the users of important information near its position. The notification is done via light, sound or vibration, while the user is not actively using the application.

The mobile application should also have some configuration options which will allow the user to specify which kinds of notifications are displayed, and which information will be presented by the application.

4.5 Database Design

As a result of the initial meetings with the project stakeholders, a set of data has been delivered to the student with the same structure used by the internal systems used by the GMP. This data was provided by the client as a set of Microsoft® Excel sheets with the following configuration:

- Each Excel sheet displayed information related to one table
- The first excel sheet showed the relation between the tables (indicating which columns has relations to columns in other tables).
- The first row of each table contained the name of the columns.

The original design provided by the GMP is displayed on Figure 4-2. This figure shows the names and relations of the tables as used by the GMP internal systems. On the figure, each computer icon represents one table in the model. The design, as originally provided, has some naming conventions and in tables and columns that make difficult its analysis and readability.

In order to understand the model and the multiplicities of the data, a series of analysis have been made on the information provided by the GMP. This analysis involved making a sample of each table, and a review the data in each column. This process allows understanding the amount of data available, the type and the context in which it is used.

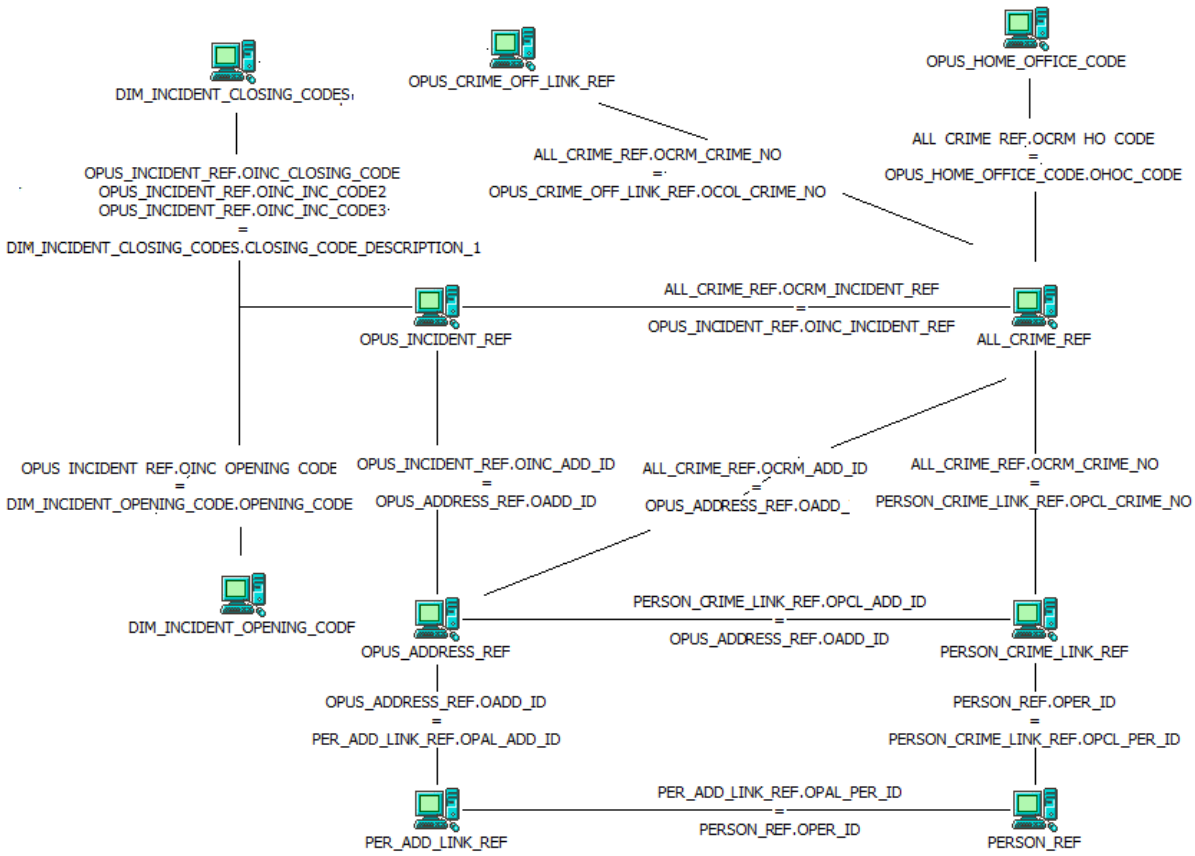

Figure 4-2: Database design provided by the GMP.

Once this process has been made, 2 main outcomes surfaced. First, a table description and table name translations. For each table provided by the GMP, a general description has been created for it, the relation to other tables, and a new name given to the table to be able to understand it better. This result is shown in Table 4-3. The second outcome of the analysis is a specific Entity-Relationship diagram, which shows the simplified table names, the relationship between the tables, and the multiplicities expressed by the data. This result is shown in Figure 4-3.

GMP TABLE	Description	Related to	TRANSLATION Table
ALL_CRIME_REF	Master Table of all the crimes related in the area of study. Hold references to the location (in British Coordinates) where the crime was executed. It has references to the persons, crime codes, and address.	OPUS_INCIDENT_REF OPUS_CRIME_OFF_LINK_REF OPUS_HOME_OFFICE_CODE PERSON_CRIME_LINK_REF OPUS_ADDRESS_REF	CRIME
DIM_INCIDENT_OPENING_CODE	Each incident is opened with a description. It explains the type of incident.	OPUS_INCIDENT_REF	OPENING_CODE
DIM_INCIDENT_CLOSING_CODES	Description of why the incident was closed. It expands on the opening codes, for example if the incident was opened with "Alarm" the closing code would be Alarm Installation, Alarm, genuine arrest, etc.	OPUS_INCIDENT_REF	CLOSING_CODE
OPUS_INCIDENT_REF	Hold the references to all the incidents; with markings to the type of incident (an incident can have several types, child abuse, domestic violence, etc).	DIM_INCIDENT_OPENING_CODE DIM_INCIDENT_CLOSING_CODES ALL_CRIME_REF OPUS_ADDRESS_REF	INCIDENT
OPUS_CRIME_OFF_LINK_REF	Information related to the linked suspects. How the person was dressed, does it have a photo, does it have fingerprints collected.	ALL_CRIME_REF	LINKED_SUSPECT
OPUS_ADDRESS_REF	Holds the address of all the crimes and persons related to the crimes. Street, district, post code, coordinates,	PERSON_CRIME_LINK_REF ALL_CRIME_REF PER_ADD_LINK_REF	ADDRESS
PERSON_CRIME_LINK_REF	Table used to have the references between the persons and the crimes. It also holds the references to the address where the crime is committed.	ALL_CRIME_REF PERSON_REF	PERSON_CRIME
PER_ADD_LINK_REF	Table used to link the person and the address of the person. The address of a person might be different than the address where the crime was committed.	OPUS_ADDRESS_REF PERSON_REF	PERSON_ADDRESS
PERSON_REF	Holds the references to all the people related to any crime/offence in the data base. A person can be associated to several crimes.	PER_ADD_LINK_REF PERSON_CRIME_LINK_REF	PERSON
OPUS_HOME_OFFICE_CODE	Has the codes and description of all the offences. It is related to the master table and it is used to describe the type of crimes	ALL_CRIME_REF	CRIME_TYPE

Table 4-3: Description and table names of the GMP data.

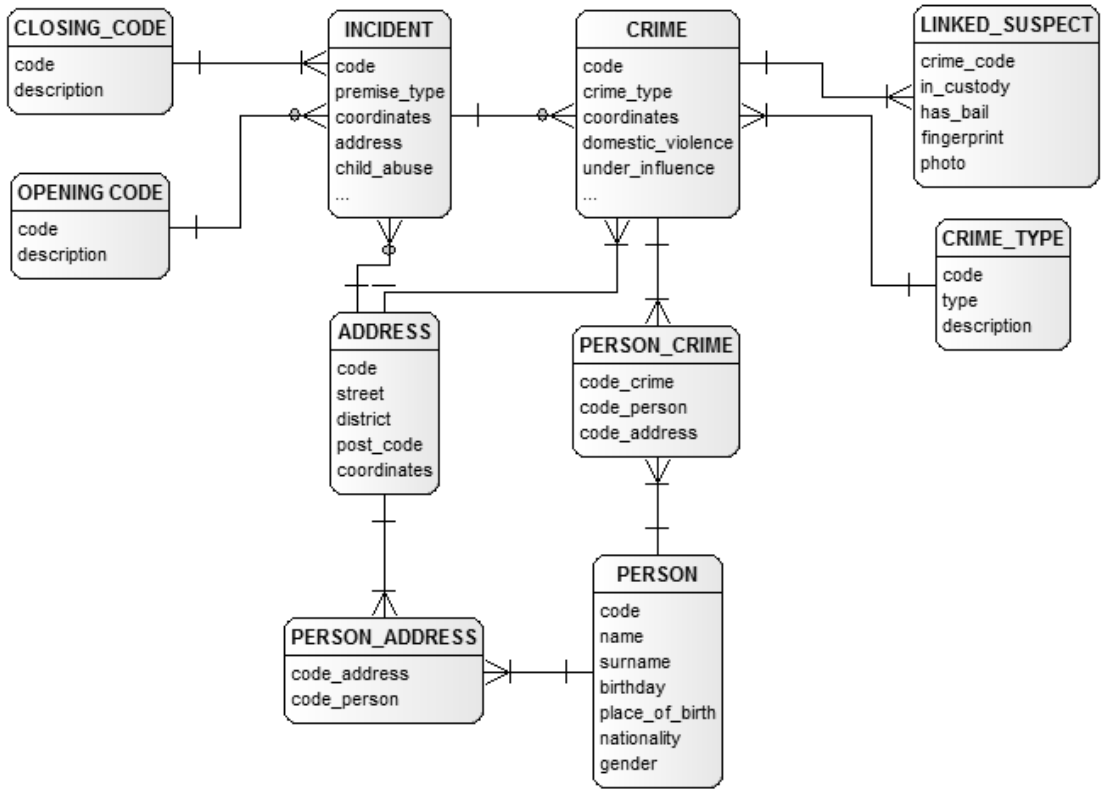

Figure 4-3: Entity-Relationship Diagram of the data provided by the GMP.

The Entity-Relationship diagram (ERD) of the data shows more clearly the table's names, relations, multiplicities and main attributes. With the help of the data analysis described above and the ERD is possible to infer several important characteristics of the model:

- The 2 main tables of the model are INCIDENT and CRIME. An incident could have from 0 to several crimes related to it.
- An incident has both an opening code and a closing code (in reality, an incident can have 3 different closing codes, but to simplify the model, these closing codes have been aggregated into one).
- The relationships for the CRIME table shows that a crime can have several persons associated to it, and a person can be related to several crimes. This type of relationship implies that an intermediate table is needed. This table is PERSON_CRIME, which helps with data normalization, by way of reducing "many-to-many" relationships [63]. On the same vein, an Address and a Person have the same type of relationship, hence, the table PERSON_ADDRESS is needed.

- In the model, both CRIME and INCIDENT have a relationship with the Address table. During the data analysis it has been possible to identify that both tables refer to the same address registry (row in the database). This implies that the application only needs to query this information from one of them. The same analysis is valid for the coordinates that identify the crimes and incident on a geographical location.
- There are several types of crimes that identify a crime. One crime has only one type associated with it. The crime table has a relation with suspects (offenders of a crime) and how were they dealt with. Originally, there is no relation from the LINKED_SUSPECT table with PERSON table, but the data analysis shows that there is a relation between them.
- All geographical coordinates' records in the database are based on the UK Ordnance Survey National Grid reference system (OSRef), which uses easting and northing as coordinates [64]. This is an important piece of information, due to the fact that the android environment works with the Geographic Coordinate System based on the GPS system, which uses latitude and longitude as coordinates [64].
- Not all the attributes are displayed on the ER diagram. Due to the amount of columns each table has, the attributes shown are the ones needed to explain the overall model, however each table significantly more attributes.

In order to provide an adequate environment to store and manipulate the data, a Database Management System (DBMS) was selected. This selection was based on the same principles described in the "selection of technology" section above (in particular: Price, Licensing, and Programmability). The DBMS selected was the latest version of MySQL as described below:

- **Engine:** InnoDB, version 5.6.11
- **Description:** MySQL Community Server (GPL)
- **Type & OS:** x86, Win32.

4.6 Middleware Design

To design the middleware layer, the main requirement that has been taken into consideration is the ability to provide communication services, so to the application layer could indirectly query the information stored in the database.

Taking into consideration the mobile characteristics of the application and the technologies available, the middleware layer is designed as a web server that queries the information from the database and presents the data on a Representational state transfer (REST) web service. REST services have been selected, due to their simplicity of implementation, and based in the fact that the Android environment doesn't have full native support to Simple Object Access Protocol (SOAP) based services.

REST services in general are supported by common internet communication protocols such as Hypertext Transfer Protocol (HTTP), Uniform Resource Identifier (URI), and HTTP Request Methods (POST, GET, PUT, etc). This supportability makes the design, implementation and use of the services much easier than other services architectures.

Each designed service outputs a result of a specific query on the database. The output of the services is text formatted in a way that could be parsed by any client. Accounting for simplicity, the text format is defined as coma separated values (with ";" as a separator token). Each row of the result is separated by a new line. A format definition example is shown below for a given result that has 3 rows and 3 columns:

Row1Column1;Row1Column2;Row1Column3;
Row2Column1;Row2Column2;Row2Column3;
Row3Column1;Row3Column2;Row3Column3;

This format can be read and parsed by any client, without the need for additional parsers (e.g. Extensible Markup Language (XML) parsers for XML results).

Taking into consideration the defined development architecture, and based in previous design principles and features (specially the use of Java Language), a web application server has been selected to provide the services for the system. The web application server selected was the latest version of Apache Tomcat (Version 7.0.40).

To design an extensible architecture for the middleware, the services will be created based on simple Java Servlets working as REST services. These constructs are Java classes defined to extend the capabilities of a simple web server by providing a programmability layer to the different HTTP requests.

The middleware is designed to present a single point of contact for the clients, however, that single point of contact should abstract calculations and connections made to retrieve the data stored in the database or other sources of information. To accomplish this feature, a design pattern was devised for the services' implementation, which provides this extensibility. The general design of the service is shown on Figure 4-4. This is a general design guideline used as a basis to implement all the required services for the system.

Each service is a class that inherits from Java `HttpServlet` class that provides the point of entry to any client that queries the information. Each service class has an associated class that will perform the appropriate database connections and needed calculations. This class is responsible to query and manipulate the data and to deliver the result to the service class so it can display it to the clients.

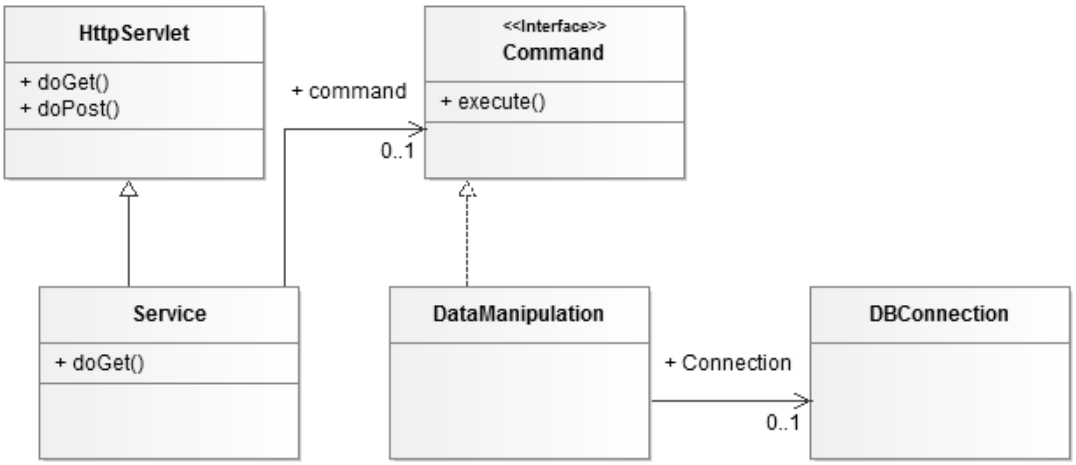

Figure 4-4: Design guideline of the middleware services' architecture.

This design guideline for the middleware services provides a way to extend the capabilities of the defined architecture. When the servlet, which is the point of contact of the clients, delegates data connection and manipulation to another class, it is possible to extend the functionality of the service. The service is not aware of the actual implementation of the data manipulation class, it only knows and makes use of

the interface that the concrete classes implements. This interface follows the "Command" structural design Pattern [59], in which the execution of the manipulation is masked on a single operation called "execute".

The concrete data manipulation classes can connect to different databases, could consume other services in different servers, and can be changed without affecting the service consumed by the system clients. The implementation class can be of any type, as long it implements the "`Command`" interface used by the service.

This middleware design allows the system not only to present a common interface to query GMP data, but it also able to extend its functionality to other types of sources if future implementations or versions require it. During the course of the project, the only data available to query will be related to the information provided by the GMP.

4.7 Mobile Application Design

The mobile application layer of the system is comprised by the mobile application that the client will use. Most of the project and client requirements are visible through this layer. There are two design concerns in this architecture layer, which consist on the User Interface (UI) and the internal architecture of the mobile application to support the desired features. This section details the overall design of both concerns.

4.7.1 User interface

As described in the system design section, the user interface has 3 modes of operation; Map view, Augmented Reality view and background service. It should be possible for the user to consistently manage the application features and doing so using the design guidelines provided for the android development environment [65]. In order to provide an adequate user experience and taking into consideration the limitations of the device (screen size, input controls, among others), the following design decisions are specified:

Figure 4-5 shows the selected device orientation for the mobile application. While setting a defined orientation may limit the user interaction, it provides a consistent experience for the user during operation. The selected orientation is "portrait", based on the common usage of the map applications and the device's camera. Appealing to

user interface standards of display and operation increases the user's recognition of an interface and makes it easier to use, as described by several usability guidelines [60] [66].

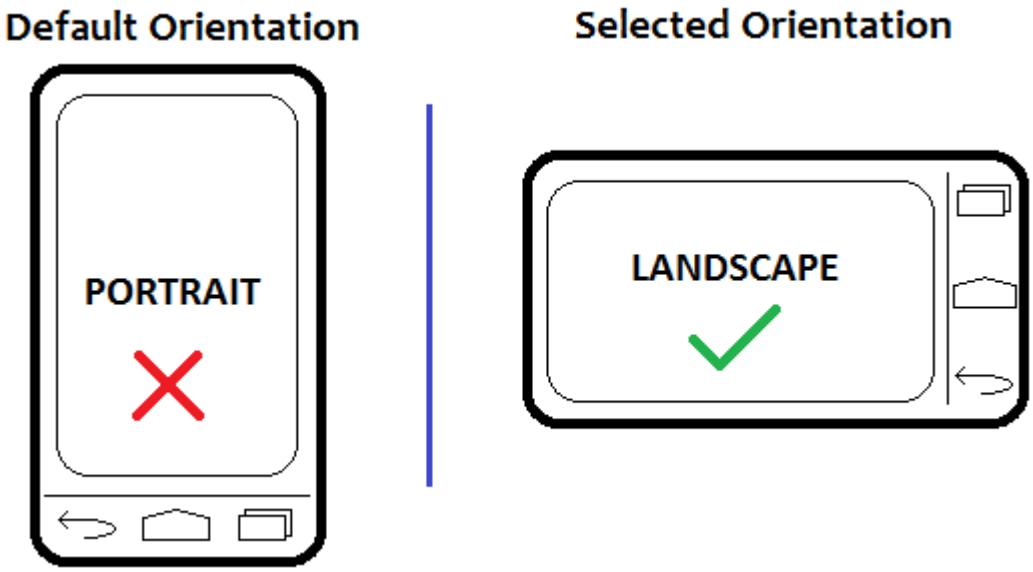

Figure 4-5: Selected device orientation for the UI.

All the screens in the mobile application follow the pattern described in the Figure 4-6. On the top of the screen the user will find an icon that represents the application, a name describing the specific mode/feature that is being used and a settings button to access the settings that the user could customize.

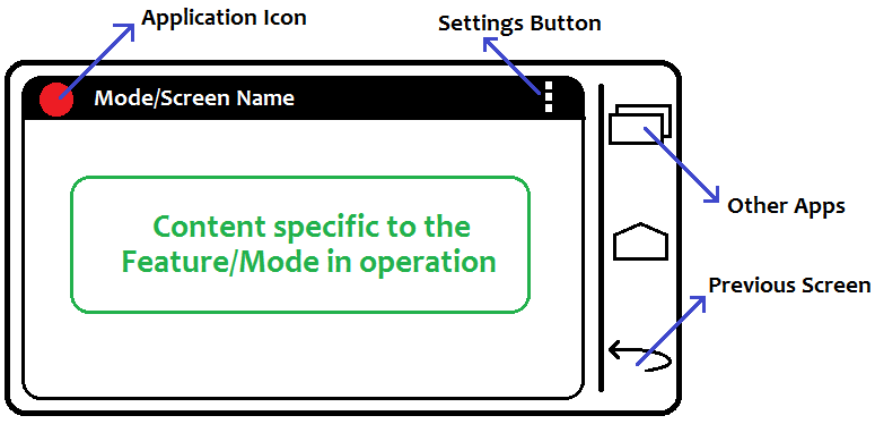

Figure 4-6: Common application interface for all screens.

On the right of the screen the user will find Android specific buttons that represent navigational operations related to the current user interface and the overall android environment. From those buttons the user can navigate to other open applications and could return to previous interfaces on the same application. The main portion of the screen is reserved to present the information related to the current feature that is

being used. The information in this screen is related to the Map and Augmented Reality features, and additional screens that are designed to present extended information related to the main modes of operation.

Figure 4-7 shows the main screen of the application. From this interface, the user can launch its two main modes of operation (Map and Augmented Reality modes). Although the application has three modes of operation; there are only two modes visible; the background service is invisible to the user.

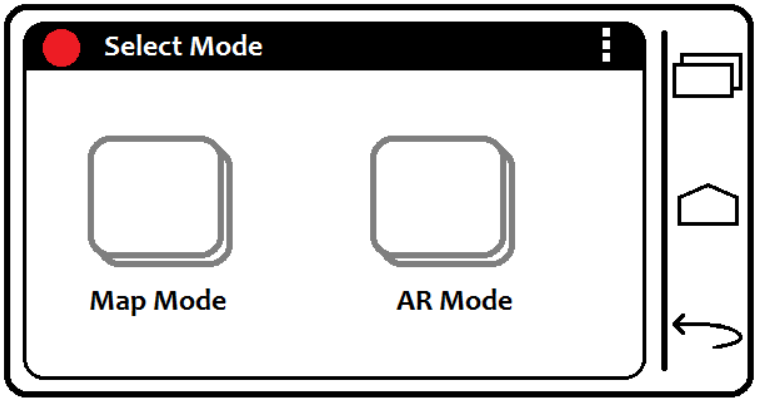

Figure 4-7: Main interface - mode selection view.

On this screen the user is presented with 2 buttons representing each of the available modes. After the user touches the Map mode's button on the screen; the interface is displayed as shown on Figure 4-8.

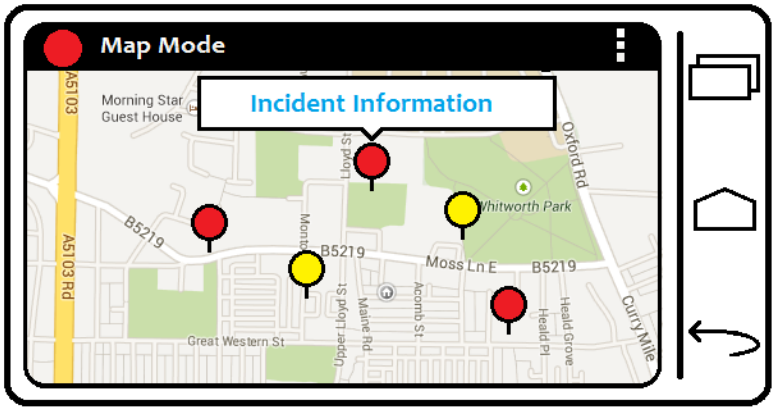

Figure 4-8: Interface design of the application's Map mode.

This mode displays all the incidents near the user's current position. There are 2 distinct markers on the map; the yellow marker makes reference to an incident that has no related crimes, the red marker references an incident that has crimes associated with it. When the user touches one of the markers in the map, a window

with a summary of data about the incident is displayed. If the user touches the summary window, the application exits the map mode and displays the detailed information of the incident.

If the user selects the AR mode's button from the selection screen, the Augmented Reality mode is launched. The design of this application mode is shown on the Figure 4-9.

Figure 4-9: Interface design of the application's AR mode.

The AR mode makes use of the device's camera to display a view to the outside world. The view presented to the user is augmented with overlays representing the approximate location and direction of nearby incidents. From this mode it is possible to locate, just by pointing the camera, all the incidents that surround the user. The interface guidelines in this mode follows the same design pattern than the one described on the Map mode interface. The overlay colours represent the same information than the map markers (a red overlay represents an incident with related crimes; a yellow overlay represents an incident that doesn't have them). When the user touches one of the overlays, a small window is displayed with a summary of the incident's information. Once the user touches this window, the application exits the AR mode and displays the detailed information of the incident.

The Map and AR modes show information about nearby incidents. When the user touches one of the indicators, each mode will exit and the detailed description of the incident is shown. Figure 4-10 and Figure 4-11 show the interface design of the main data entities, which are "incident", "crime" and "person". To help the user's navigation, the interfaces follow a simple navigation pattern, in which each screen

shows the detailed information of the specific entity. On the bottom of each screen there are buttons that link to its following entity.

According to the relationships expressed in the database design, an incident could have related crimes and a crime could have related persons. The person is a final entity; it doesn't have other entities associated. For each of the entities a screen has been designed.

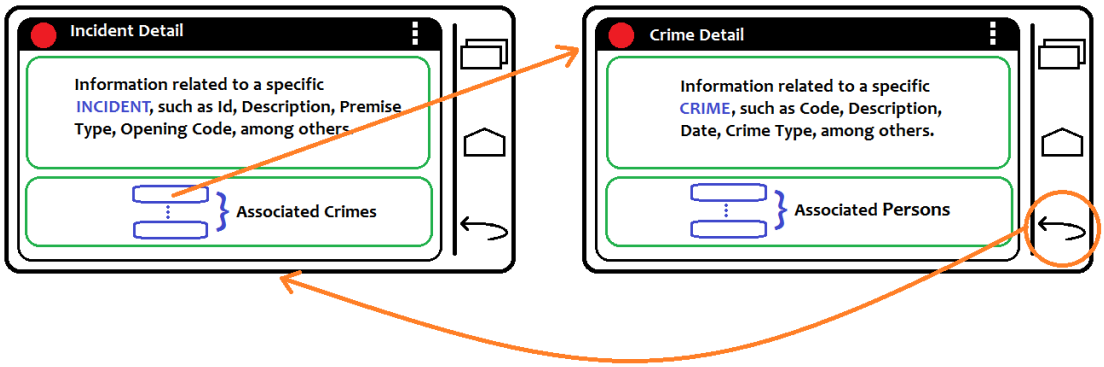

Figure 4-10: Interface and transition between "Incident" and "Crime".

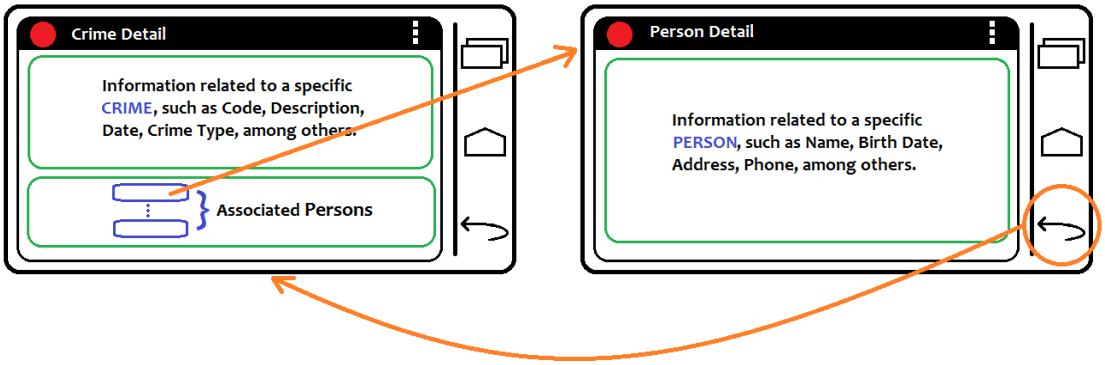

Figure 4-11: Interface and transition between "Crime" and "Person".

The previous figures (4-10 and 4-11) also show the transition between the different entities pages. The initial screen refers to the "incident detail" page. This screen show the main incident attributes, and in the bottom of the page, one button for each related crime is displayed. When the user touches one of the crime buttons, the "crime detail" page is displayed. According to the android navigation guidelines, a user could return to the previous page by touching the "previous" button on the right side of the user interface. The transition between the crime and person interfaces follows the same guideline, taking into consideration that the "crime detail" page has as many buttons as persons associated with the crime.

The interface has a settings button on top of all the application screens. This button displays a menu with all the configuration settings that a user can apply to the system. There are two types of settings, Filter and Application. Filter settings have all the filter options that the user can apply over the data queried by the application e.g. query incidents that have crimes associated, only show crimes where alcohol influence is present, etc. Application settings are related to the overall application functionality, e.g. server address, vibration options for notifications, etc. Figure 4-12 shows a mock-up design of the settings pages that are available for the user.

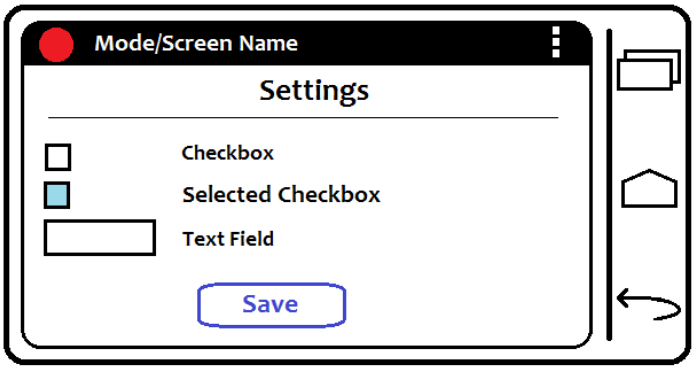

Figure 4-12: Design page for settings options.

All settings pages contain different types of “widgets” such as check boxes, text fields, among others, to provide the user the possibility to save the desired configuration, according to the type of data. At the end of each settings page a “Save” button is present which allows saving the desired changes.

The above design sketches provide a view of the overall structure of the application features, complete with navigation options and the type of information that the user has available, in order to use the application and take decisions with it. The design takes in consideration the structure of the data, size of the device, mobile environment and usability practices to allow a simple, yet effective operation while providing a great amount of freedom in its use.

4.7.2 Architectural design

Based in the overall system design, the data structures and the user interface definition, it is necessary to create an application architecture that supports the features and requirements of the users. The designed architecture takes into

consideration the development constraints imposed by the mobile environment, its design guidelines, and the separation of the interface from the underlying data model.

The application architecture can be divided in different layers (UI, Control and Data Model) that hold distinct responsibilities and functions as shown in Figure 4-13. The UI layer is responsible to display the information to the user and to manage the application lifecycle. The control layer is responsible to provide the settings, sensor and location services to the classes that that perform the main calculations and data processing. The model layer refers to the data structure retrieval, representation and caching mechanisms used for data persistence.

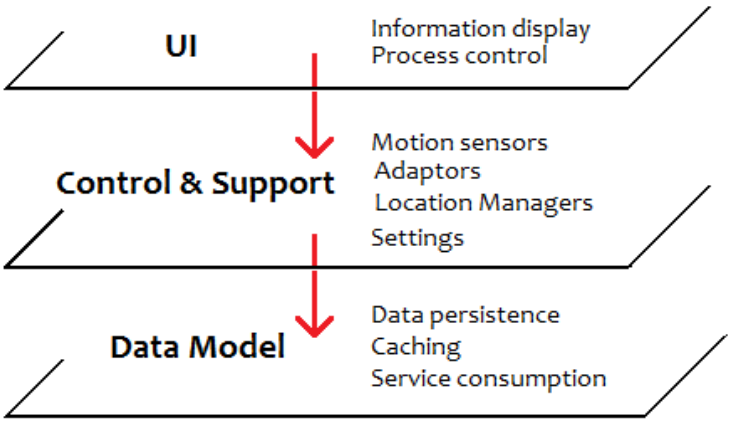

Figure 4-13: Mobile application design layers.

UI Layer: This layer contains all the classes that represent visible interfaces to the user. Each page in the user interface is represented by a class that is responsible to retrieve information from the control layer to display the appropriate information to the user. Based on the UI mockups described in the previous section, an architecture has been created to follow the navigation pattern defined for the application. This architecture is presented in the form of a skeleton class diagram in Figure 4-14, which shows the general class structure without implementation details.

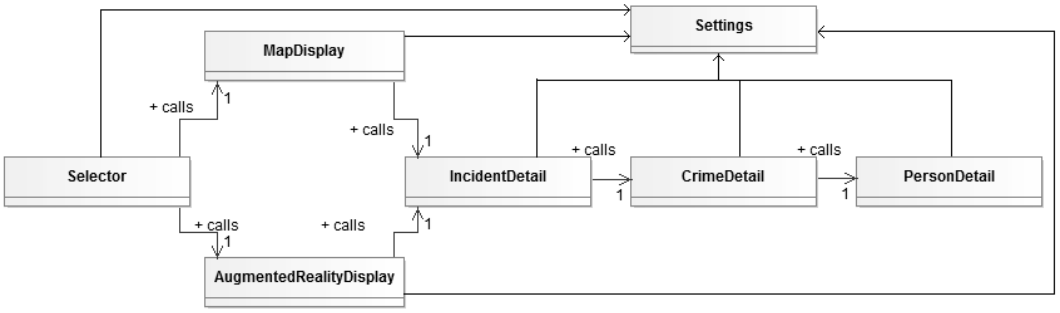

Figure 4-14: UI design – skeleton class diagram.

Each class on the Figure 4-14 represents a visible interface for the user. The first class on the UI design is "`Selector`", which has the responsibility to display the Mode selection interface to the user. After receiving input of the mode selection from the user, the class gives the UI control to the appropriate class in the process. If the selected mode is "Map", the "`MapDisplay`" class takes control of the Interface; otherwise, the "`AugmentedRealityDisplay`" class takes it.

Once the classes that display the main application modes (Map, AR) have displayed the UI and received input from the user, the "`IncidentDetail`" class is then given control on the UI. From this class the UI process is linear, where the "`CrimeDetail`" and "`PersonDetail`" classes take control respectively after the previous class has displayed the interface and received the appropriate command from the user.

From each class in the UI, it is possible to give control to the `Settings` class that represents the settings interface. This is possible since every user interface has "settings button" displayed on top of the screen as described in the UI mock-ups. In the android environment, it is possible to return to the previous interface by touching the "previous" button. Since this feature is standard to all android interfaces, it is not modelled in the architecture.

Model Layer: The classes that make part on the model layer are represented by the main entities of the system, which as described before are "Incident", "Crime" and "Person". The model layer has the responsibility to query the entity data from the services provided by the middleware and instantiate the entity classes according to the information retrieved from them. As shown on Figure 4-15, the model layer design provides a set of classes that represent the entities, a mechanism to provide data

persistence and the classes that provide the information retrieval from the service. Each entity (Incident, Crime and Person) implements the user design structure shown on Figure 4-15.

Each entity has a `CacheManager` class associated with it. The cache manager has the responsibility to store references to the entities in memory to improve performance by reducing middleware service calls to retrieve information that has been queried before. If a specific entity is not on the cache, the cache manager makes use of the "`ServiceConnection`" class to query the data from the services.

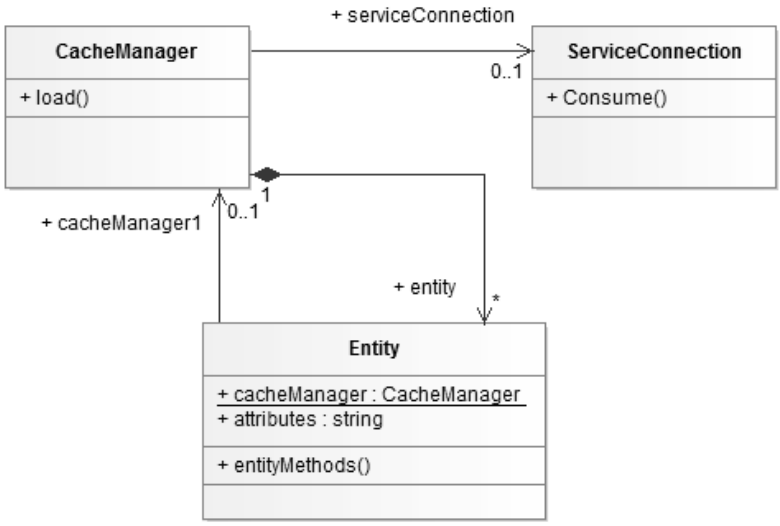

Figure 4-15: Model Representation – skeleton class diagram.

Once the service has been called and its response has been processed, the cache manager creates an instance of the entity, and stores it in memory. The outside layers of the application only interact with the entity class through the entity's statics methods; all the operations needed to retrieve the entities data from the services is encapsulated on the entities' implementation.

Control and Support Layer: The main modes of the application are the Map, AR and service. Map and AR modes have user interfaces while service is a background process without an UI. These interfaces interact with several support classes that provide information about the users' location, device orientation, and communication with the model layer.

The design of the Map mode UI operation is shown in Figure 4-16. The UI class has relations with several support classes; `LocationManager` class has the responsibility

to provide the information about the user's current location to the interface class. `GoogleMapService` class provides a connection to the Google Maps service that provides UI capabilities to display map information.

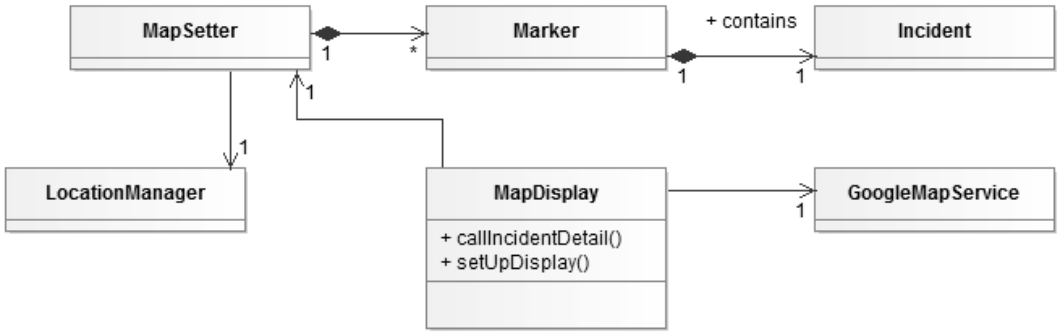

Figure 4-16: Map display – skeleton class diagram.

`MapDisplay` also holds a reference to the `MapSetter` which has the responsibility instantiate the different markers to display in the map interface. `MapSetter` uses the current filter settings defined by the user to know which instances of incident had to be created. Each marker instantiated by `MapSetter` also holds a reference to the incident that is going to be displayed in the interface.

The design for the second interface mode (Augmented Reality) follows a similar structure than the map interface design as shown on Figure 4-17. The class responsible to display the user interface is `AugementedRealityDisplay`, which has references to classes from the control layer in order to retrieve the user and device's information, and show the incident overlays to the user.

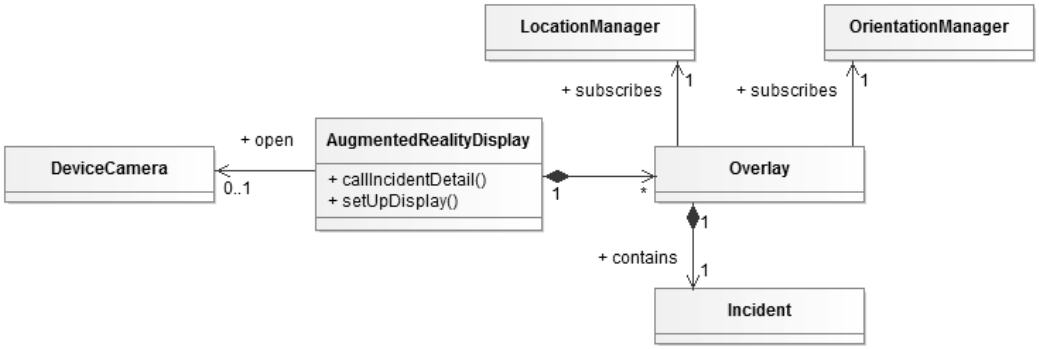

Figure 4-17: Augmented Reality display - skeleton class diagram.

The `AugmentedRealityDisplay` class uses `DeviceCamera` to retrieve a live view of the environment around the user and display it on the interface surface. This

`AugmentedRealityDisplay` also contains several instances of the `Overlay` class that represent, and store a reference to an incident around the user's location. The overlay is an image representation of an incident presented on top of the live view provided by the device's camera as described on Figure 4-9. Each overlay class has references to the `LocationManager` class which provides the user's current geographical location. It also has references to the `OrientationManager` class which senses data about the orientation and rotation angles of the device. The `OrientationManager` class uses information of the device's accelerometer, magnetic and gravitational fields to calculate the orientation of the device in a world reference frame.

The `Overlay` class uses the geographical locations of both the user and the associated incident; and the device's orientation to calculate and display the overlay image on the screen at the most accurate time and position. The entire set of overlays has references to the location and orientation managers. In order to minimize the number of class instantiations, their relationship is designed to work through an Observer pattern [59], in which single instances of the managers provide the information to the full set of overlays. The details of the observer pattern implementation are specified on the implementation chapter of the dissertation.

An additional mode of the application is based on the background processes that can execute procedures while the user is not using the application. These types of processes are called Services. Android provides an API to design background services that can process information without the need of user interaction. The application users expect that the application is able to display a notification with the nearest points of interest. Figure **4**-**18** shows the general design of the service mode.

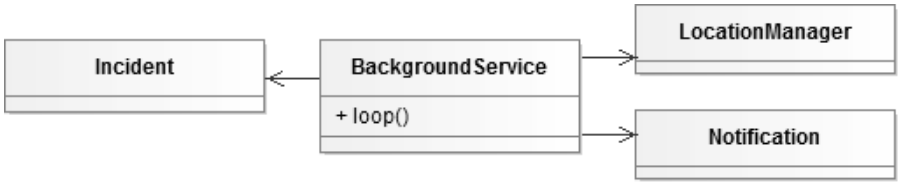

Figure 4-18**:** Service mode design - skeleton class diagram

The service class (`BackgroundService`) has the responsibility to get the user's current location and query the surrounding incidents so it can build a notification and show it to the user. The notification has several characteristics, among them the

possibility to reproduce a sound, make the device vibrate and display a light of certain colour to notify the user. The service has to constantly run on the background so it has to loop within a predetermined period of time, to review if there are any changes among the location or the incidents. If there are no changes in the location or the incidents, the service should not notify this information to the user.

The main aspects of the mobile application have been designed to provide a simple but powerful to support all the client's needs and requirements, by using a layered system architecture that separates design concerns and allows better support for future changes and improvements that may arise.

4.8 Chapter Summary

The design aspects of the project cover all the guidelines to implement a working application. The foundations of the implementation start with a requirements gathering process, an adequate selection of technology and a system design that can fulfil the user's needs and expectations. The system design covers all parts of a multi-layered architecture that supports data, service and application concerns.

Several aspects of the design have been left out, e.g. Settings classes, helper classes, operations, among others. This is made in order to focus the design of the main concerns of the application and because some of these aspects are tied closely to the specific development environment. Due to these constraints, the aspects that are not defined in this section are explained in more detail in the implementation chapter.

5. Implementation

After the overall system application and support elements were designed, the implementation phase has been developed. This chapter covers the main elements of the implementation, in which the system designs, standard practices and the development environment's guidelines are used to create a working and usable application.

For the most part, the implementation follows standard development practices, like the use of design patterns; providing each layer and element of the design the adequate software responsibility. These principles and standards, which are outlined in the General Responsibility Assignment Software Patterns (GRASP) [57] [58], improve the system's cohesion and reduces unnecessary coupling between the system components and layers. Whenever these principles are applied they will be referenced in the document.

The intent of this chapter is to cover the main technical aspects of the different application layers, focusing in the most interesting and challenging elements of the implementation. Elements such as the inner workings of standard interface widgets are out of the scope of this document. As the implementation increases the detail of the design, it is possible that some class names specified in the design change.

This chapter covers the detailed setup of the environment used to develop and test the system and the technical implementation of each of the designed system layers.

5.1 Environment setup

Taking into consideration the technology selection process described in previous chapters, a dedicated hardware environment was prepared and deployed for the implementation. The hardware and software components are shown in Figure 5-1. The database (DB) and middleware (MW) layers are deployed in a dedicated laptop. While in production environments is recommended that the DB and MW layers reside on different hardware instances/servers, during the development phase of the project this setup is enough to cover the needs and requirements of the client.

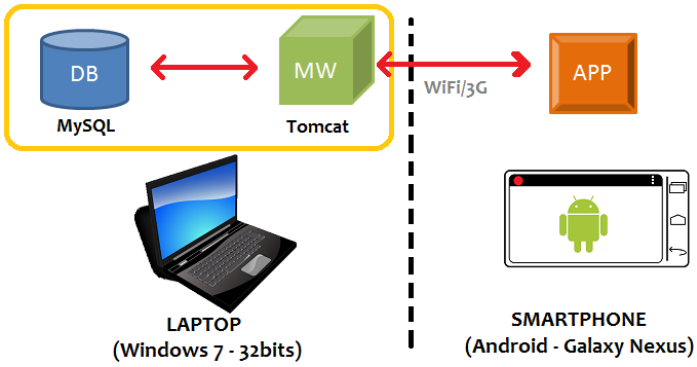

Figure 5-1: Implementation environment

The database (DB) layer is located on a MySQL instance configured specifically for the project. The middleware (MW) layer is deployed as a Tomcat application server on the same machine. Given this setup there is no need to establish a communication network between both instances, since each component can reach the other.

The application resides on an android smartphone with the Android operating system (Jelly Bean, 4.1.1). The connection between the application layer and the MW layer is supported with either Wi-Fi or the smartphone's data plan (e.g. 3G, High Speed Packet Access-HSPA, among others). To guarantee that the application and MW layers successfully connect, the MW layer should reside in a server instance accessible by a public IP address. Another option is to have both the application and MW layers reside in the same private sub network and their ports are open and visible for the client.

All the layers where developed and deployed using the Eclipse Integrated Development Environment (IDE) v. 4.2.2. To develop the MW and application layers, the Web Tools Platform (WTP) and the Android development plug-in (ADT) were downloaded and installed from the Eclipse and Google's repositories respectively [67] [68].

5.2 Database Implementation

There are several factors that have been taken into account to implement adequately the DB layer. The design chapter detailed the general database structure that stores the ported data of the system. With the database structure finalized, a process to define the actual columns and data to be ported from the original sources has been made.

The original source files contained tables with a large amount of columns. Some of the original tables have over 150 columns with several data types. The information stored in each column ranged from internal GMP fields, system-only information and data relevant to the project. Due to the fact that the system is designed to be used through mobile devices, it has been necessary to select the most appropriate type and number of fields to be used.

As stated before, the project interface relies on information based in three entities, Incident, Crime and Person. Using this information as a starting point and analyzing the tables' columns and their relationships, the most important attributes have been selected for each entity. This process consists of an analysis of each column in the entities' tables. The relevance of the column based on the client's requirements and design features have been used to decide if that column was selected for the model implementation or not. This process has been made independently for each of the three entities. Some of the attributes are based on specific columns; however some of them are be calculated fields or aggregated data from other tables.

Table 5-1 shows the names, description and examples for each selected attribute for the incident entity. The table shows two types of attributes. The first set of attributes can be viewed by the user from the application interface. Some of the attributes are retrieved from the incident table directly, while others like address are taken from the relation of the Incident table with the Address table. The second set of attributes (rows highlighted in grey in Table 5-1) is hidden from the user, but the user can use them to filter the results through the application interface.

Entity	Attribute	Description	Example - Notes
Incident	code	Identifier for the incident	**010113/0002**
	date	Date of Incident	**01/01/2013**
	premise_type	Premise type	**HOSPITAL**
	Address	Complete Address of the Incident	**1, ANONYMOUS STREET 1, UNIVERSITY, MANCHESTER, M15 6AZ, E1C3** Taken from "Address" table
	Gridref	OSRef Coordinates of the Incident	**384968395967** - first 6 digits easting coordinate, rest of digits northing coordinate.
	Opening_Desc	Opening Code of the Incident	**Alarm Central Monitoring Intruder** Taken from "Opening_code" table
	Closing_Desc	Closing Code of the Incident	**Alarm - Genuine Activation - ARREST, Public order crime, Arrested - s136 Mental Health Act** Taken From Table: "Closing_code". Incident has 3 closing codes; the app will show a concatenation of the three descriptions where available.
	child_abuse	Marker denoting an incident with child abuse	**Y/EMPTY** - Marker that will be used to filter through application settings
	domestic_violence	Marker denoting an incident with domestic violence	**Y/EMPTY** - Marker that will be used to filter through application settings
	visit_required	Marker denoting an incident that requires a visit from the GMP	**Y/EMPTY** - Marker that will be used to filter through application settings
	has_crimes	Number of crimes associated to the incident	**0,1,2… -** Calculated field with using a join to the crime table.

Table 5-1: Column selection for the Incident entity

Table 5-2 shows the names, description and examples for each selected attribute for the crime entity. This table shows the same characteristics than Table 5-1, the first set of columns are the attributes visible for the user in the interface and the second set represent the attributes in which the user can filter the data to be shown through the application modes.

Table	Attribute	Description	Example - Notes
Crime	Number	Identifier for the crime	**888892Z/13**
	Crime_type	Type of crime	**FALSE STATEMENTS ETC - FINANCIAL SERVICES ACT** Taken from table: "Crime_Type"
	Date	Date of the crime	**20/02/2013 01:30**
	Gridref	Coordinates	**384968395967** - First 6 digits easting coordinate, rest of digits northing coordinate.
	Address	Address of the Crime (Same as Incident)	**1, ANONYMOUS STREET 1, UNIVERSITY, MANCHESTER, M15 6AZ, E1C3** Taken from "Address" table
	assoc_persons	Number of persons asociated to the incident	**0,1,2…** - Calculated field with using a join to the person table.
	Drugs_influence	Filter for drugs influence in crime	**Y/EMPTY** - Marker that will be used to filter through application settings
	drugs_stolen	Filter for Stolen drugs in the crime	**Y/EMPTY** - Marker that will be used to filter through application settings
	police_assault	Filter for Police Assault	**Y/EMPTY** - Marker that will be used to filter through application settings
	alcohol_influence	Filter for Alcohol Influence	**Y/EMPTY** - Marker that will be used to filter through application settings
	guns_stolen	Filter for Guns Stolen	**Y/EMPTY** - Marker that will be used to filter through application settings
	guns_used	Filter for Guns Used	**Y/EMPTY** - Marker that will be used to filter through application settings
	domestic_violence	Filter for Domestic Violence	**Y/EMPTY** - Marker that will be used to filter through application settings

Table 5-2: Column selection for the Crime entity.

Table 5-3 shows the names, description and examples for each selected attribute for the person entity. Unlike previous entities, all the selected person attributes are visible through the user interface. There is no related filter for the person entity. This is a design decision which it's not expected to negatively impact the application features.

Table	Attribute	Description	Example - Notes
Person	Id	Identifier of the Person	**01000**
	Name	Full Name of the Person	**MRS. JESSICA LAKE** Forename, Surname and Title columns are concatenated
	Birth_date	Date of Birth	**17/02/1977**
	Address	Full Address of the Person	**1, ANONYMOUS STREET 1, UNIVERSITY, MANCHESTER, M15 6AZ, E1C3** Taken from "Address" table
	Place of Birth	Place of Birth	**BURY**
	Nationality	Nationality	**EN**
	Sex	Sex	**F/M**
	home_phone	Home Phone	**162342431**
	mobile_phone	Mobile Phone	**7450515425**
	e-mail	E-mail	**jlake@example.com**

Table 5-3: Column selection for the Person entity.

After the entities' attributes were selected, the data had to be inserted into the DBMS instance. The source data has been stored in Excel sheets, where each sheet represented a table in the database. Each sheet has then been exported to a readable format (comma separated values) to facilitate the database import process.

Each of the tables detailed on the ER diagram shown in Figure 4-3 has been created from standard "CREATE" SQL statements inside the MySQL instance. Once the tables were created, the import statement shown in Table 5-4 was used to import each the information one table at a time.

```
load data local infile 'table_name.csv'
   into table table_name
   fields terminated by ','
   lines terminated by '\n'
(columnName1, columnName2, . . . columnNameX)
```

Table 5-4: Import statement from files to the database instance.

Special care had to be taken with date fields, since the source data had dates stored in different formats. Once the import process was finished, indexes were created on several attributes (columns like identifiers, dates, coordinates, among others).

With this process, the main database implementation has been finalized, and the Database layer is ready to accept requests from the Middleware layer.

5.3 Middleware Layer

This section covers the implementation details of the most important technical components of the MW layer.

In order for the mobile application to be able to query the data stored in the database, the MW layer has been implemented following the design guidelines specified in previous sections. The MW layer design allows the layer to query and connect to additional sources of information other than the local GMP database; however an actual implementation of such capabilities is out of the scope of the project.

The MW layer follows the design defined on Figure 4-4. Basedi the attributes and entities specified on the DB layer implementation, a series of services where implemented to query and process the stored data. Table 5-5 shows the list of services that provide communication between the application layer and the DB Layer.

Name	Description - Information Returned	Parameters
ListIncidents	List of Incidents and their attributes. Parameters are the filter attributes applied to the query.	**Incident:** - Has crimes - Child Abuse - Domestic Violence - Visit Required - Incident Coordinates **Related Crimes:** - Domestic Violence - Alcohol Influence - Drugs Influence - Drugs Stolen - Guns Used - Guns Stolen - Police Assault
Crime	Attributes of a single crime	Crime ID
CrimesFromIncident	List of crimes and their attributes associated to a single Incident	Incident ID
Person	Attributes of a single person	Person ID
PersonsFromCrime	List of Persons and their attributes associated to a single Crime	Crime ID
Incident	Attributes of a single Incident	Incident ID

Table 5-5: Services description of MW layer

Each of the 6 services serves a single purpose, which is to query either single entities or a list of them based on a set of parameters. The services that retrieve the attributes of a single entity receive a single parameter, which is the entity's ID. The services that retrieve a list of entities receive the ID of the grouping entity, e.g. **CrimesFromIncident** gets a list of crimes based on their relation to a single incident. **ListIncidents** is the

most complex service of the group; it is also the most important. This service is used by all modes in the application to retrieve the list of incidents based on the filter specified by the user in the application interface. The service has parameters for Incident and crimes. This is designed to give a greater possibility to the user to filter the number of possible results in any of the applications modes.

The services are consumed through an Http request made by the client application. To call a service, the http request follows the following Uniform Resource Identifier (URI):

```
http://ServerName:port/ServiceName?Parameter1=value&Parameter2=value
```

Any client can query the results by making a well formed http request to any of the services and the MW instance can be reachable through the network by the client.

Figure 5-2 shows a partial implemented structure of the MW layer with three of the six services. It shows the `Crime`, `CrimesFromIncident` and `ListIncidents` services (the rest of the services follow the same pattern). Each service implements the `HttpServlet` abstract class that provides the Http request methods used by the client applications.

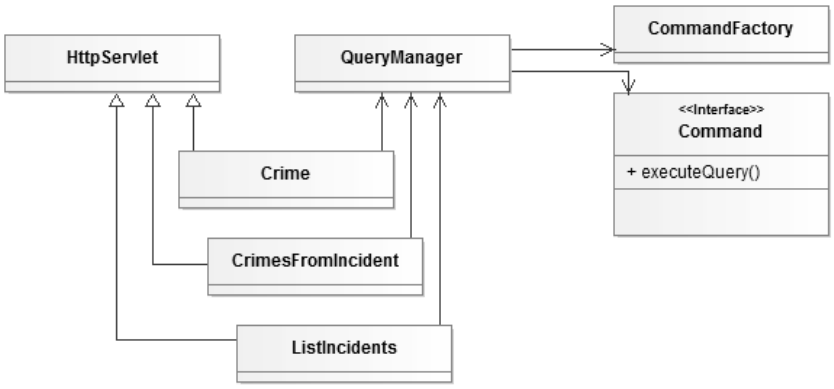

Figure 5-2: Service Implementation example

All services represent a point of entry to the MW layer for the clients. Each service has a reference to the `QueryManager` class which manages the execution of each class that is responsible for the actual data query and manipulation. Query manager has a reference to a `Command` Interface and relies on the `CommandFactory` to instantiate the appropriate class to be executed.

This Implementation follows the “Command” structural design pattern [58], on which the `QueryManager` class only executes the `executeQuery` method in the interface

(implementation shown in Table 5-6). Using indirection and polymorphism (in this case by referencing an interface rather than concrete implementations), allows the system to separate the implementation classes from the classes that use them. This makes the system easier to maintain and protects the implementation from future variations, shielding the service class from the changes made on the class that performs data processing.

```
public interface Command {
      public String executeQuery();
}
```

Table 5-6: Command interface definition.

As shown of Table 5-6, the `executeQuery` method on the command interface returns a string. This string is the result of the process executed by the implementation of the data manipulation class. The result is then displayed by the service methods.

Figure 5-3 shows the internal implementation of the data manipulation classes being abstracted through the command pattern. The data manipulation classes (`CrimeCommand`, `CrimesFromIncidentCommand` and `ListIncidentsCommand` in the figure) implement the `Command` interface. All classes implementing the interface are referenced by the `CommandFactory` class. `CommandFactory` is responsible to return concrete instances of the data manipulation classes. This class is a singleton which is a class that implements the "Singleton" creational design pattern [57]. A singleton has only one instance in the system.

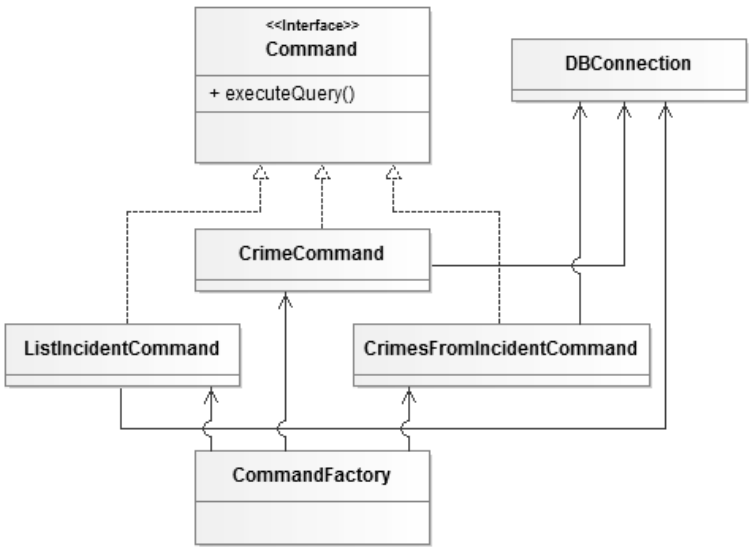

Figure 5-3: Command implementation on MW Layer.

Each data manipulation class has the responsibility to connect to the DB layer, query the specific data and return the result string. The project implementation only deals with the DB layer connection through the `DBConnection` class. It is possible that one of these classes, instead of querying the DB layer could connect to other databases, consume services in other services and manipulate other types of data. Each data manipulation class only needs to implement the command interface and return a result string with the appropriate format (as described on the design chapter).

Each data manipulation class uses a SQL query to retrieve the information from the database. To illustrate the type of queries used in the implementation, Table 5-7 shows an example of such query (for the Person service) and Table 5-8 shows the service's text output for that specific query.

```
SELECT p.id, CONCAT_WS(' ',p.title,p.forename,p.surname) as full_name,
p.birthdate, p.place_of_birth, a.description as addressName,
p.nationality, p.sex, p.home_phone, p.mobile_phone, p.e_mail
FROM PERSON p
      left join person_address pa
            left join address a
            on pa.address_id = a.id
      on pa.person_id = p.id
WHERE p.id = ?

-- Id taken from the request parameters sent to the service
```

Table 5-7: Query example used by service.

```
01529;MISS JOSIE DELTA;1978-02-16;OLDHAM;67, ANONYMOUS STREET 49,
UNIVERSITY, MANCHESTER, M15 6AZ, E1C3;EN;F;;;;
```

Table 5-8: Example of service text output.

The MW design and implementation provides a single point of contact for the clients and provide the structures to extend the services capabilities to consume other information sources. With the MW and DB layer implementation and data design implemented, the following step is to provide the application layer implementation for data visualisation and processing.

5.4 Application Layer

This layer has the responsibility to query, process and display the most relevant data to the user in a variety of interfaces. To present a compelling, efficient and simple experience to the user the application has to take into consideration the type and quantity of information provided by the MW layer. The application also has to take into consideration the characteristics of the mobile medium in which it resides. By taking advantage of location and map services, and the device's features like sensors and video camera, the application helps transform simple data into a more relevant user's experience and interaction.

This section describes the technical characteristics of the layer implementation. It contains a description of the overall application features, including a review of the android user interface framework. The section also focuses in the 3 main operation modes of the application, and their interaction with the supporting class structures that provide necessary methods and operations to retrieve the necessary information.

5.4.1 Device features

The client device used for the implementation is a Samsung Galaxy Nexus, which runs a stock android operating system. The device has several characteristics which makes it suitable for implementing a variety of different user interfaces. Among these characteristics, there is a video camera that is used to take pictures video, or just for displaying a view to the outside world. Other features include a vast array of sensors that can measure a lot of different environment variables around the user and device.

For the application implementation, especially the 2 main modes of operation, it is necessary to calculate the orientation of the device over a reference frame using the sensors in the device, retrieve the user location and to and access the device's camera. A general overview of each feature is described below.

Sensor Management: Table 5-9 shows the list of available sensors available to measure environmental information, some of which will be used to calculate the device orientation.

From this list, the main sensors that are going to be used are the magnetic field, gyroscope, accelerometer and the rotation vector. A more detailed analysis of each one of these sensors is specified on the AR mode section of this chapter. In order to use any of the sensors, it is needed to use the sensor architecture defined by the android operating system. Figure 5-4 shows the standard android sensor architecture.

Sensor Type	Description
TYPE_ACCELEROMETER	A constant describing an accelerometer sensor type.
TYPE_ALL	A constant describing all sensor types.
TYPE_AMBIENT_TEMPERATURE	A constant describing an ambient temperature sensor type
TYPE_GRAVITY	A constant describing a gravity sensor type.
TYPE_GYROSCOPE	A constant describing a gyroscope sensor type
TYPE_LIGHT	A constant describing a light sensor type.
TYPE_LINEAR_ACCELERATION	A constant describing a linear acceleration sensor type.
TYPE_MAGNETIC_FIELD	A constant describing a magnetic field sensor type.
TYPE_ORIENTATION	This constant was deprecated in API level 8. use SensorManager.getOrientation() instead.
TYPE_PRESSURE	A constant describing a pressure sensor type
TYPE_PROXIMITY	A constant describing a proximity sensor type.
TYPE_RELATIVE_HUMIDITY	A constant describing a relative humidity sensor type.
TYPE_ROTATION_VECTOR	A constant describing a rotation vector sensor type.
TYPE_TEMPERATURE	This constant was deprecated in API level 14. use Sensor.TYPE_AMBIENT_TEMPERATURE instead.

Table 5-9: Types of sensors available on Android OS **[69]**.

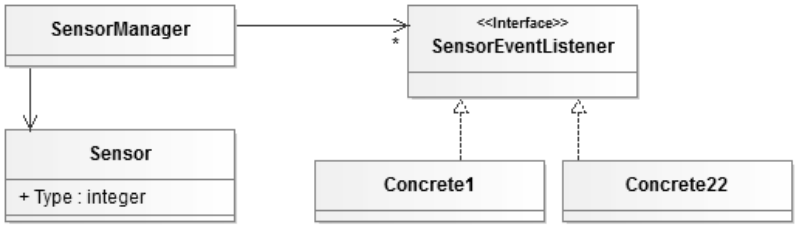

Figure 5-4: Android sensor architecture.

The sensor architecture is defined by several classes. The main class is `SensorManager` of the `android.hardware` package. This class is instantiated by the Android's system service with the instruction:

```
mSensorManager = (SensorManager)
        act.getSystemService(Context.SENSOR_SERVICE);
```

Where the *`mSensorManager`* object is an instance of `SensorManager` class. After `SensorManager` is correctly instantiated, it is possible to access its methods.

Most sensors hold information on 3 available axes (x,y,z). To retrieve the information on each sensor, `SensorManager` can register an instance of a `SensorEventListener`. This interface is implemented on the developer's code to receive the sensors values on the `onSensorChanged` method of the interface (On Figure 5-4, the implementation is reflected on the concrete classes). A simple implementation of the interface to retrieve the sensors values is shown in Table 5-10:

```
public class SensorListener implements SensorEventListener{

	private String x,y,z;

	@Override
	public void onSensorChanged(SensorEvent event) {
		// "values" holds the sensor's value in each of
		//the 3 slots of the array
		float[] values = event.values;
		x = values[0] + " Value 1 - X axis";
		y = values[1] + " Value 2 - Y axis";
		z = values[2] + " Value 3 - Z axis";
		//do calculations with x, y and z variables
	}
```

Table 5-10: Basic SensorEventListener implementation.

The listener registration is made by using the SensorManager methods:

```
sensorSpec = mSensorManager.getDefaultSensor(Sensor.SENSOR_TYPE);

mSensorManager.registerListener(listener, sensorSpec,
        SensorManager.SENSOR_DELAY_NORMAL);
```

sensorSpec is an instance of class Sensor and it is instantiated using `SensorManager's` method of `getDefaultSensor` (in an Android device there could be several sensors of the same type). This method receives a "sensor type" parameter, accessed statically from the Sensor class. Table 5-9 shows the types of available sensors that a device could provide. After instantiating the object from `Sensor` class, the implemented `SensorEventListener` is registered through `SensorManager's registerListener` method.

Location retrieval: The device is able to retrieve the user's current location in a global coordinate system. To provide said information, the device uses 2 types of sensors, Global Positioning Service (GPS) and Network Location. The GPS sensor is a much more detailed positioning sensor that Network location sensor, however, if the GPS

sensor is not available on the device or if the user is not reachable by the satellite system used by the GPS, the network positioning could replace the functionality (reducing precision). Figure 5-5 shows the location management architecture on Android.

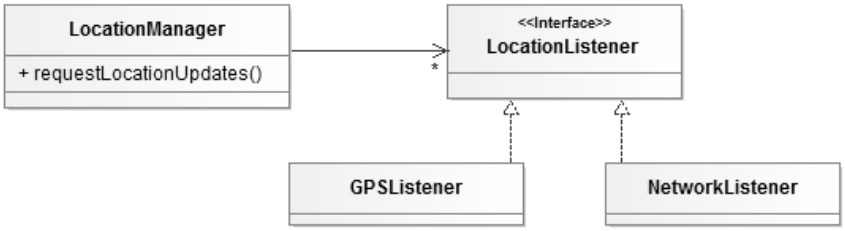

Figure 5-5: Android location architecture.

To read the position information on the device, 2 classes are used on the application, `LocationManager` and the interface `LocationListener` from the package `android.location`. `LocationManager`, as the name implies, manages for the user the sensor information. To update the sensor information, the class receives implementations of the `LocationListener` class which will receive the updates on the method `requestLocationUpdates`. An example of how to register location listeners is shown in table 5-11.

```
LocationObserver obsNet, obsGPS;
...
LocationManager locationMan = (LocationManager)
        this.getSystemService(Context.LOCATION_SERVICE);
...
locationMan.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,0,0, obsNet);
locationMan.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, obsGPS);
```

Table 5-11: Location registration example.

First, the `LocationManager` class is instantiated by the System Service of the Android OS. Then, through the `requestLocationUpdates` method of `LocationManager`, specific implementations of the `LocationObserver` interface are given as parameters for the method. On the method it is specified which kind of sensor (GPS or Network) is desired. On this example `LocationObserver` is an implementation of the `LocationListener` class. There are several methods that can be implemented of the `LocationListener` class. The most important is `onLocationChanged` which receives a parameter of the class `Location` which holds the latitude and longitude of the device's position from the `LocationManager`.

Table 5-12 shows a basic implementation of the `LocationListener` class, where this method is called every time the location of the device changes.

```
@Override
	public void onLocationChanged(Location newLocation) {
		if(isBetterLocation(newLocation,lastLocation)){
			lastLocation = newLocation;
		}
	}
```

Table 5-12: Retrieval of new Locations in LocationListener.

This implementation uses a function (not on table) that calculates if the new location received is better than the previous one (using accuracy and time of the measurement) and if that's the case, it is the last location is changed to the new one.

Camera usage: The Android API provides a functionality to control the camera images through several software classes and operations. Figure 5-6 shows the camera classes used to retrieve the camera images from the physical device. For this project, functionality such as capturing video or photos is not necessary, so their classes and methods are not described in this section.

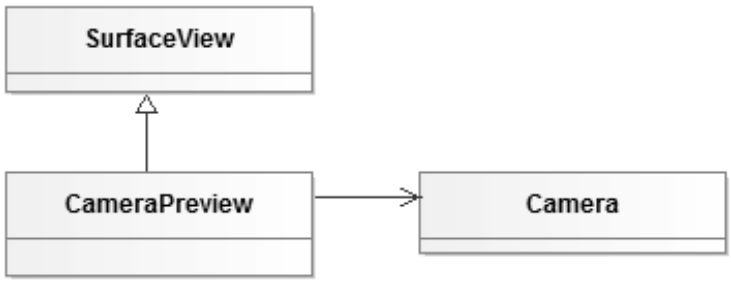

Figure 5-6: Camera classes in Android.

The `Camera` class has the responsibility to communicate to the device hardware, including opening the shutter and retrieve the images from the camera's sensor. `CameraPreview` has the responsibility to display the camera's image frames on the device screen. It inherits the methods and functionality of an abstract android class (`SurfaceView`). Using a `SurfaceView` on the android UI allows the possibility to locate additional images and overlays on top of the video feed.

Table 5-13 shows how to create and instantiate the camera on the device. The camera is a restricted element in the device, so special care has to be taken to retrieve the instance (only one application can use the camera at the same time).

```
Camera cam;
CameraPreview preview;


...
//open camera and assign to View
cam = Camera.open();
preview = new CameraPreview(cam);
```

Table 5-13: Camera instantiation and usage.

A `Camera` instance can only be created through a static method of `Camera`, so every application has to use the same method.

The mobile application won't process any information received from the camera, so there is no need to do more than just create the `CameraPreview` and adding that preview to the user interface. Adding `CameraPreviw` to the android interface allows displaying a live view of the world without the need to create a support framework to process and display the images from scratch.

Permissions: To use the different features of the device on the application code, it is necessary to request the appropriate permissions to the Android OS. This has two main functions; the first is reason it's to inform the OS, so it knows what applications are using its resources. The second is to display to the user (at install time) the resources that the application needs to operate adequately. If the user doesn't agree to the resources being requested, the application won't be installed by the OS.

Most of the features need a specific permission to be requested. The permissions requested for the application are shown on Table 5-14. These permissions are specified on the `AndroidManifest.xml` file stored on the root folder of the application source files.

The following resources are being requested in the permission statement: Use of Google Maps service, the device's camera, data connectivity (Wi-Fi, 3G), storage writing, Google services for compatibility, network location services, GPS location services, network status, receive signals from OS boot, and the possibility to for the device to vibrate.

```
<uses-permission
      android:name=" uk.ac.man.gmp.full.permission.MAPS_RECEIVE" />
<uses-permission
      android:name="android.permission.CAMERA" />
<uses-permission
      android:name="android.permission.INTERNET" />
<uses-permission
      android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
      android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission
      android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission
      android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
      android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
      android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission
      android:name="android.permission.VIBRATE" />
```

Table 5-14: Permission specification for the application.

Location management: By default, the android OS manages the user location based in the global coordinate system commonly used by a GPS. Based on previous information, the data provided by the GMP formats its data in the OSRef coordinate system. In order to be able to use the geographical information provided by the GMP it was necessary to transform it into the desired coordinate system.

The OSRef coordinate system uses coordinates based on a grid that divides the UK in squares with an accuracy of meters. To convert the coordinates a public java library licenced by the GPL (General Public Licence) was utilized. This library was created by Jonathan Stott and it is available in [70].

The library provides several classes that can be used to convert from one coordinate system to another. To provide flexibility to different clients, the conversion process was not executed on the MW layer, but it is delegated to the clients. Table 5-15 shows how to convert an OSRef coordinate to a variable that can be used in android.

```
private static LatLng convertToLatLong(double easting, double northing){

      OSRef osr = new OSRef(easting,northing);
      uk.me.jstott.jcoord.LatLng latlong1 = osr.toLatLng();
      latlong1.toWGS84();
      return new LatLng(latlong1.getLat(),latlong1.getLng());
}
```

Table 5-15: Conversion from OSRef to Latitude and Longitude.

An `OSRef` attribute is created with easting and northing coordinates. The attribute can convert to a Latitude and Longitude attribute in one method. The global coordinate system can work with different reference frames. The GPS and android systems use the WGS84 reference frames, so the `LatLng` variable has to be formatted with that reference frame. Once is converted a new `LatLng` attribute is created. The conversion library uses the same name for the `LatLng` class used by android. Due to this, it is necessary to name the library's `Latlng` attribute with its full signature name so the compiler can understand which library a specific object is referencing. The returned value is an instance of the `LatLng` internal android coordinate system.

5.4.2 UI overview

The android OS has a special way to create and display the user interface. Any user interface on android works as a group of two different elements, Activity and Layout as seen on Figure 5-7.

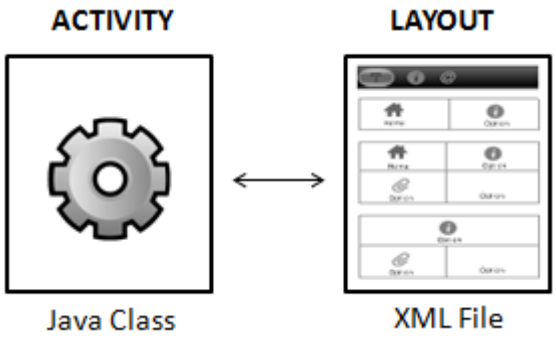

Figure 5-7: Android interface files

An Activity is a java class that provides the programmability support to user interactions with the UI. Through the Activity the system receives the user actions, gestures; and controls the application lifecycle and control flow.

The layout is an XML that defines a specific UI in terms of "views" which are graphic elements (e.g. icons, text fields, buttons, graphics, etc.) displayed on the screen. Each UI element is called a **View** and share descriptive attributes like type, background colour, size, weight, among others.

There are some special views called **ViewGroups** as shown on Figure 5-8. ViewGroups are special kinds of Views that can be composed of other Views (including other ViewGroups).

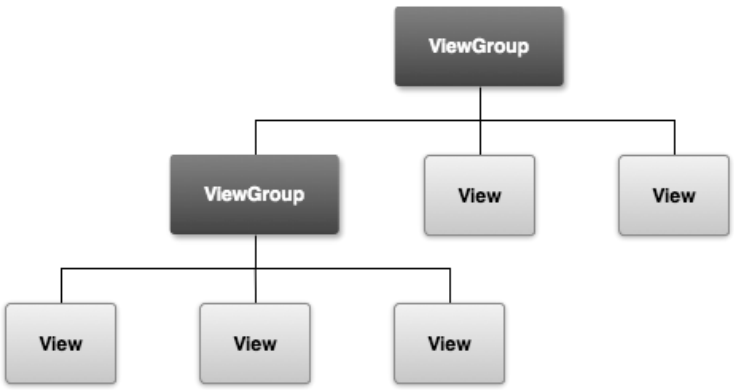

Figure 5-8: View hierarchy on the Android interface **[71]**.

Activities on the Android development environment have a lifecycle that must be understood to properly manage the application resources. Figure 5-9 shows all the states and transitions on each activity lifecycle.

The first state is *created*, which is the state that an application resides after being first instantiated by the system. In this state, the basic application operation logic is done only once during the activity lifecycle. The second state is started, which is the state where the activity is visible for the user. *Resumed* is the main state of the application, and receives external input from it. Paused is the state where the activity resides once another activity partially obscures the main activity. In this state operation should be made to ensure that only the main resources of the device are kept running. Stopped is the state where another activity is running and being displayed to the user. In this state the application releases all the resources that are being used by the activity and saves all the user information that needs to be stored. The last state is destroyed, and this represents that the activity has been released from the device's memory.

To reach each of the states, the operating system calls certain methods on the activity class. These methods (`onCreate`(), on`Start`(), on`Resume`(), `onPause`(), `onStop`(), `onDestroy`()) represent the transitions between the states. These methods can be overloaded to ensure that specific user operations can be made on the activity during each transition.

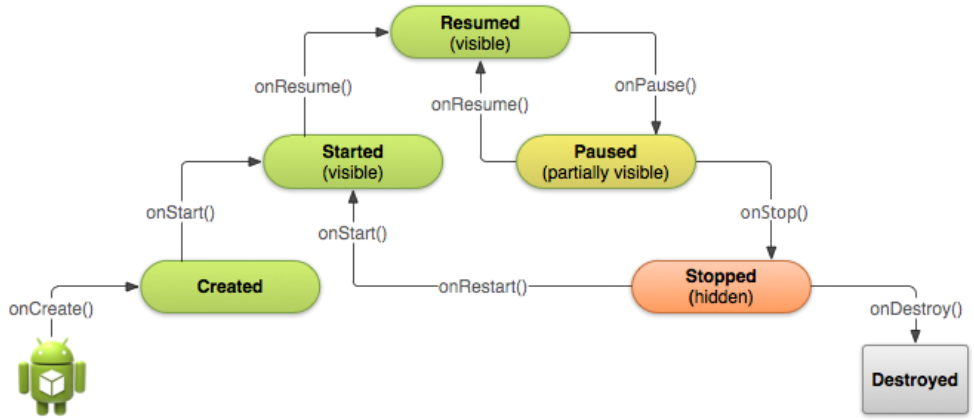

Figure 5-9: States and transitions of the activity lifecycle **[72]**.

While the user interface is usually designed on the layout file, android allows the creation and instantiation of views dynamically. Creation of buttons, images, and other types of widgets can be instantiated and added to the static layout during the lifecycle of the activity.

5.4.3 Internal application logic and common package structures

This section describes the overall application architecture defined for the mobile environment. It will describe the application lifecycle, including most of the components and their interactions in a global perspective. Most of the structure of the implementation follows the design specified in the previous chapter and the guidelines expressed by GRASP principles.

Application logic: The implementation made has been structured in several packages as shown on Figure 5-10. Each package holds a responsibility within the system and the package structure allows separation of different concerns within the application. The UI package has a relationship with the control packages of the three main modes of operation (Map, AR, and Service). It also has relationship with the settings classes which holds all the user options and application attributes. The application settings are directly updated through the UI using a series of classes in the settings package which loads and saves each defined attribute in the android storage system.

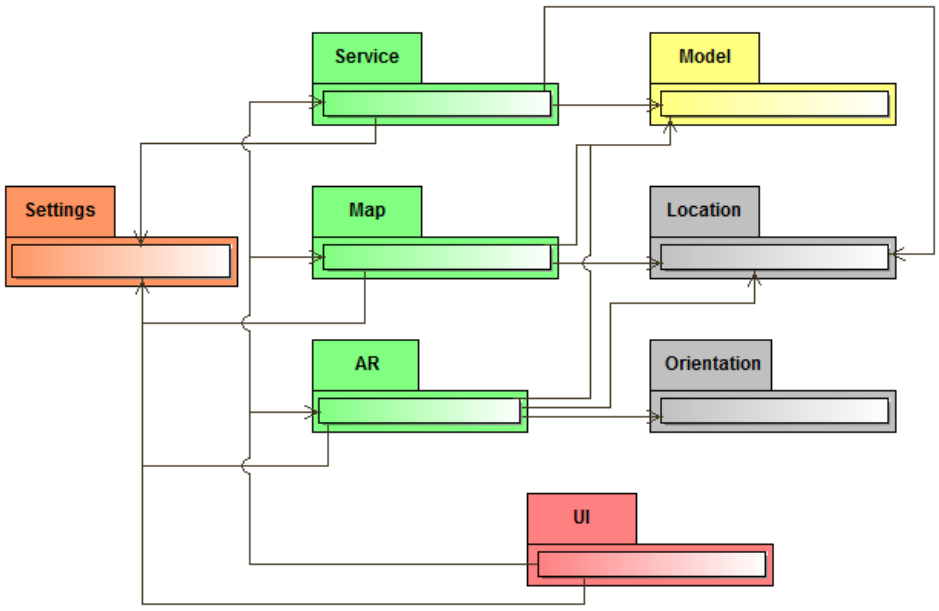

Figure 5-10: Package structure for the mobile application.

The three modes of operation connect to the settings package to retrieve application and filter settings. They also use support packages (Location and Orientation) to control the user and device information. All three services have a relation with the model to retrieve the entities information from the services provided by the MW layer.

Figure 5-11 shows the general interaction between the different application packages. While specific classes are not represented into the diagram, it shows the process and sequence of each package interaction.

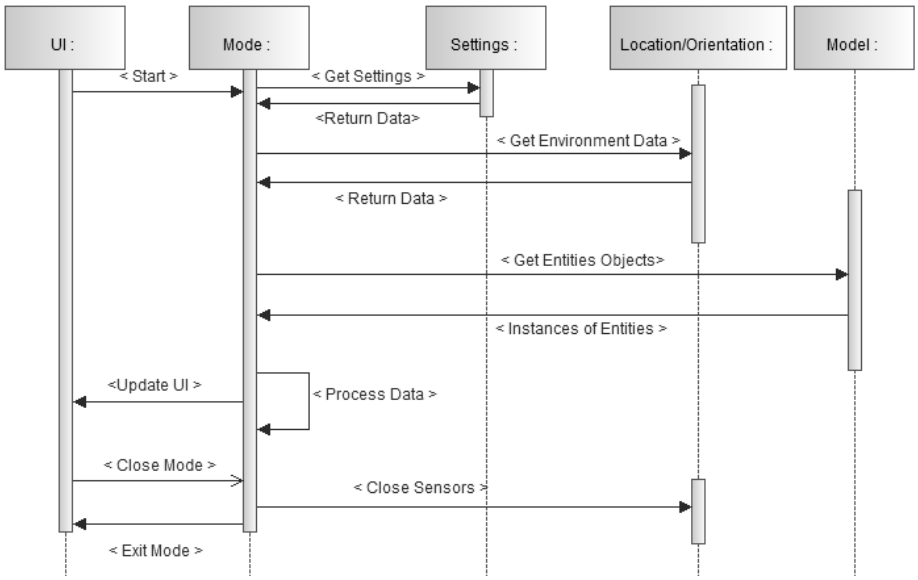

Figure 5-11: Interaction between application packages.

By user selection, the UI starts one of the three modes. Each mode will in turn retrieve settings attributes, environmental information (device location and orientation) and

the entities' objects from the model, using the filters specified by the user in the settings. The model processes the information and updates the UI with the data to display. It is possible that the environmental information changes over time (e.g. user changes its position, device turns to another direction). This information is updated to the mode once it occurs. Once the user selects to change its mode or to go to another interface within the application, the UI updates interface mode. The mode in turn closes the sensors to reduce battery usage and closes the mode. While each mode may have different ways to display the data, most of the general interaction between the packages during the application operation follows the same pattern described in Figure 5-11.

Settings Management: The settings package follows the structure specified on Figure 5-12. Each of the settings managers can load and store the specific attributes related to its function.

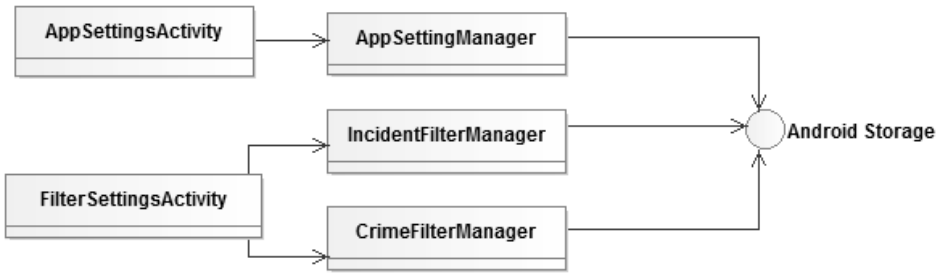

Figure 5-12: Settings package structure.

To store and load data, each manager class has a reference to the android classes `SharedPreferences` and `SharedPreferences.Editor`. Each class has the responsibility to load and store attributes respectively. The mechanism to instantiate, save and load attributes is described on Table 5-16. Each manager instantiates each storage class (`SharedPreferences.Editor`) and each class can be reused to load and store each attribute. The attributes that can be stored using this mechanism are the native types of the java language (string, integer, long and float).

```
SharedPreferences sharedPref = context.getSharedPreferences(
            shared_preference_file, Context.MODE_PRIVATE);
SharedPreferences.Editor  editor = sharedPref.edit();

   ...
//Mechanism to load an attribute from storage
String valueToLoad = sharedPref.getString(keyAttribute,defaultValue);

   ...
//Mechanism to save an attribute from storage
editor.putString(keyAttribute, valueToStore);
```

Table 5-16: Settings load and store mechanisms.

Each settings manager is a Singleton [57], which means it only has an instance on the system. A basic implementation of the Singleton pattern is shown on Table 5-17. The settings managers need to be created as singletons so different modes of the application can load and store the information using a single instance of the manager. Each manager also has their load and store methods defined with the synchronized keyword to make them thread safe.

```
public class SingletonClass {

      private static SingletonClass instance;

      //The object instance is returned statically
      public static synchronized SingletonClass getInstance(){
            if(instance == null){
                  instance = new SingletonClass();
            }
            return instance;
      }

      //Constructor is private
      private SingletonClass (){
            //Class creation code
      }

      //Methods declaration.
      ...
}
```

Table 5-17: Singleton Pattern scheme for Settings classes.

Model structure and persistence management: The model representing the application's core structures is closely implemented based in the design specification. It's worth noting that the model package has the responsibility to provide persistence for the information stored in the database. It also has the responsibility to consume the services provided by the GMP to retrieve the database information. A class representing each of the entities was created as shown on Figure 5-13. Each class has

a series of specific attributes that identifies and differences them from the others. It is important to note that the different entity's objects are not instantiated by any class outside the Model package. The classes are instantiated internally because the information to create each instance is retrieved from the MW layer's services.

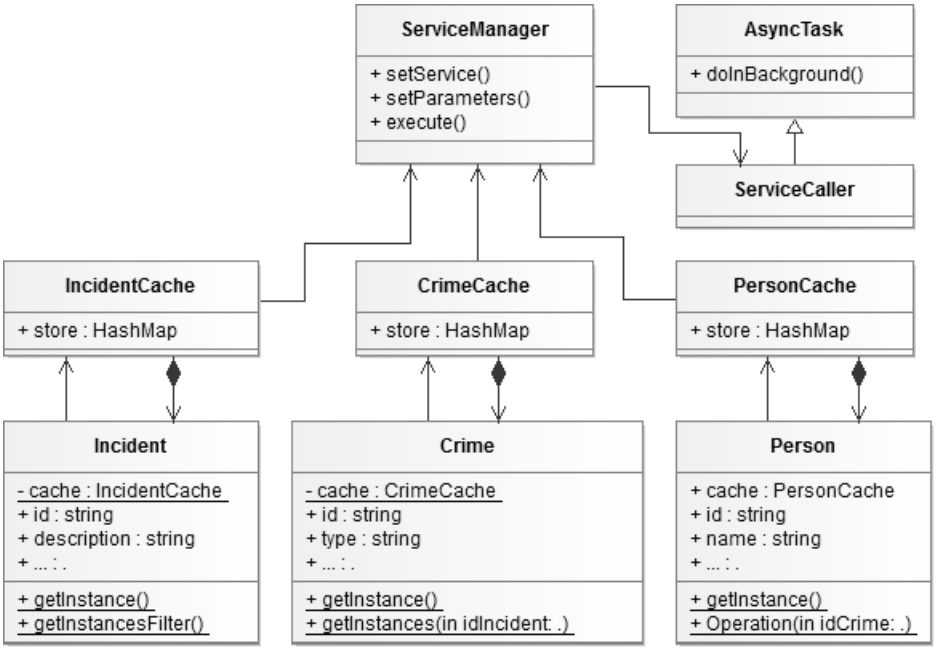

Figure 5-13: Model Package's class structure.

Each entity class has static methods to retrieve an instance or an array of instances of each type. Incident has methods to retrieve a single instance (using the incident id as a parameter) and an array of instances based on the filter specified by the user on the application settings. Crime has methods to retrieve an array of instances with the associated incident id as a parameter. The same methodology is followed by the Person entity. It should be noted that the services provided by the MW layer are based on these same methods.

Each entity class has a reference to a Cache class that has the responsibility to store in memory the instances of loaded entity classes. If the specific entity is not stored on the cache, it uses the services provided by the `ServiceManager`, a singleton class, which has the responsibility to prepare the data to consume the MW services. This class delegates to `ServiceCaller` the actual consumption of the service. Service caller has a hierarchical relationship with the android class `AsyncTask`. This class is a thread that runs in its own process. `ServiceCaller` is implemented in this pattern due to the fact that service consumption can be done in a background process.

Consuming the services using these means improves the application responsiveness and allows the OS to manage its internal resources more adequately by way of process prioritization. `ServiceCaller` uses the application settings class to retrieve the server address attribute in order to connect to the MW layer.

5.4.4 Map mode

The map mode is implemented using the Google maps component on the application. The component, named `GoogleMap`, is a fragment on the Android development environment. Fragments are a portion of a user interface that can be included on an application activity, which is the main class that interact with the user on a given time.

To create the map mode in the application, an Activity to control the UI was created, and the map fragment was initialized on the `onCreate`() method. The GoogleMap Fragment holds all the functionality to zoom the map and navigate through the desired position (configured on the instantiation phase).

To add functionality to the map, it is possible to add markers on the map at run time. These markers can be configured to be displayed and to respond to user actions once touched. This responsibility is taken by the `MapSetter` class defined in the design specification. There are 3 types of markers implemented to display in the map, as shown on Figure 5-14.

Figure 5-14: Marker specification for Map display

The yellow marker represents an incident that doesn't have associated crimes on it. The red marker represents an incident that has crimes associated. The blue marker represents the user's location. As described on the design, once the user touches one of the markers, a small description of the incident is shown. An example of the description window is shown on Figure 5-15.

Figure 5-15: Message description of incident on Map mode.

To display the description window, a special class was implemented called `WindowAdapter`. This class has been included into the map mode design as shown on Figure 5-16.

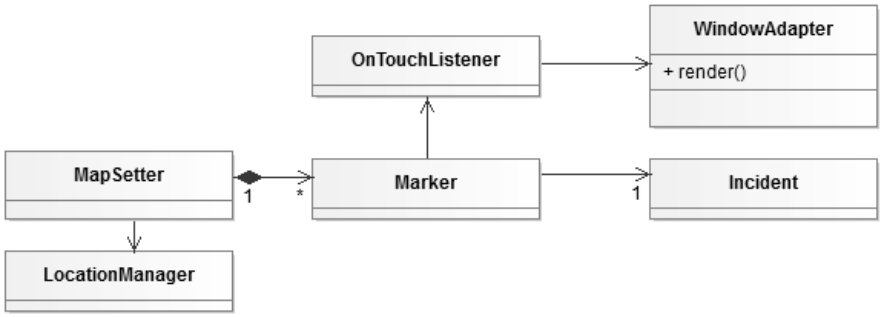

Figure 5-16: Modified Map mode design.

Each android element on a UI has the possibility to use a listener for user interactions. The listeners are classes that that receive information about different user gestures (e.g touch, pinch, zoom, among others) on the screen. By implementing the listeners it is possible to receive user input and execute actions according to them. A listener was implemented (`OnTouchListener`) that generates a `WindowAdapter` class that has the responsibility to render the information on the screen. The `WindowAdapter` class is also a UI element, so it can also have its own Listener. This listener is then used to exit the map mode and display the incident specific data in the UI on a different class as described on the UI specification. Figure **5-17** shows the interaction between the different elements on the Map mode.

The user starts the Map mode through the `MapActivity` class. Map Activity uses the `MapSetter` which loads the marker using the Model information from the Model Package. Once the user touch a specific marker, the `WindowAdapter` is used to render the specific incident description on the window view on the map. If the user touches the window description the window adapter loads the next activity (Incident Detail interface) and closes the current mode represented by the `MapActivity` class.

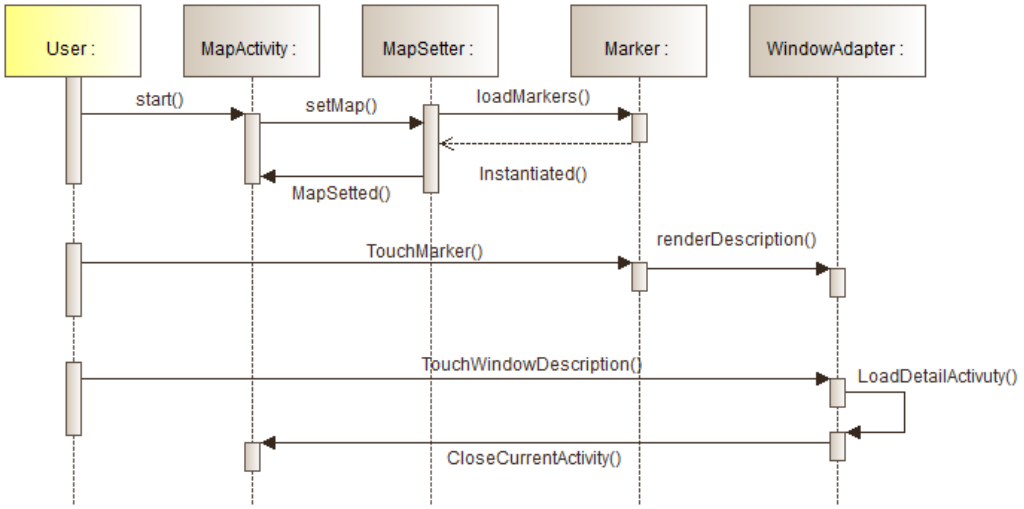

Figure 5-17: Interaction between Map mode components.

Most of the functionality is provided by the Google services, so the most important aspect of the Map mode is to accurately query the incidents based on the user-specified filter settings. Figure 5-18 shows the user interface with a series of markers representing incidents. One of the markers is being touched and it is displaying information about the specific incident.

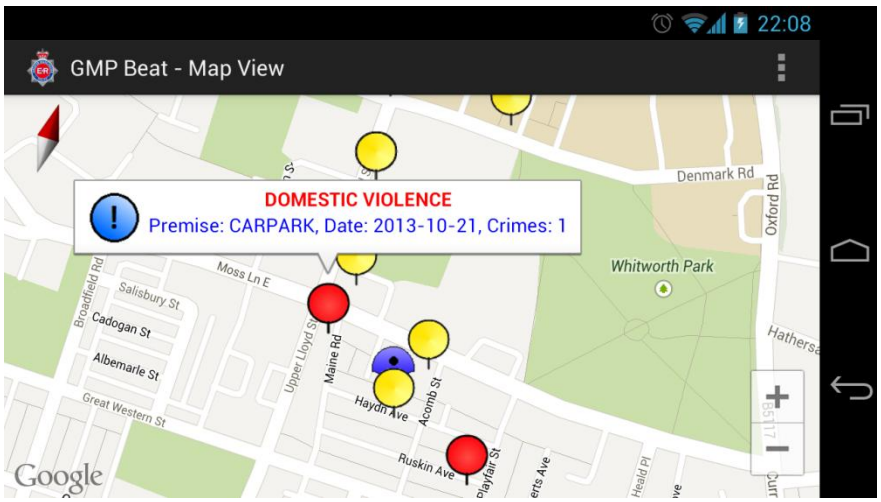

Figure 5-18: Screenshot of Map mode Implementation.

5.4.5 AR mode

To create an augmented reality mode, it is necessary to access the device's camera and display its images to the user. On top of the video feed, location sensitive information is displayed. The information is represented as image overlays. Each overlay has the functionality to display information after a user touches it, showing a small description of the data represented by it.

The image overlays have to move according to the device's location and the user's movement and orientation. There are several aspects that have to be taken into consideration to create a realistic experience using the AR mode:

- The change in orientation of the display/device in relation to the earth's North Pole.
- The relative location between the user device and points of interest. In this case, the points reference incidents queried using the filter settings specified by the user.
- The angle of vision through the user camera. This is used to understand which elements should be visible through the device display.

The orientation of the device is defined in three angles of rotation, Azimuth, Pitch and Roll as shown on Figure 5-19.

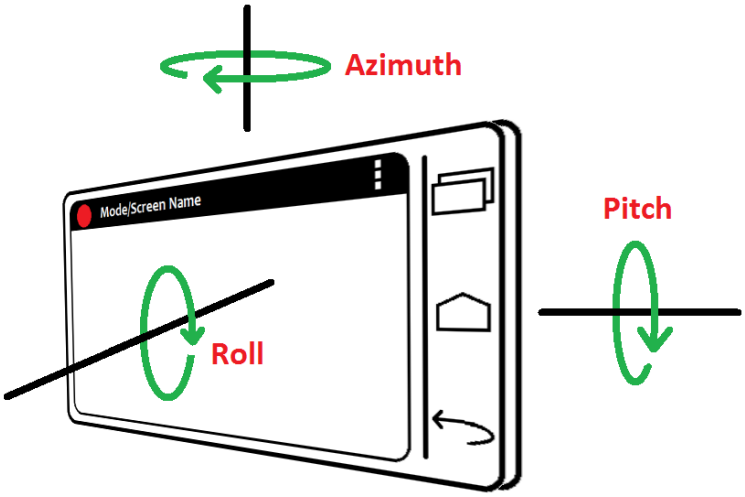

Figure 5-19: Rotation angles of the device.

Azimuth rotation is based on the lateral rotation of the device and is measured as the rotation relative to the North Pole on a world reference frame. Pitch is the rotation of the device on an axis parallel to the screen plane (up and down rotation). Roll is the rotation of the device along the axis that goes perpendicular to the screen plane.

The responsibility of retrieving the orientation data resides on the orientation package. The implementation of the package follows the structure devised on Figure 5-20. The `OrientatonManager` class allows to registering and unregister the listener for getting the sensor data. The sensor data is retrieved by a Listener class that receives information from the different sensors on the device.

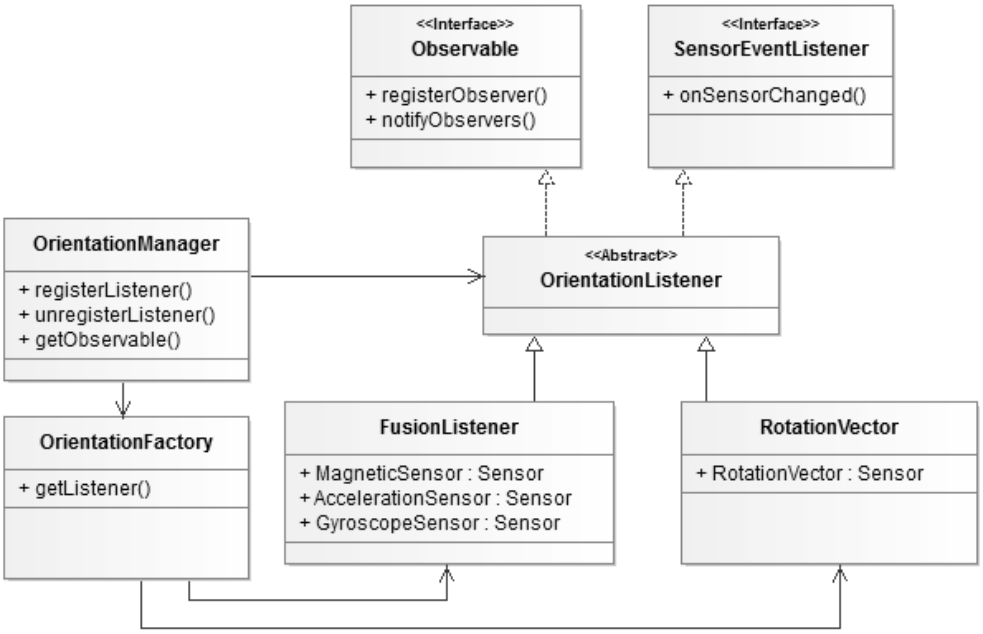

Figure 5-20: Orientation manager structure.

The orientation data (angles of rotation on each axis) can be obtained from the sensors following two different procedures. The first one is based on a fusion of the data retrieved from the Accelerometer, Magnetic Field and Gyroscope Sensors. The other procedure is based on an internal sensor called Rotation Vector. Both methods are explained below.

Fusion Listener: The idea of the Fusion Listener is to combine the information of the Accelerometer (measures acceleration of the device on each axis accounting including gravity), Magnetic Field (measures the earth's magnetic field around the axis) and the gyroscope which measures the angular velocity over each axis. Android provides a rotation matrix based on the Accelerometer and Magnetic Field sensors called `SensorManager.getOrientation()`; however these two sensors measure a lot of noise, specially the magnetic sensor which is sensible to magnetic fields other than the earth's own magnetic field. The gyroscope has more accurate information, but to get the orientation angles, the information has to be integrated over time. Minimal variations on each measurement create an effect called Gyroscope drift. Gyroscope drift is the variance between a calculated measurement and the real measurement of a gyroscope.

The fusion sensor is based on studies and implementations made by Colton AND Sachs in [73] [74]. The idea behind the fusion sensor is to mix all three measures using

different kind of filters to return accurate information over a period of time. Figure 5-21 shows the process to use the different sensors to calculate the desired orientation.

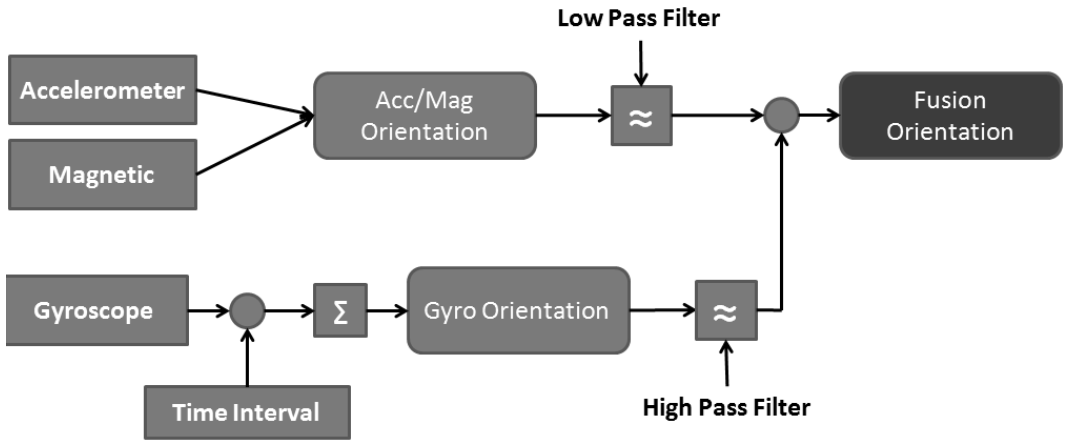

Figure 5-21: Fusion Orientation calculation methodology. **[75]**

A low pass filter is a process that takes low frequency signals and filters data with a higher frequency. A High pass filter is the opposite, on which the process takes high frequency signals but filters data that varies on a lower frequency. Using this methodology, it is possible to reduce the main problems that appear using a single approach.

The low pass filter is made using the following procedure:

$$AccMagOrientation = (1 - factor) * AccMagOrientation + (factor * newAccMagValue)$$

Factor is defined as the frequency to be filtered. The high pass filter is done by replacing the low pass filtered data with the integrated orientation signals:

$$FusedOrientation = (1 - factor) * newGyroValue + (factor * newAccMagValue)$$

The Fusion Sensor was implemented following the guidelines and tutorials specified by Lawitzki in [75].

Rotation Vector: The rotation vector is a virtual sensor, which automatically combines the data of internal sensors to produce an orientation matrix for the device. The data produced by the rotation vector is a combination of a rotation angle and a magnitude for each axis.

The data is retrieved as a vector following this specification:

$$< X * \sin\frac{\theta_x}{2}; Y * sin\frac{\theta_y}{2}; Z * \sin\frac{\theta_z}{2} >$$

Each of the three values of the vector represents a product between the magnitude and the sine function of the angle on a given axis. Using this vector, android provides a function to retrieve the rotation angles from the rotation vector (returns each angle θ from the above vector). This function is called `SensorManager.getRotationMatrixFromVector()` and then using the `getOrientation()` function described above. This function returns a Float array of three positions that hold the three device angles defined as:

```
Float rotation = {Azimuth , Pitch , Roll};
```

Both the Fusion and RotationVector mechanisms use the same `getOrientation` function to calculate the orientation, so their result can be used equally by the rest of the implementation classes.

While the rotation vector is the preferred method to retrieve the rotation angles of the device, not all devices have the availability to use that specific sensor. The orientation package implements both mechanisms and it decides at runtime which method to use based of the availability of each sensor on the device. In order to have a simple and extensible implementation that can use different mechanisms, the Strategy behavioural design pattern has been used on the Orientation package.

The Strategy pattern is used to externalize the behavior of a class and to give the capability to change its behavior at runtime [58]. This design pattern is used on the `OrientationManager` class. This class implements its functions using the abstract class `OrientationListener` as shown on Figure 5-20. It relies on the `OrientationFactory` class to create an instance of the appropriate implementation of the abstract class. This pattern allows the `OrientationManager` to register and unregister the sensors without knowing the actual implementation of the listener, improving the extensibility and maintenance of the project (changing the listener implementation won't modify the rest of the orientation package, as long as it follows the methods defined on the abstract class).

Based in the design specification, the overlay is the main object that receives information regarding the orientation and location of the user. Each `ImageOverlay` is created with an incident that surrounds the user. The class `ImageOverlay` has the responsibility to calculate its position on the screen based on the device's location, the position of the incident and the orientation of the device. Since every overlay receives the information from the same sources (location and orientation listeners), the implementation follows the observer pattern as shown on Figure 5-22.

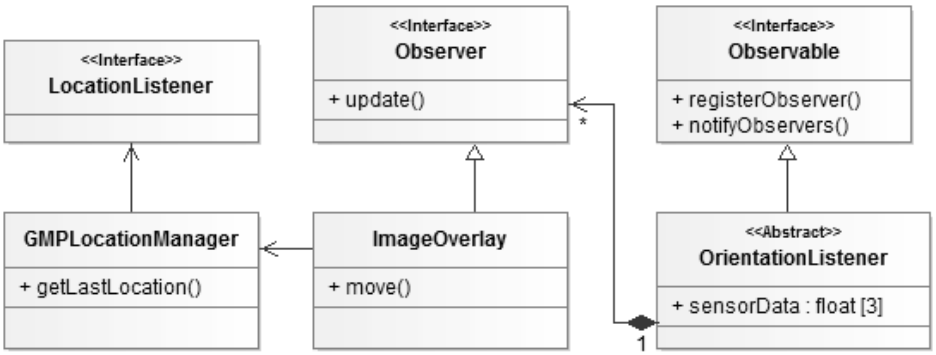

Figure 5-22: Overlay and Observer Implementation.

The `OrientationListener` abstract class represent the "`Observable`" part of the pattern. The `ImageOverlay` class represents the "`Observer`" part. Once a change in orientation is measured, the observable informs all the observers of said change. The class that implements the `Observable` interface has the responsibility to store all references to the observer instances.

Every time that the `ImageOverlay` receives a change in measurement, it has to move itself on the user's screen (on top of the video feed from the camera). This movement is calculated using orientation data received from the orientation observer and the incident location the overlay references.

The process to calculate the user position is based mainly on the azimuth rotation of the device since it is the angle that references the orientation over a world reference frame. The complete process to calculate the user movement is described below:

The first step of the calculation is to provide a reference angle of the incident according the device orientation. This calculation takes the information of the azimuth, bearing of the incident, and the declination of the device. The three types of angles are shown on Figure 5-23.

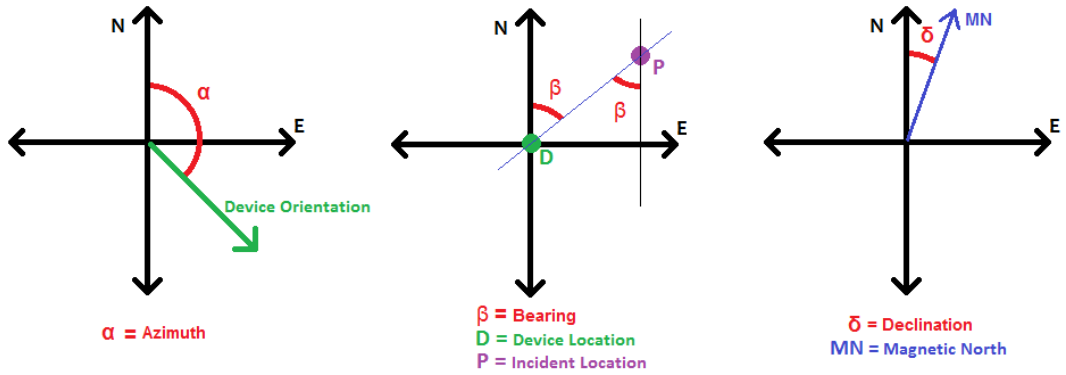

Figure 5-23: Angles used to define orientation.

- The azimuth is measured as the angle between the Magnetic North and the direction the device is pointing (measured eastwards). The azimuth value is received from the `OrientationListener` class. The azimuth changes every time the user moves the device.
- The bearing between two points (user location and incident location) is the angle between the Magnetic North and the line that intersects both locations, measured eastwards. The bearing is calculated by taking the location of the device and the incident location. The user location is retrieved from the `LocationManager` class. The location object uses the method `bearingTo`, which receives the incident location as parameter and returns the bearing. The bearing changes every time a user changes its location.
- The declination is the angle difference between the true North and the Magnetic North. The device's sensors measure the data based on the magnetic north, so it is necessary to compensate for the difference in order to correctly position the overlays (location information is based on the true north). The declination is calculated using the class `GeomagneticField` from the android API, which receives the location of the user and the current date (the declination changes with time and location) and returns the declination angle.

Using the azimuth, bearing and declination, a translated rotation is calculated. The translated rotation is the angle measured from the device orientation vector to the incident location. Figure 5-24 shows an example of three incidents, with their translated rotations specified on the orientation plane of the device. This rotation angle is measured clockwise from the device's orientation.

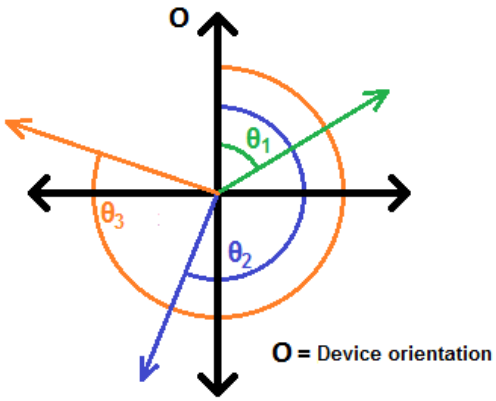

Figure 5-24: Orientation measurements of incidents with device orientation.

The translated rotation angles are closely related to the device orientation. To understand this measurement, Figure 5-25 shows a projection of the angles based on the device usage.

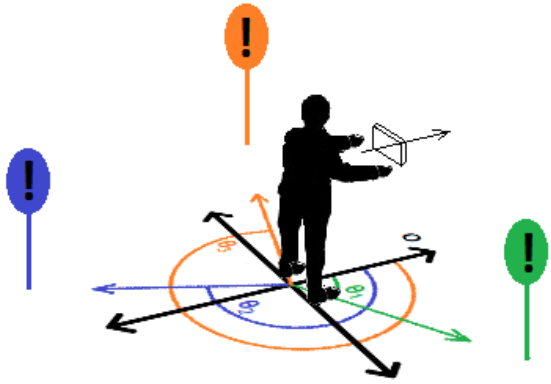

Figure 5-25: translated rotation angles of three incidents.

When the user rotates along its location, the orientation of the device changes, generating variations over the rotation angles. If the user in Figure 5-25 rotates towards the green incident, the rotation angles (Θ_1, Θ_2 and Θ_3) would change to a smaller value. If the user rotated towards the orange incident, the values of the rotation angles would increase.

Once the translated rotation angle is calculated, it is necessary to calculate if the overlay is visible on the user's screen. All camera devices have a predefined angle of vision. The angle of vision is the angle on which the camera lens can capture images from the outside world. Figure 5-26 shows how to understand the angle of vision related to the device orientation.

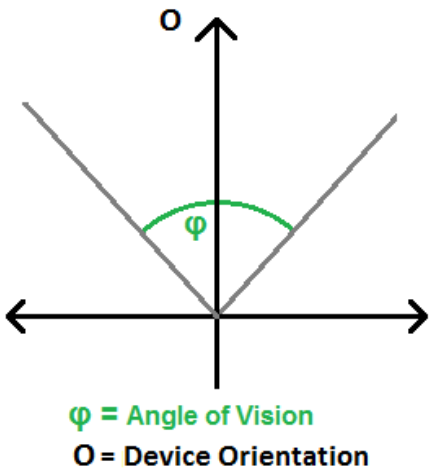

Figure 5-26: Measurement of the camera's angle of vision.

An overlay is visible if the rotation angle is located within the angle of vision. If a rotation angle (Θ) is visible on the screen, Θ has to adhere to the following conditions:

$$0° < \boldsymbol{\theta} < (90° - \frac{\varphi}{2})$$

Or

$$(360° - \frac{\varphi}{2}) < \boldsymbol{\theta} < 360°$$

The code used to calculate the visibility of the overlay is shown on Table 5-18. It is a simple procedure that returns a Boolean with a true value if the rotation angle is in the angle of vision and false otherwise.

```
private boolean isVisible(){
            boolean visible = false;
            if(currentRotation > 90 && currentRotation < 270){
                  visible = false;
            }else{
                  if(currentRotation <= 90){
                        if(currentRotation <= angleVis/2){
                              visible = true;
                        }else{
                              visible = false;
                        }
                  }
                  if(currentRotation >= 270){
                        if((360 - currentRotation) <= angleVis/2){
                              visible = true;
                        }else{
                              visible = false;
                        }
                  }
            }
            return visible;
      }
```

Table 5-18: ImageOverlay visibility implementation

When the overlay visibility is determined, a calculation is made to measure the position of the overlay image in the device's screen. The measurement is based in right-angled triangles, where the hypotenuse of the triangle is the distance between the user position and the incident, and the angle opposite to the right angle (angle of 90°) is known. Figure 5-27 shows two different incidents and the triangles used to calculate their position.

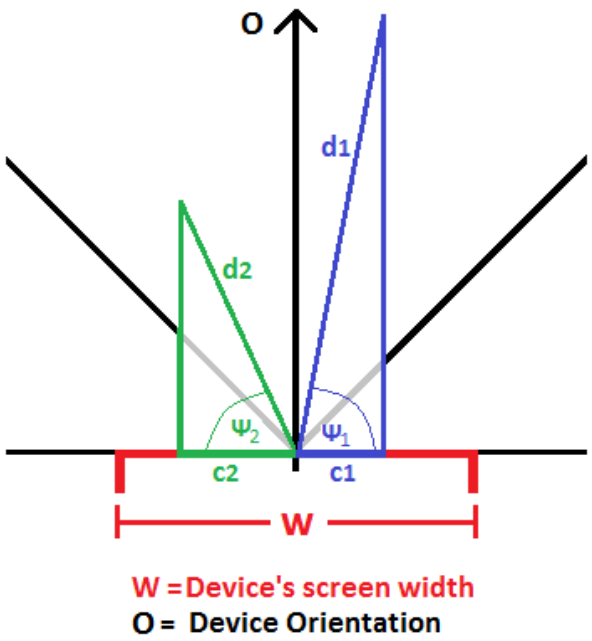

Figure 5-27: Calculation of screen movement of overlays within angle of vision.

The position of the overlay in the screen is measured to the right and to the left from the centre of the screen. The measurement is calculated as follows, where c is the distance from the centre of the screen and d is the distance between the user and the incident:

$$c = d * \cos(\Psi)$$

In this case, Ψ is different from the translated rotation angle and has to be calculated. This calculation is made with the following rules.

- If the rotation angle (Θ) is less than 90°:

$$\Psi = 90° - \Theta$$

- If the rotation angle (Θ) is greater than 270°:

$$\Psi = \Theta - 270°$$

The overlay position has to be calculated taking into account that the rotation measurements are based on the orientation of the device, which is measured on the axis that goes perpendicular to the plane of the screen. Table 5-19 shows the implementation calculation of the overlay's position using the above rules.

```
if(currentRotation <= 90){
       angle = 90 - currentRotation;
       x = centerx + (int) (coeficientAz * distanceFromDevice *
                     Math.cos(Math.toRadians(angle)));
}else if(currentRotation >= 270){
       angle = currentRotation - 270;
       x = centerx - (int) (coeficientAz * distanceFromDevice *
                     Math.cos(Math.toRadians(angle)));
}
```

Table 5-19: Calculation of the overlay position in the screen.

In table 5-19 `centerx` is the center point of the screen (screen Width / 2). A coefficient (`coeficientAz`) had to be added to the function, due to the fact that the distance between the two locations is measured in meters, while the screen width is measured in pixels. The coefficient is calculated based on the distance of the device from the user position and the angle of vision of the device:

$$\boldsymbol{Visible\ Width} = 2 * distance * \tan\frac{\varphi}{2}$$

$$CoeficientAz = \frac{Screen\ Width}{\boldsymbol{Visible\ Width}}$$

Using the variable `coeficientAz` in the calculation, allows the distances in meters to be converted to pixels so it can be shown accurately in the screen. Figure 5-28 shows the calculations taking into account the device screen.

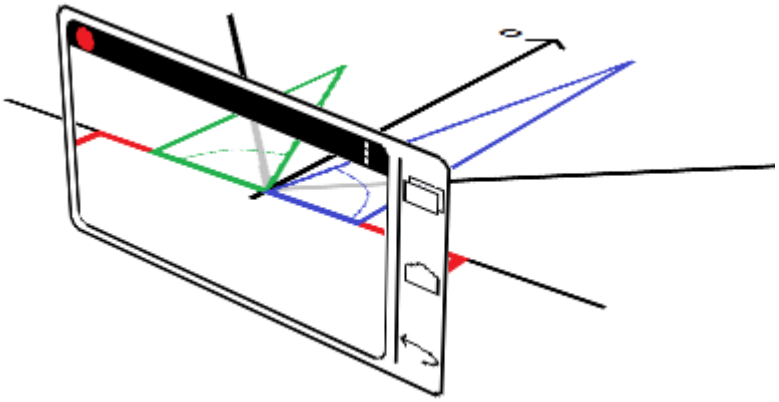

Figure 5-28: Overlay position calculation related to the screen

The calculations made for the change in orientation for Pitch and Roll movements are calculated based on the same principles as the azimuth rotation change.

Due to the fact that not all incidents are visible through the user screen at any given time, a special view was created to display all the incidents surrounding the user. The view is similar to a radar display as shown on Figure 5-29.

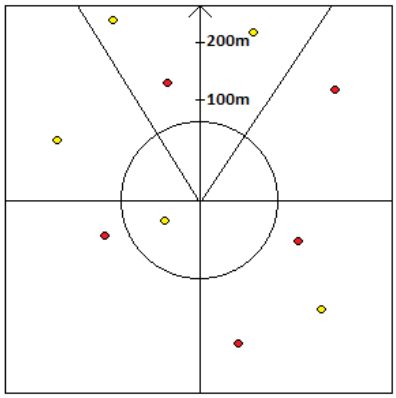

Figure 5-29: Radar display of incidents.

The view uses the azimuth rotation to change the incidents position so it uses the data returned by the `OrientationListener` class through the observer pattern as shown on Figure 5-30.

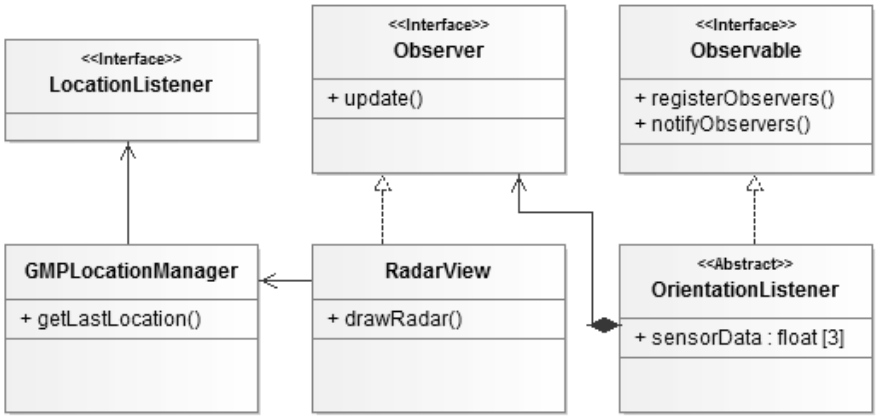

Figure 5-30: Radar class implementation.

The view uses two special attributes called `Canvas` and `Paint` provided by the Android API. `Canvas` can draw lines and points using the `Paint` class. Every time a new measurement is updated from the overlay listener, the `Canvas` and `Paint` classes draw the view from scratch. The process to paint the radar is shown in Figure 5-31.

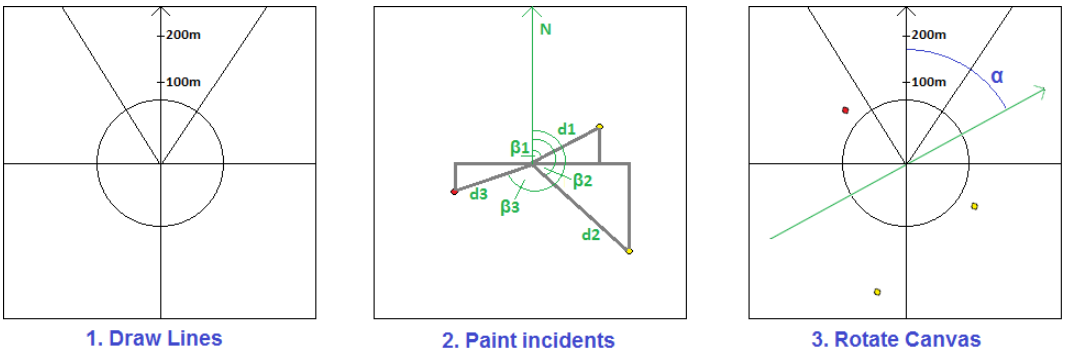

Figure 5-31: Radar painting process.

First the painter draws over the canvas the horizontal and vertical lines intersecting at the centre of the view. Then, it draws a circle and lines representing the angle of view. The view then sets the user's location in the centre of the radar and paints the incidents based on the distance (d) and bearing (β) between the user's and incident's location. The canvas is rotated according the azimuth rotation (α) received from the `OrientationListener`. The radar view is placed to the left of the main AR view (video feed with overlays). So the user can use the main and radar views without disrupting the overall operation of the application. A working implementation of the AR mode with both views is shown in Figure 5-32 and Figure 5-33.

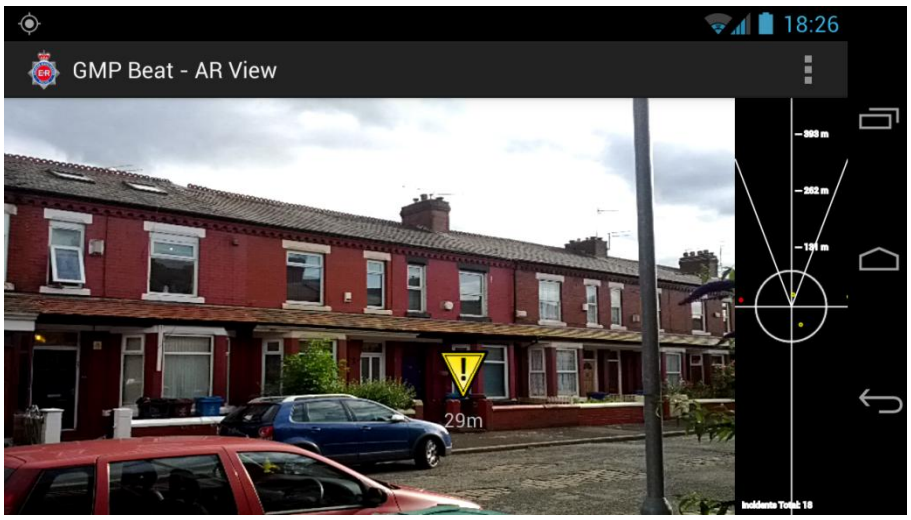

Figure 5-32: Screenshot of AR mode implementation

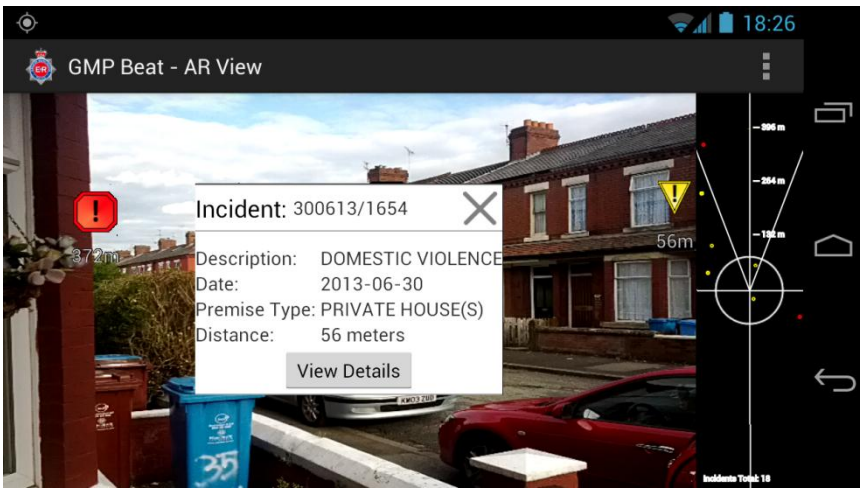

Figure 5-33: AR mode implementation with description window.

5.4.6 Service mode

The service mode has been implemented following the design specifications; however the design has been extended to provide a better fit with the characteristics of the environment.

To create a service, the application has to reference the class that represents the service in its global properties. These properties are stored on the AndroidManifest.xml file of the project as seen on Table 5-20. There are two properties that need to be set, which are “service” and “receiver”.

```
<service android:name="uk.ac.man.gmp.full.service.BackgroundService" />

<receiver android:name="uk.ac.man.gmp.full.service.ServiceStart" >
   <intent-filter>
         <action android:name="android.intent.action.BOOT_COMPLETED" />
   </intent-filter>
</receiver>
```

Table 5-20: Service mode configuration.

The “service” tag represents the main class of the service which has the responsibility to manage the process lifecycle. In order for the service to be kept alive, even when the application is not in use by the user, a receiver property has to be set to retrieve a specific signal from the operating system. This signal is called “Intent” and it describes the type of action that is being executed. In this case the signal references the boot or start of the device. Each class on the property description has to be set with the full java qualified name description (`package_name.class_name`).

Figure 5-34 shows the class architecture implemented for the service mode. The class that receives the intent is called `ServiceStart` as described on the service configuration. This class references the main service `BackgroundService`.

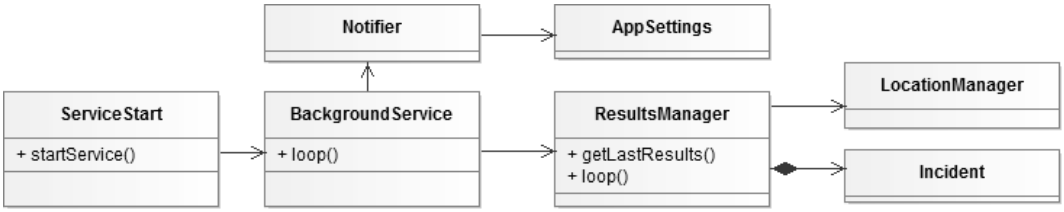

Figure 5-34: Service mode class implementation.

The `BackgroundService` class is responsible to generate the Notification given a certain time. To do this, it references the `Notifier` class that has the responsibility to notify the user if there are new incidents near its position.

The `BackgroundService` class also references the `ResultManager` class, which has the responsibility to query the nearest incidents based on the location of the device. `ResultsManager` is also a thread that runs in parallel with the process thread since the process execution could have different timescales on which they operate.

The `Notifier` class references the application settings manager to retrieve information about the user preferences regarding the sound, vibration and light options of the notification to display.

Figure 5-35 shows the interaction between the classes and the sequence of events that leads to a notification.

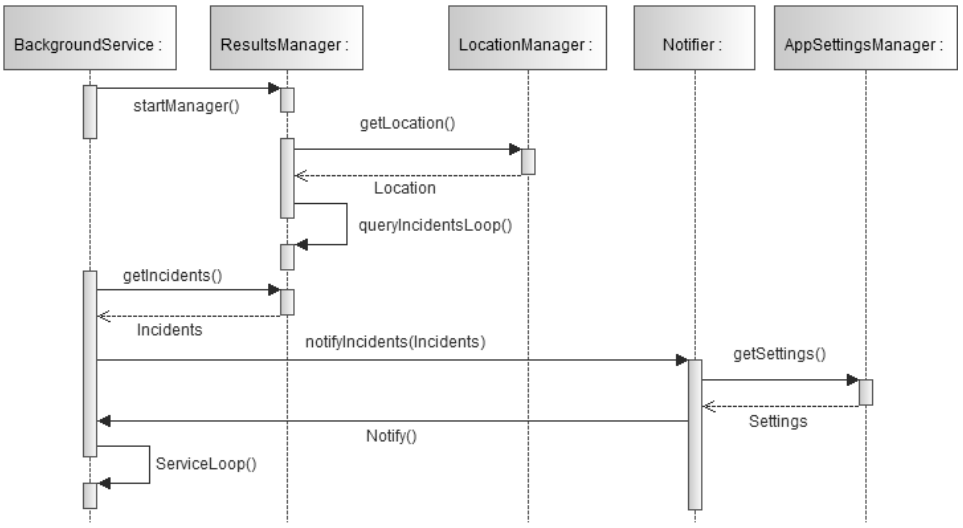

Figure 5-35: class interaction on service mode.

`BackgroundService` runs the notification process, while `ResultsManager` runs a query process. The notification is made at different intervals than the query process (`ResultsManager` has to search for the location of the user more times within a single execution of the main background process).

`Notifier` uses the android API, specially a class called `NotificationManager` which generates a small notification to the user once it has found incidents nearby. Figure 5-36 shows the notification on top of the android interface and the message that is displayed to the user:

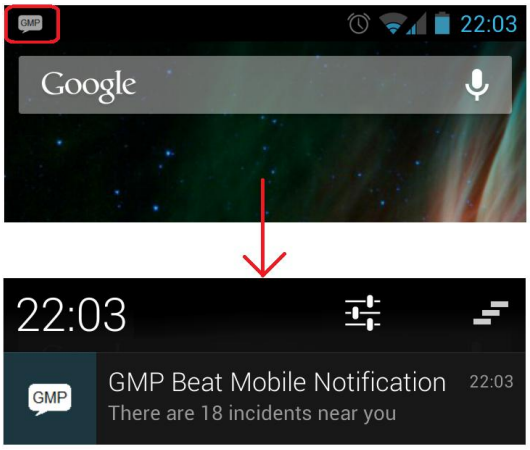

Figure 5-36: Notification display generated by the service mode.

When the user clicks the notification, the application will be started and the user can use one of the main modes of operation to search for the important incidents around the area. The notification is only generated if the user has moved past certain distance of the last notification and if there are more than zero incidents in the area based on the current filters applied by the user.

5.4.7 Additional interfaces

In addition to the main three modes of operation, work had to be done on the UI to display the detailed information of the entities and the configuration interfaces.

The AR and Map modes show the same incidents in different interfaces, and from each mode an incident detail can be shown. From the entity detail interface the other entities interfaces can be displayed. Each entity has an interface to display its information.

Each entity detail interface follows the same functionality. Each Activity in charge to display each entity's UI receives an `Intent` class. The intent is an android class that it is used to signal the OS that a new activity needs to start. The intent can carry messages to other activities (String variables) with information that needs to be shared between them. From the Map and AR modes, the `IncidentActivity` class is launched with the code shown in Figure 5-21:

```
Intent intent = new Intent(ModeActivity, IncidentActivity.class);
intent.putExtra("ID_INCIDENT", incident.getCode());
ModeActivity.startActivity(intent);
```

Table 5-21: Mechanism to start a new activity (Incident).

If the user selects a marker on the map or an overlay on the AR mode, the above piece of code is executed, which loads the `IncidentActivity` class, and submits the incident code. When the new Activity is loaded (`onCreate`() method of the Activity), it requests the message from the intent and recreates the Incident as a class as shown in Table 5-22.

```
Intent intent = this.getIntent();
String idIncident = intent.getStringExtra("ID_INCIDENT");
Incident inc = Incident.getInstance(idIncident);
```

Table 5-22: Activity receiving and using an Intent.

Figure 5-37 shows the interaction between the entity activities and the model classes representing the entities. Each activity gets the instance of the associated entity and the instances of the related entities (e.g. `IncidentActivity` loads the incident and the crimes related to that incident). Each Activity loads the entity's data on text fields defined in their associated UI's layout file.

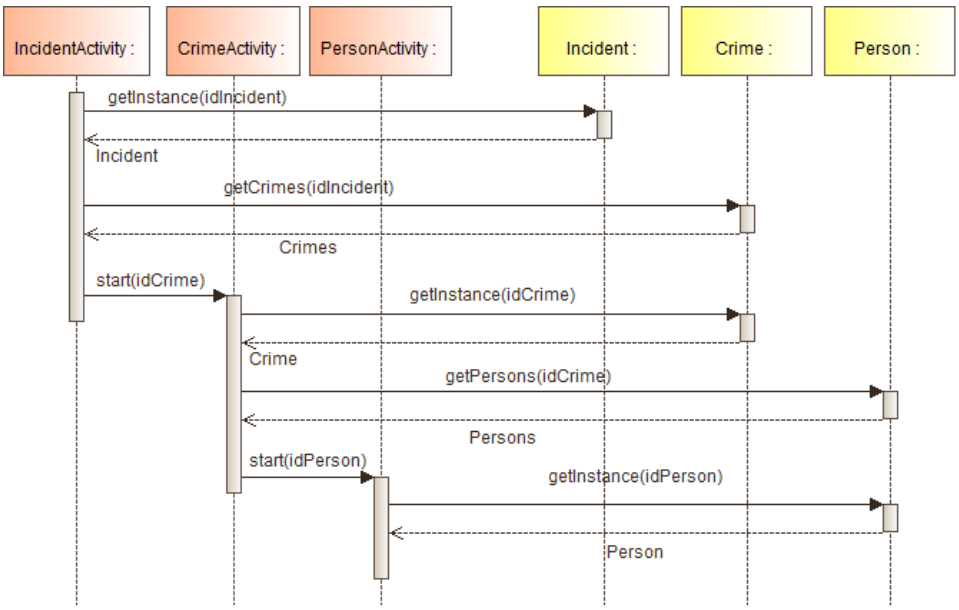

Figure 5-37: UI and model class interaction.

Person is the bottom entity; it doesn't have additional related entities attached to it. To return to the any of the modes, the user only has to select the "previous" button provided by android and the OS returns to the previous activity. This is a behaviour used by default, and it doesn't need to be coded into the system. The OS knows the previous activity through the intents that have been sent to load the classes. An example of the activity interface, showing the detail information and the buttons to load the related entities is shown on Figure 5-38.

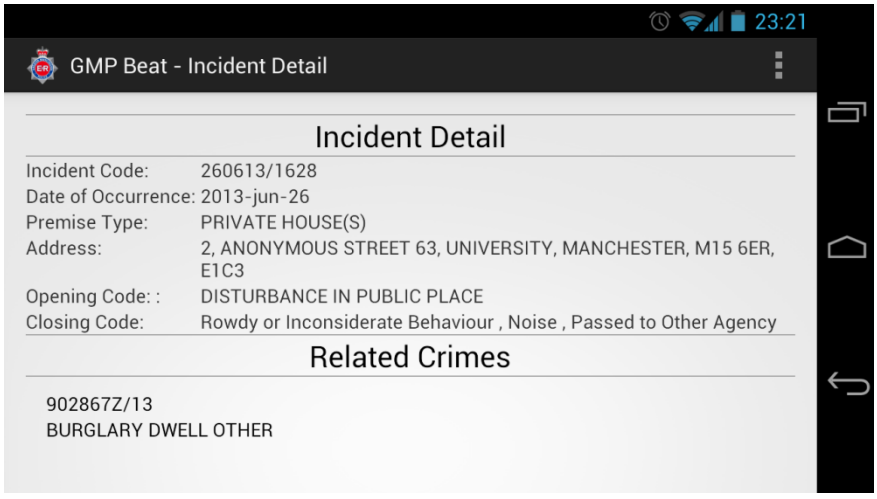

Figure 5-38: Incident Activity interface example.

The configuration interfaces have been implemented to display a simple and user friendly experience. There are 2 types of settings that can be set, "application" and "filter" settings. Application settings are related to the overall application features. A composited interface for the application settings is shown on Figure 5-39.

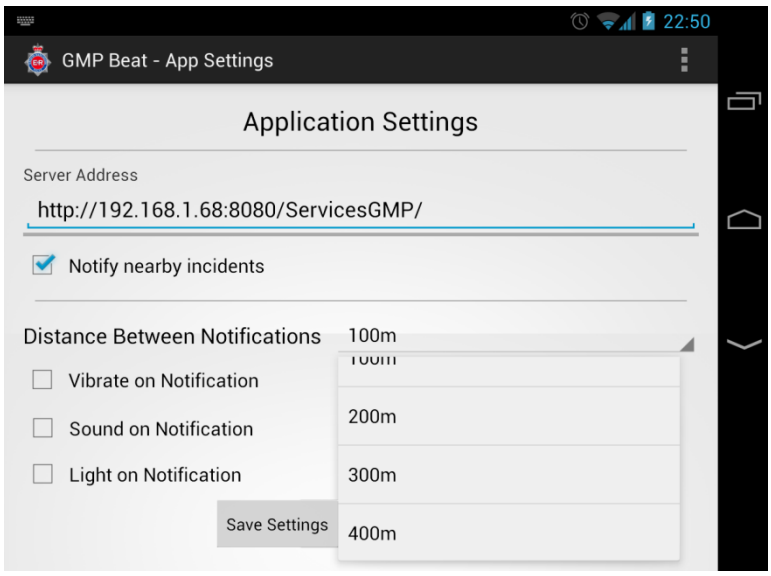

Figure 5-39: Composited screenshot of settings interface (application).

In this interface the user can configure the server address where the MW layer is located. It is the only option on which the user may need to type any data. The other configuration options are based in checkboxes and list pickers. This allows the user to have better UI recognition and reduce command recall in the interface. The options allowed in this interface are: Notify nearby incidents (tells the service mode to send a notification or not). It is also possible to select the number of meters that the user has to move before a new notification can be shown. The notification options are very simple; the notification can be configured to provide a sound (standard android notification sound), a small vibration and an option to generate an intermittent light (Red, if the device supports light notifications). At the end of the interface a button is located and which saves the current configuration for all attributes.

For the filter setting a single interface was created as shown on Figure 5-40. In this interface the user only has the option to select each filter option from a series of check boxes.

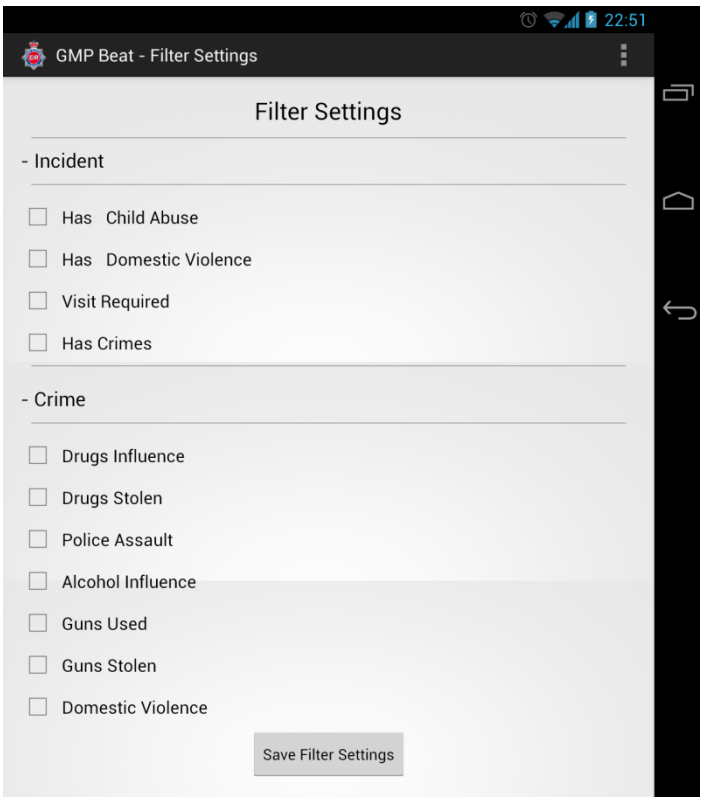

Figure 5-40: Composited screenshot of settings interface (query filter).

The user has the possibility to select each and every one of the settings options. There are two groups of filters, “incident” and “crime”. All options generate an “AND” type filter, that is, each incident returned has to comply with each filter check selected. All filter options are designed to follow the DB layer data attributes.

The Incident filter has only 4 options to select. The first three options are based in incident attributes, while the last one (Has Crimes) is a calculated attribute. Once this option is checked, the filter will only return results of the incidents that have crimes related (Seen as red markers on the 2 main modes of the application).

The crime filter has several more options. They are all based on crime attributes. While the result of the filter on the UI only brings incidents through the AR mode, the crime filter was implemented to increase the flexibility in the information requested by the user. Once an option of the crime filter is selected, all the incidents returned will have related crimes that contain the selected filter option. All the incidents returned using a crime filter option have red markers.

5.5 Chapter Summary

To implement the design, several efforts have been done to meet the specification described on the requirements and its technical design. The implementation effort has been directed into understanding the implications of such development, including learning the development environment and the understanding of the mobile application lifecycle. Work has also been done to implement the system's layers and all its modes of operation, with emphasis on the more challenging AR mode. The application development effort has been separated on three different aspects, applying the Map interface to the mobile application, creating overlays over live video feed using geographic data and creating a background service. All the user interfaces were implemented following the project and development guidelines.

6. Evaluation

The development of the complete system has been extensive and several tests have been done to ensure that all components and the integration between them worked as expected. This chapter covers the evaluation performed over the full system; including tests made during the application development and after the development was finished.

The evaluation is separated in two aspects, technical and functional. Technical evaluation covers the technical aspects during development and integration of components made during and after the development process. The functional testing covers the evaluation of the project and client's requirements based in the guidelines specified in chapter 3.

6.1 Technical Evaluation

The system is composed of 3 different layers and each one has been evaluated during implementation and after it. During implementation, data integrity tests have been done for the DB layer. Unit testing for the MW and Application layers have also been done. After the implementation, Integration tests, which covers reliability and performance on several features has been done.

6.1.1 Evaluation during implementation

The evaluation is separated between the 3 layers that have been implemented. Each layer has different characteristics, and each layer has been evaluated differently during the implementation phase.

DB Layer: This layer has been implemented importing data from source files provided by the GMP. The source files that were provided in excel format were ported to CSV files and then ported to the database engine. To ensure a proper implementation, tests have been made to validate data integrity after the importing process finished. A test for this has been designed to evaluate the number and types of values of each table; also a manual comparison with the source information has been done. The test has different aspects that the import process must comply in order to be considered a successful process.

Among the aspects that has been validated during the process is the number of rows on each table must match with source data. Queries on the data have also been made using SQL joins to test accuracy on data and to verify match of data types. Each numeric type on each table has been added to verify totals to guarantee that each number was imported correctly.

While all the information has been imported correctly, the columns with date types presented difficulties on the process. MySQL requires the date columns to have a certain format; however the information stored on the data files had inconsistent formats, which required a change in the import statements. Once these difficulties were corrected, the validation was successful for all the data's tables and columns.

To test the performance of the database, queries involving the main and largest tables (incident and crime) have been performed. All queries were based with different joins and filters and based on the most common information requested by the MW layer. All queries performed quickly, which is expected due to the use of indexes and the amount of information stored. On average all queries have an execution time of less than 0.2 seconds with a standard deviation of 0.1 seconds. This response time ensures that the DB layer is not a bottleneck of the system and the response times are uniform and without much variation.

Security tests have not been made as part of the technical evaluations on the DB layer. While is an important aspect during development, implementation of security restrictions is outside of the scope of the project.

MW Layer: During implementation, to evaluate the MW layer, a series of Unit tests have been designed to test the validity of each class and to review the quality of the code. Unit tests are focused on testing individual classes to find issues early in development and simplify the integration between the different classes. Unit tests usually validate the class methods signature and return values.

The unit tests have been classified in the types of packages defined on the MW layer as shown on Table 6-1. For each class a test procedure have been created to evaluate each method. Each test procedure defined a series of input parameters, evaluated the result value and checked if the result value is the expected one or different. The tests

were made twice, first after the first implementation and then after corrections were applied to the code.

			Success Rate	
Type	**Class**	**Methods**	**Initial**	**Final**
Connection	ConnectionPool	3	66%	100%
Command	CrimesFromIncidentCommand	2	50%	100%
	IncidentDetailCommand	2	100%	100%
	ListIncidentCommand	2	100%	100%
	PersonsFromCrimeCommand	2	100%	100%
	SingleCrimeCommand	2	50%	100%
	SingleIncidentCommand	2	50%	100%
	SinglePersonCommand	2	100%	100%
Manager	QueryManager	1	100%	100%
	CommandFactory	1	100%	100%
Service	Crime	1	100%	100%
	CrimesFromIncident	1	100%	100%
	Incident	1	0%	100%
	ListIncidents	1	0%	100%
	Person	1	100%	100%
	PersonsFromCrimes	1	100%	100%
		Average	76%	100%

Table 6-1: Unit tests results for the MW layer.

The main tests performed were related to the connection to the DB layer, the relationship between the services and the commands that executed the query. To test the result values, SQL queries where created and executed directly on the database, and their result was compared with the service text output. Some of the tests included more than one class (for example a class from service used a class from manager to be able to execute the complete test).

During the implementation all MW classes have been tested, and once a specific service passed the tests it was available for use in the application layer code. The most common type of errors that have been found, are related to mistakes made on parameter processing, and DB connection management.

Application Layer: To test this layer, unit tests have also been applied to all the code's classes and packages. As with the MW layer, unit tests have been executed until the errors found in the code were corrected. Table 6-2 shows the results of the unit tests executed to evaluate each class in the layer. The table shows summary results based on the number of classes and methods.

			Success Rate	
Package	**Classes**	**Methods**	**Initial**	**Final**
Service	4	13	77%	100%
AR	7	28	79%	100%
Map	4	20	70%	100%
Orientation	7	21	71%	100%
Location	2	8	88%	100%
Model	9	36	92%	100%
Settings	3	12	100%	100%
UI	19	57	82%	100%
		Average	82%	100%

Table 6-2: Unit tests results for the application layer.

Since each package has a different purpose, the number and type of errors varies. A summary of the most common errors found for each package evaluation are described below.

- **Service**: Thread errors in class initialization, Race conditions, and private attributes not correctly initialized.
- **AR**: Errors in orientation calculation and private attributes not correctly initialized.
- **Map**: `GoogleMap` permission errors, the window description listener was not displayed and markers' initialization with null values.
- **Orientation**: Fusion listener calculation errors, sensor initialization with erroneous values, and strategy pattern not functional with both orientation listeners.
- **Location**: Returned location was null after sensor initialization.
- **Model**: Creation of entities with wrong attributes, service consumption failed with correct parameters and private attributes not correctly initialized.
- **UI:** Class initialization errors, activity lifecycle mismanaged and attributes not correctly initialized.

It is worth noting that while the results of the unit tests are satisfactory (100% final success rate), the system is not guaranteed to be bug free. Unit tests reduce the number of errors, but there are different scenarios and parameter values that would trigger an exception in the code that is not caught by a simple unit test. A systematic

evaluation of functionalities between different packages provides a better view of the overall code quality of the project.

6.1.2 Evaluation after implementation

Once the individual classes have been evaluated, an analysis that tests the communication between the layers and the overall system performance is needed to ensure that the most outstanding bugs have been corrected. Table 6-3 shows a series of test cases that have been designed to ensure the quality of the application. The Table 6-4 shows the bugtracker used during the tests, which shows the state of each error found and final state for each test case.

Code	Test Case	Expected Results
GMP1	Create filter on settings page with combination of attributes; display results on Map mode	Yellow Markers found on incidents without crime. Red markers otherwise.
		Current location marker displays the image within 5 meters of actual physical location.
		A touch of each marker displays description window. Once it's touched, launch incident detail page.
GMP2	Create filter on settings page with combination of attributes; display results on AR mode	Yellow overlays found on incidents without crime. Red overlays otherwise.
		Position of overlays match with the orientation of the device (compared with the map view).
		Movement of device produces overlay movement according with angle and speed of movement.
		Radar interface displays the number of incidents with yellow and red points.
		Radar's angle of vision shows the same information as the camera display.
		A touch of each overlay displays description window. Once it's touched, launch incident detail page.
		GPS Indicator is on, while GPS is activated and the user is on AR Mode.
GMP3	Create filter with combination of attributes; Force service. Generate notification.	Service starts in background, shown on Android process manager.
		Service generates notification with number of incidents.
		With change of position, service generates new notification.
		Device vibrates, makes sound and show light according to application settings defined.
GMP4	Show Incident detail interface, with information from the MAP and AR modes	Display detailed information from incident.
		Display related Crimes as buttons.
GMP5	Navigate to the Crime and Person entities from Incident Interface	Touch gesture on related entity button display Entity detail page.
		Number of associated entities match with query retrieved from MW Layer.
		Touch gesture on related entity button display Entity detail page.
		Touch on "previous button" returns to previous related entity.

Table 6-3: Test cases description.

The source of each error has been different in each case. The debugging process has been much more resource expensive than simple unit tests. Most of the time, the root cause of the errors has been found on the application layer. Some test cases have more than twenty classes interchanging information at any given time, increasing the difficulty to locate and correct each error.

Code	Errors Found	State
GMP1	Filter parameters badly constructed, unexpected incidents in result.	Corrected
	Location displayed is different from actual location. Accuracy errors.	Corrected
	Window description does not display incident data.	Corrected
GMP2	Number of overlays lower than the number of expected results.	Corrected
	Overlay position doesn't match. Orientation difference between 45 and 90 degrees.	Corrected
	Overlays move in different direction than the device's movement. Erratic overlay movement.	Corrected
	The number of incidents differs from the data points in the radar.	Corrected
	Incident inside angle of vision not displayed on camera screen.	Corrected
	Description window not rendered properly. Touch gesture does not display incident interface.	Corrected
	GPS Indicator is intermittent.	Corrected
GMP3	Service is not displayed on android management console, failed to start.	Corrected
	Number of incidents differs from the expected number of results based on the filter.	Corrected
	The notification is generated without a change in user position.	Corrected
	Light notification does not show the appropriate colour.	Corrected
GMP4	Information on interface appears empty.	Corrected
	Crime selection buttons did not appear in interface.	Corrected
	Interface layout shown misaligned.	Corrected
	Number of related crimes differs from the actual crimes retrieved from query.	Corrected
GMP5	Number of related entities differs from the actual number retrieved from query.	Corrected
	Entities pages are displayed incorrectly / with bad layout.	Corrected

Table 6-4: Bugtracker with error information and state.

After all the test cases have been performed, a series of performance tests have been included to validate the application responsiveness over different scenarios. Due to the nature of the system (mobile environment), the internet/data connection is an important aspect of the system, also the type of location sources. While the results are qualitative in nature, they show the most important aspects in the system operation.

Table 6-5 shows the qualitative results based in different scenarios that have been tested. IT shows the results gathered from the 3 main modes of operation for different connection speeds and location providers.

Scenario	Map Mode	AR Mode	Service Mode
Wi-Fi	- Smooth map navigation. - Fast load times.	- Fast load times.	- Fast load times.
3G	- Map navigation tolerable. - Fast data retrieval on small data sets.	- Fast load times on small data sets. - Slow load on big data sets.	- Fast load times on small data sets. - Slow load on big data sets.
3G+ (HSPA)	- Smooth map navigation. - Fast data retrieval on small and medium data sets.	- Fast load times on small and medium data sets. - Tolerable load times on big data sets.	- Fast load times on small and medium data sets. - Tolerable load times on big data sets.
GPS	- Accurate location (~5m)	Accurate overlay orientation.	Accurately generates notification after user moves
Network Location	- Imprecise location (~100m).	Variations on overlay orientation (> 40 degrees difference).	Imprecise notification generation, new notification before user covers enough distance.

Table 6-5: Qualitative performance on different networks and location providers.

From the table's qualitative analysis, it is recommended that for an optimal application performance, the user's device should have a working GPS sensor and should be able to connect to a 3G navigation (High Speed Packet Access – HSPA recommended). For the different tests performed, the location retrieved from the network provided was very imprecise. Sometimes the difference could be greater than 100 meters from the actual location. This significantly affects the AR mode, on which most of the calculations are based on the current location as a source for the orientation measurement. Since the application is supposed to work on a mobile environment, Wi-Fi connections are not widely available. For regular 3G connections the application works without issue, but it is recommended that the user use the filter to reduce the number of incidents retrieved from the MW layer.

6.2 Functional Evaluation

To evaluate the different functionalities of the system, the full list of requirements needs to be tested to ensure that the application covers the user's needs. It is also important to qualify the application's overall user experience, so the SUS is executed according to the guidelines specified on section 3.5.

6.2.1 Requirement evaluation

As stated on Table 4-1, there are several high level user and project requirements that need to be validated. Table 6-6 shows a series of use cases based in these requirements**.** These use cases provide the methods and steps to guarantee that the application fulfils the user's needs.

Req.	Use case	Result
1	- Start application - Select Map or AR Mode on selection screen. - Select one incident marker. - Select Open detailed interface.	Application displayed summary information of Incident, Crime and Person entities.
2	- Start application. - Select AR Mode from selection screen. - Application must display a photo-realistic view with geo-referenced information.	Information displayed on a photo-realistic view (Camera) and with references to the information's geographical position.
3	- Open internet browser. - Go to Police.co.uk and retrieve information around M14 4DJ post code. - Start application near M14 4DJ post code - Select Map and AR modes from selection screen. - .Mode must display same incidents than police.uk website. Display more detailed data.	Filter attributes on both (police website and application) are similar, map modes present similar features. The website displays aggregate data (on close but inaccurate locations). Website does not show information about crimes, victims and offenders. Website can't support AR mode.
4	- Start device. - Wait for 5 minutes to generate incident notification. - Move for 100m from original position. - New incident notification displayed.	The user received information of nearby incidents. User received it without asking for it. (Push mode).
5	- Start application - Select Map or AR Mode on selection screen. - Select one incident marker. - Select Open detailed interface. - Navigate to crime interface, and person interface.	Information displayed details addresses, crime description, information about victims and offenders, incident addresses and geographical information.

Table 6-6: Use cases for Requirement evaluation.

The project requirements (numbered 6 through 8 on Table 4-1) have been represented by the different chapters described in the dissertation. The review of available technology is described on the chapter 2 (Background). The development of proof of concept designs is described in chapter 3 (Design). The development of application prototype that reflects the user requirements is described in chapter 5 (implementation).

After each use case has been executed, the result has been compared with the expected outcome of the evaluation. In all cases the result is satisfactory, giving a requirement evaluation of 100%, which is over the threshold specified for a successful implementation.

A live preview of the application has also been performed. The participants of the preview have been the project's student and supervisor. The preview has been designed to present the application features and capabilities on a real world scenario. The tasks that have been performed on the preview are described further.

- Walk to an area marked with incidents.
- Open the application and test the map mode. Ensure that the current location marker on the map is accurate.
- Use filter attributes to change the number and type of incidents received from the MW Layer.
- Open AR mode with the previous set of filter attributes. Check that the orientation matches with the incidents displayed on the screen on the MAP mode.
- Walk to a selected incident. AR display should be updated with the orientation as the user walks to it.
- Display the detailed information of the incident, related crimes and persons.

After all the tasks have been performed, some issues have been found. The most important of all, it is that the user's location was not being updated as fast as it should to provide a seamless experience. This forced a change in the `LocationManager` code which helped improve the user's experience and increased the application's accuracy and responsiveness.

6.2.2 Usability evaluation

The usability evaluation helps to complement the requirement evaluation by having a usability perspective on the application implementation. The usability evaluation helps determine if the application is user friendly and has a larger probability for adoption among its user base.

The usability evaluation has been offered to the GMP users; however, due time constraints of the GMP and the timescales of the project dissertation, this evaluation could not be carried out. This evaluation cannot be assigned to other users, as different users will not have the same perspective nor the usability arguments needed to evaluate an application designed specifically for the GMP.

As a future work, it is proposed that if improvements are made to the application development, these should include a usability assessment as is designed in this dissertation.

6.3 Chapter Summary

To test the application, technical and functional evaluations have been performed. The technical evaluations have been performed during and after the implementation process. The aim of the technical evaluation is to correct errors on the code and to increase the reliability and performance of the system. Functional evaluations have been performed to test the system capabilities to successfully fulfil the needs and requirements of the GMP.

The results of both sets of evaluations were satisfactory. The technical evaluations have been used to improve the system quality. The functional evaluations have been used to confirm that the user requirements are being met and the application is considered usable by an actual user on a real environment.

7. Conclusions and Further Work

This report contains the progress and effort applied to develop a prototype of a mobile application to query and display relevant location-based information for the GMP on real time. Based on the results of the implementation several important factors have been selected to highlight and to further improve this work in the future.

7.1 Conclusions

The project dissertation encompasses a complete application development process, from an idea to a tangible product. This process included an analysis of the current GMP needs, requirements and overall project scope. This requirements and needs helped formulate an appropriate design and implementation phases. It also defined an evaluation that allows a much clearer understanding of the limitations in the implementation effort. The most important achievements and analysis of the work are described below.

From the background review process, it was possible to understand the number of environment variables in which the final application would perform. An analysis of the cultural, social and economical limitations of the user's needs and requirements allows the possibility to understand the most important aspects of the final product. Aspects such as cost of the device and real world usage allow focusing in the most important elements and characteristics during implementation. During the technical analysis of the available technologies for the implementation, it has been possible to understand the characteristics of the mobile environment and the amount of features that could be used to develop the application. Having this technical background has been a great aid to model and visualize the most important characteristics and features, including the Augmented Reality mode, which presents the most challenging, but also the most interesting aspect of the mobile application.

A methodology has been established to describe all aspects that are taken into account to gather user requirements, specify a design process, implement and evaluate the final mobile application. Having this process and methodology, allows determining and specifying the boundaries and overall projecting scope. Among these boundaries are included the different phases of the development process which allows an accurate estimation of the effort necessary to finalize the project. The most important

result of defining a methodology is the approximate estimation of effort needed to design, implement and evaluate a full application taking into account all the variables that affect the project and its stakeholders.

The design phase helps understand and define all the components that compose the application, starting from the classification of all the requirements as a valuable first step. All the stakeholder's requirements have been selected and categorized which allows the construction of solid functional foundations on which the application is designed and built. In these phase the use of standard design patterns helped create from the ground up an extensible application, which allows with some degree of simplicity if variations in its implementation occur. The design has also been helpful to understand scale of the project and the appropriate technologies and features needed to fulfil the requirements specified by the GMP. The use of UI sketches (mock-ups) from the initial stages of the project has been instrumental to describe the application functionality without effort and additional cost. The sketches also have the characteristic of being easily modifiable, which allowed the possibility to share early design concepts with the project stakeholders.

The greatest amount of effort has been spent on the system implementation, from the database layer to the actual mobile interface used by the final GMP personnel. The implementation included the installation, deployment and configuration of the different components that support each layer of the system. The implementation has been made component by component, based on the design specification. This has allowed the possibility to create from the ground up all the features and achieve the vision and requirements needed by the clients. The use of common components, technologies with free (as in freedom) licenses and standard protocols has created an environment in which the amount of available documentation helps the implementation without a lot of unexpected issues. While the design provided the appropriate foundation to start the implementation, this phase allows the possibility to modify, and in most cases, enhance the design decisions that have been made in previous stages. During the implementation, the android development environment, which uses common a widely used IDE (Eclipse), helps to streamline the development efforts, by giving a natural environment for the developer to work. The use of common tools, including IDE plug-ins, compilers, standard java constructs and

operating systems facilitates several aspects of the process. The structure of the android UI interface helps to structure and separate interface logic from business logic. This allows the possibility to apply the design guidelines in parallel with the development of the business logic, increasing the implementation speed. One important result of the implementation phase has been the identification and formulation of all the necessary equations and calculations to provide a good experience on AR interfaces. Extracting the most important aspects from these calculations has paved the way to future implementations to reduce the implementation time.

During the evaluation phase, a series of functional and technical evaluations have been performed to test the quality and validity of the finished application, during and after the final deployment on the environment. The utilization of technical evaluations such as unit tests, have been a great asset to understand and tackle the different environment restrictions related to the specific implementation. The execution of technical evaluations during development has also been a great tool to find errors early in the process. This reduces the cost (in terms of time and effort) to correct errors compared with the cost associated to correct them after the application has been fully deployed. Using functional evaluations is a great instrument to understand how the user rates the specific implementation, based on the requirements gathered at the start and during the project execution. It also shows how closely the development fulfils the client's needs and the probability of a given implementation to be adopted by the final users of the application / system.

7.2 Further work and improvements

While the effort made on the implementation was considerable, the resulting application is still a prototype which serves the purpose as a technology demonstrator rather than a full production deployment. A lot of characteristics can still be improved and enhanced to provide better results and increased usability to the intended users.

The DB layer has been created with the sole purpose of store the GMP data. A more generic application that uses geographical locations such as source data has to take into consideration another database design to improve the extensibility of the layer. The DB layer during implementation has been using a limited set of data. With a full

environment, further tests have to be done to guarantee an adequate response time, and to deploy an improved database model which could support an increase in the size and the frequency of the queries on the layer.

The MW layer has been designed to extend its current services to other sources of data; however, the services that procure external information have not been implemented. Implementing those features could extend the capabilities and extensibility of the MW layer, and could provide and increased source of information for other clients outside the GMP. The MW layer has been implemented using the same hardware as the DB layer, which on a real production environment it would be needed that the information (DB) and operations (MW) be separated to standalone servers. This separation could provide a better environment to manage and troubleshoot the daily operations of the platform. It could also increase the hardware and software performance.

The Application layer has been implemented to query and display the GMP information consumed from the MW layer. Each of the operation modes can be extended and improved from the UI and design perspective. The Map mode displays an icon to show incidents. These icons cannot be configured or extended to display additional information unless new code is added to the project. To create an even more extensible platform, user configurable options for the UI on the AR mode are needed. Currently the AR mode relies on the accuracy of the location and orientation data measured from the device's sensors. Additional mechanisms to improve accuracy could be implemented, including visual recognition of buildings and streets, and collection of information from surrounding elements, including specially located Radio Frequency Identification (RFID) tags.

The Map mode can be implemented in the same way as the AR mode, including in the design the possibility to include additional entities' types. User configurable options can be added, including the modification of the map interface to include a terrain view and other public accessible layers of information. The Map mode can also be extended to include a feature that allows the user to get directions from the current location to the desired point of interest in the map.

On general terms, the UI interface is strictly designed to work with the GMP data. The inclusion of other entities would lead to a redesign of the UI and the application process flow to make it much more generic and extensible. Improving the overall visual elements, icons, widgets, layout and look and feel is a must needed improvement in order to provide a friendlier experience to the user and increase the probability of its adoption and use.

Security concerns were specified to be outside of the scope of the project. Once the application is extended and deployed on a production environment with real users and transactions, security elements have to be added to each layer and application component. Security access policies to the database, digital certificates to secure the connection between layers and the use of proper authentication policies to access the application data have to be implemented.

The current system specification design does not include any intelligent data filtering schemes. Additional real-time data analysis performed over the database can increase the probability of reduction in the number of searches and increase the accuracy of the most relevant queries. Including an abstraction layer over the data that provides an intelligent analysis based in different data manipulation mechanisms (found in big-data processing) can provide additional capabilities in real world scenarios.

Currently the system prototype in the mobile application is designed and implemented on the Android OS. Further work has to be done to port the mobile application to additional mainstream devices, including the latest versions of Apple's iOS and Microsoft Windows Mobile. Each port has to take into consideration the availability of the different sensors on which the mobile application has been implemented. Each mobile port also has to implement the capabilities to connect and consume the MW layer's services and other available information sources.

Further implementations have to take into account the inclusion of custom visual displays. An interesting alternative is to implement the mobile application using a bespoke headset to use near the user's range of vision to increase the capabilities of the information provided by the application. Another improvement is to use commercial products designed for augmented reality displays, including Google's glass project. While the glass project does not implement all the features to develop a full

AR display, it will be possible to do so in the future. Other devices that would be able to implement current functionality and extend it are META glasses[1], and possibly the Oculus Rift[2] device for virtual environments.

Overall, the amount of effort to implement the project has been considerable, however, it is an exciting research field and there is still significant effort to be done to increase the availability of the implemented technologies into everyday use. The different technologies that have been implemented can be used in other possible GMP needs. These needs include real time localization of fellow officers, display of common routes for police on the beat, among others. The application can be used by the GMP but can also be implemented by other police departments in the UK. The most interesting aspect of the implementation is related to the Augmented Reality mode. This mode, which as demonstrated by the amount of publicity the project glass has been receiving, is an attractive field of study. It also has enormous commercial applications that can be extended from the basic foundations provided in this dissertation.

It is possible to declare that this project has provided a more than adequate solution of the GMP's needs. It is also important to highlight that the project has added a good amount of research to the field of augmented reality applications on mobile devices. This has been made by providing a stable and easy to understand code structure and simplified calculations that can be used to provide a realistic experience to the user. It is expected that the application, and by extension the field of study, is improved further in the future and that the result of this project dissertation is more than enough platform for it.

[1] Meta View website: http://www.meta-view.com/
[2] Oculus VR website: http://www.oculusvr.com/

8. References

[1] D. Hauner and A. Kyobe, “Determinants of Government Efficiency,” *World Development,* vol. 38, no. 11, pp. 1527-1542, 2010.

[2] T. Guardian, “Cutswatch map: public sector cuts across the UK,” The Guardian, 12 11 2012. [Online]. Available: http://www.guardian.co.uk/society/cutswatch. [Accessed 15 02 2013].

[3] P. Arocena and D. Oliveros, “The efficiency of state-owned and privatized firms: Does ownership make a diference?,” *International Journal of Production Economics,* vol. 140, no. 1, pp. 457-465, 2012.

[4] Greater Manchester Police, “Policing in austerity: One year on,” 07 2012. [Online]. Available: http://www.hmic.gov.uk/media/greater-manchester-policing-in-austerity-one-year-on.pdf. [Accessed 12 02 2013].

[5] W. McCarty and L. Ren, “Determinants of Police Strength in Large U.S.Cities During the 1990s,” *Crime & Delinquency,* vol. 58, no. 3, pp. 397-424, 2009.

[6] West Midlands Police, “The Roles and Responsibilities of Constables,” West Midlands Police, 3 11 2012. [Online]. Available: http://www.west-midlands.police.uk/np/walsall/findoutmore/roles-responsibilities.asp. [Accessed 01 03 2013].

[7] UK Legislature, “Serious Organised Crime and Police Act 2005,” Legistalition UK - The National Archives, London, 2005.

[8] UK Legislature, “Firearms Act 1968,” UK Legislature - National Archives, London, 968.

[9] C. Brower, “Police officer duties & responsibilities,” eHow Family, 21 07 2011. [Online]. Available: http://www.ehow.co.uk/list_6539928_police-officer-duties-responsibilities.html. [Accessed 11 03 2013].

[10] IOS IP Mobility, CISCO, “CISCO Case Studies,” 07 2007. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/routers/ps272/prod_case_study0900aecd80695c7f.pdf. [Accessed 28 02 2013].

[11] H. Wolinsky, “Riding Radio Waves For 75 Years, Motorola Milestones,” *Chicago Sun Times,* 23 03 2012.

[12] C. Ennis, “Pocket Radio Pages Doctors Night Or Day,” *Popular Science,* p. 104, 1951.

[13] J. Tyson, “How Restaurant Pagers Work,” HowStuffWorks, 17 01 2010. [Online]. Available: http://electronics.howstuffworks.com/everyday-tech/restaurant-pager.htm. [Accessed 12 03 2013].

[14] London Ambulance Service, “Response to London Assembly 7 July Review Committee report,” London Ambulance Service - NHS, 05 06 2006. [Online]. Available: http://www.londonambulance.nhs.uk/news/news_archive/response_to_london_assembly_7.aspx. [Accessed 15 02 2013].

[15] E. Messmer, “Tech Talk: Where'd it Come From, Anyway?,” PCWorld, 29 06 2008. [Online]. Available: http://www.pcworld.com/article/147698/tech.html. [Accessed 02 04 2013].

[16] M. Shiels, “A chat with the man behind mobiles,” BBC News, 21 04 2003. [Online]. Available: http://news.bbc.co.uk/1/hi/uk/2963619.stm. [Accessed 01 03 2013].

[17] B. Greene, “38 years ago he made the first cell phone call,” CNN, 03 04 2011. [Online]. Available: http://edition.cnn.com/2011/OPINION/04/01/greene.first.cellphone.call/index.html. [Accessed 01 04 2013].

[18] A. Kilng, “Cell Phones - History,” in *Cell Phones*, Farmington Hills, Lucent Books, 2010, pp. 24-26.

[19] A. Kempe, “Mobile Phones 1980-1990,” Swedish National Museum of Science and Technology - tekniskamuseet.se, 29 07 2009. [Online]. Available: http://www.tekniskamuseet.se/mobilen/engelska/1980_90.shtml. [Accessed 28 03 2013].

[20] UMTS World, “UMTS / 3G History and Future Milestones,” UMTS World, 20 07 2006. [Online]. Available: http://www.umtsworld.com/umts/history.htm. [Accessed 12 02 2013].

[21] National Research Council (United States), “The Global Positioning System: A Shared National Asset,” National Academies Press, 1995.

[22] B. Stumpe and C. Sutton, “The first capacitative touch screens at CERN,” CERN Courier: International Journal of High Energy Physics, Munich, 2010.

[23] ABIresearch, “Touch Screen Controllers Bring Smartphone Experience to Feature Phones,” ABIresearch, New York, 2011.

[24] R. Kable, “Electrographic Apparatus”. United States Patent 4600807, 26 10 1984.

[25] Y. Lancet, “What Are The Differences Between Capacitive & Resistive Touchscreens?,” makeuseof, 19 07 2012. [Online]. Available: http://www.makeuseof.com/tag/differences-capacitive-resistive-touchscreens-si/. [Accessed 02 04 2013].

[26] elo TouchSystems, “Acoustic Pulse Recognition (APR),” tyco Electronics Press, Menlo Park, CA, 2006.

[27] A. Goff, “The first portable digital camera – cassette included,” *New Scientist,* vol. 213, no. 2855, pp. 28-29, 2012.

[28] G. Jungpen, A. Ronen and D. Brady, “Single-shot subpixel response measurement with an aperture array pixel mask,” *Optics Letters,* vol. 31, no. 23, pp. 3441-3444, 2006.

[29] M. Musgrove, “Nikon Says It's Leaving Film-Camera Business,” *The Washington Post,* pp. 1-2, 12 01 2006.

[30] A. Einstein and R. Lawson, Relativity the special and the general theory, New York: Henry Holt and Company, 1920.

[31] N. Katzakis, “Mobile devices as multi-DOF controllers,” in *IEEE Symposium on 3D User Interfaces*, Waltham, Massachusetts, USA, 2010.

[32] C. Barthold, K. Pathapati and R. Dantu, “Evaluation of Gyroscope-embedded Mobile Phones,” in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference*, Anchorage, AK, 2011.

[33] M. Graham, M. Zook and A. Boulton, “Augmented reality in urban places: contested content and the duplicity of code,” *Transactions of the Institute of British Geographers,* vol. 2012, no. 00539, p. 1475, 2012.

[34] J. Rolland, F. Biocca, F. Hamza-Lup and H. Yanggang, “Development of Head-Mounted Projection Displays for Distributed, Collaborative, Augmented Reality Applications,” *Presence: Teleoperators & Virtual Environments,* vol. 14, no. 5, pp. 528-549, 2005.

[35] K. Grifantini, “Augmented Reality Goggles,” MIT Technology Review, 10 09 2010. [Online]. Available: http://www.technologyreview.com/news/421606/augmented-reality-goggles/. [Accessed 01 04 2013].

[36] P. Milgram, H. Takemura, A. Utsumi and F. Kishino, “Augmented Reality: A class of displays on the reality-virtuality continuum,” in *SPIE 2351, Telemanipulator and Telepresence Technologies*, Boston, MA, 1995.

[37] M. Tidwell, R. Johnston, D. Melville and T. Furness, “The Virtual Retinal Display - A Retinal Scanning Imaging System,” *Proceedings of Virtual Reality World '95,* pp. 325-333, 1995.

[38] C. Reid, “Turning print into an interactive storefront: Layar's augmented-reality technology gives life to print pages,” *Publishers Weekly,* vol. 260, no. 8, p. 2, 2013.

[39] C. Davies, “Quantigraphic camera promises HDR eyesight from Father of AR,” Slashgear, 12 09 2012. [Online]. Available: http://www.slashgear.com/quantigraphic-camera-promises-hdr-eyesight-from-father-of-ar-12246941/. [Accessed 12 04 2013].

[40] C. Dillow, “BMW Augmented Reality Glasses Help Average Joes Make Repairs,” Popular Science, 03 09 2009. [Online]. Available: http://www.popsci.com/scitech/article/2009-09/bmw-developing-augmented-reality-help-mechanics. [Accessed 18 03 2013].

[41] M. Livingston, A. Zhuming, K. Karsch and G. Gibson, “User interface design for military AR applications,” *Virtual Reality,* vol. 15, no. 2-3, pp. 175-184, 2011.

[42] A. Webster, S. Feiner, B. MacIntyre, W. Massie and T. Krueger, “Augmented Reality in Architectural Construction, Inspection, and Renovation,” in *ASCE Computers in Civil Engineering Congress*, Columbia, 2005.

[43] S. Ong, M. Yuan and A. Nee, “Augmented reality applications in manufacturing: a survey,” *International Journal of Production Research,* vol. 46, no. 10, pp. 2707-2742, 2008.

[44] H. Wu, S. Lee, H. Chang and J. Liang, “Current status, opportunities and challenges of augmented reality in education,” *Computers & Education,* vol. 62, pp. 41-49, 2013.

[45] K. Lee, “Augmented Reality in Education and Training,” *TechTrends,* vol. 56, no. 2, pp. 13-21, 2012.

[46] D. Sumadio, “Preliminary Evaluation on User Acceptance of the Augmented Reality Use for Education,” *Computer Engineering and Applications (ICCEA), 2010 Second International Conference IEEE,* vol. 2, pp. 461-465, 2010.

[47] Y. Lim, Y. Park, J. Heo, J. Yang, M. Kang and Y. Byun, “A Smart Phone Application Based on AR for Jeju Tourism,” in *ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, Jeju, South Korea, 2011.

[48] S. Perry, “Wikitude: Android App With Augmented Reality: Mind Blowing,” digital-lifestyles.info, 23 10 2008. [Online]. Available: http://digital-lifestyles.info/2008/10/23/wikitude-android-app-with-augmented-reality-mind-blowing/. [Accessed 12 04 2013].

[49] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier and B. MacIntyre, “Recent Advances in Augmented Reality,” Georgia Tech College of Computing - Computer & Graphics, Atlanta, 2001.

[50] Apple Inc., “App Store Tops 40 Billion Downloads with Almost Half in 2012,” Apple Press, Cupertino, California, USA, 2013.

[51] E. Hutchings, “Sayduck aim to solve the missing link between online and offline retail by letting customers visualize objects in their intended environment,” PSFK, 18 05 2012. [Online]. Available: http://www.psfk.com/2012/05/augmented-reality-furniture-app.html. [Accessed 03 04 2013].

[52] M. McDonough, “Lego’s Augmented Reality App Shows You What’s Inside the Box,” Laptop Magazine, 15 09 2011. [Online]. Available: http://blog.laptopmag.com/legos-augmented-reality-app-shows-you-whats-inside-the-box. [Accessed 13 03 2013].

[53] SAP AG, “Give your sales team the tools they need to wow their customers – with our CRM sales mobile app,” SAP Highlights - Mobile Solutions, [Online]. Available: http://www54.sap.com/pc/tech/mobile/software/lob-apps/sales-crm-app/index.html. [Accessed 21 04 2013].

[54] R. Azuma, “A Survey of Augmented Reality,” *Presence: Teleoperators and Virtual Environments,* vol. 6, no. 4, pp. 355-385, 1997.

[55] Greater Manchester Police, “Police.uk API documentation,” [Online]. Available: Police API documentation. [Accessed 13 03 2013].

[56] Greater Manchester Police, “POLICE.UK APPS,” [Online]. Available: http://www.police.uk/apps. [Accessed 21 03 2013].

[57] C. Larman, Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development, Upper Saddle River, N.J: Prentice Hall PTR, 2005.

[58] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Reading, Mass: Addison-Wesley, 1995.

[59] E. Freeman, E. Freeman, B. Bates and K. Sierra, Head First Design Patterns, Sebastopol, CA: O'Reilly Media, 2004.

[60] J. Scholtz, “Usability Evaluation,” National Institute of Standards and Technology, Maryland, USA, 2009.

[61] J. Brooke, “SUS - A quick and dirty usability scale,” Redhatch Consulting Ltd, Reading, UK, 1996.

[62] J. Sauro, “Measuring Usability With The System Usability Scale (SUS),” 2011. [Online]. Available: http://www.measuringusability.com/sus.php. [Accessed 21 04 2013].

[63] E. Codd, “A Relational Model of Data for Large Shared Data Banks,” *Communications of the ACM,* vol. 13, no. 6, p. 377–387, 1970.

[64] Ordinance Survey, “A guide to coordinate systems in Great Britain,” Crown, Southampton, SO16 4GU, 2010.

[65] Google Inc., “Design Principles,” Android Developers, [Online]. Available: http://developer.android.com/design/get-started/principles.html. [Accessed 15 04 2013].

[66] J. Nielsen, “Usability Engineering,” Academic Press, London, 1993.

[67] The Eclipse Foundation, “Web Tools Platform,” Eclipse, [Online]. Available: http://www.eclipse.org/webtools/. [Accessed 10 03 2013].

[68] Google Inc., “ADT Plugin,” Android Development Tools, [Online]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html. [Accessed 15 03 2013].

[69] Google Inc., “Sensor Overview,” Android Developers, [Online]. Available: http://developer.android.com/guide/topics/sensors/sensors_overview.html. [Accessed 27 02 2013].

[70] J. Stott, “Easily convert between latitude/longitude, Universal Transverse Mercator (UTM) and Ordnance Survey (OSGB) references with Java using the Jcoord package,” Jcoord - Jonathan Stott, 11 02 2006. [Online]. Available: http://www.jstott.me.uk/jcoord/#download. [Accessed 26 05 2013].

[71] Google Inc., “Building a Simple User Interface,” Android Developers, [Online]. Available: http://developer.android.com/training/basics/firstapp/building-ui.html. [Accessed 10 02 2013].

[72] Google Inc., “Managing the Activity Lifecycle,” Android Developers, [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/starting.html. [Accessed 13 02 2013].

[73] S. Colton, “The Balance Filter, A Simple Solution for Integrating Accelerometer and Gyroscope Measurements for a Balancing Platform,” MIT Education, Boston, Mass, 2007.

[74] D. Sachs, “Sensor Fusion on Android Devices, A revolution in Motion Processing,” Google Inc. Tech Talks, 2010. [Online]. Available: http://www.youtube.com/watch?v=C7JQ7Rpwn2k. [Accessed 15 05 2013].

[75] P. Lawitzki, “Android Sensor Fusion Tutorial,” Thousand Thoughts, 2012. [Online]. Available: http://www.thousand-thoughts.com/2012/03/android-sensor-fusion-tutorial/. [Accessed 20 05 2012].