Universitetet
i Stavanger

**FACULTY OF SCIENCE AND TECHNOLOGY**

# MASTER'S THESIS

| | |
|---|---|
| Study program/specialization:<br>Master in Computer Science | Spring semester, 2017<br>Open |
| Author: Jhon Jairo Gutierrez Lautero | …………………………………………<br>(signature author) |

Instructor: -

Supervisor(s): Erlend Tøssebro

Title of Master's Thesis: Augmented Reality Browser for Android Smartphones
Norwegian title: -

ECTS: 30

| | |
|---|---|
| Subject headings: Augmented reality, android app development | Pages: 68<br>+ attachments/other: source code (ARBrowser.7zip)<br><br>Stavanger, 15 June / 2017<br>Date/year |

University of Stavanger

Master's Thesis

# Augmented Reality Browser for Android Smartphones

*Author:*
Jhon Jairo Gutiérrez Lautero

*Supervisor:*
Erlend Tøssebro

*A thesis submitted in fulfilment of the requirements for the degree of Master in Computer Science*

*in the*

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

June 15, 2017

# Abstract

Augmented Reality capabilities complement the reality captured by the sensors (human's senses or technological devices) by adding virtual elements that improve the experience with the real world.

The purpose of this project is to develop an application for android smartphones with Augmented Reality (AR) characteristics. Principal characteristic includes an AR view location-based, by using the GPS, camera, and the sensors such as accelerometer, magnetic field, and gyroscope. In addition, the integration of different open data sources that provide relevant information of points of interest (POI's) like touristic places, natural locations, and local business. Based on the searching filters and coordinates of current position.

For a better understanding, the mathematical foundation will be explained (three axes space x, y and z) for calculate positioning of markers that will be projected in the augmented reality view, by processing and filtering the data provided by the sensors, as well, an explanation of the libraries used to develop the different components of the application.

After this, a revision to the accuracy of sensor methods for capture the localization and motion changes, by using low/high pass filters in the signals and fusion of sensed data, to cancel noise and drift, and get more accurate orientation vectors.

The results of this project are presented in form of an application with AR capabilities, such that it can be used for smart and interactive browsing, and to understand the basis of AR concept applied in a real use case.

Further research, in this project is to present additional helps in the dashboard by using image recognition, for interact with the objects around, and contextualized searching based on filtering history and user preferences.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Over the recent years, the increasing development of the technology has improved the characteristics of the smartphones, allowing the creation of new and better applications to support the daily activities of people.

The aim of this project is to develop a browser for android smartphones with an AR layer, that complements the information captured from the real-world by using the motion sensors as well as GPS and camera.

The development of an AR browser for smartphones improve the experience of search for places, by allowing users to interact with the results and obtaining a visual guide.

The AR browser will provide contextualized information by retrieving places details with a significant level of accuracy and precision with low latency in the responses, based on the user's location and filtering search options.

Aim of this project is to present an improved version of previous jobs of AR app[1] and another Android applications as Yelp and Wikitude (ongoing projects explained in section 3.4 Related work), by integrating gyroscope sensor and more options to configure the AR browser.

## 1.1 Objectives

1. Develop a smartphone application for Android platform with AR capabilities that use the information provided by the sensors of the device with acceptable level of accuracy.
2. Design and implement an AR browser to present information based on geolocation, by creating a dashboard to show the POI's where user can interact with the results and get access to detailed information through a visual guide in the phone screen.
3. Design and implement a backend system to interconnect different location data sources, i.e. Google Places API and open use geographical databases, to consume the information provided for them and present the results by projecting markers in the camera screen and map view.
4. Define a reliable service to save new places and user preferences, and allow to filter results by defining a configuration panel to setup all the components (data sources, place types, sensors).

## 1.2 Report organization

This report presents a contribution in the AR field, by developing an Android App with AR capabilities. Through the evaluation of current technologies, it is proposed an improved design and implementation of an AR Browser. In addition, review of all the background related to augmented reality as technical foundation, technologies, and ongoing projects.

This report is organized as follows. Section 1 presents an introduction to AR and the objectives of the project, Section 2 create a context by presenting a background and theoretical foundation.

Section 3 presents related ongoing projects, by reviewing technologies and related work, Section 4 illustrated an analysis of the solution to implement.

Section 5 presents the proposed design, Section 6 subsequent implementation of an AR Browser and technical details of interest.

Section 7 presents an evaluation with some additional findings and 8 presents final conclusions.

# 2 Background

Augmented reality (AR) merges real-world interaction with virtual objects, through the recognition of images and objects as well as providing real-time context and data [2].

AR tools enriched the information captured from the real-world. Information is generated in form of video, audio, or coordinates (geolocation) and is "augmented" by adding virtual elements in an upper layer that complements it.

The virtual elements are computer-created entities such as an interface overlapping a video (a kind of dashboard), a 3Ds objects interacting in a surface, a 2D map showing Points of Interest (POI).

There are several systems that provided information in a wide range of categories like local businesses and touristic places, those systems allow interconnection by an API's through REST services invocation, for example Google Places and Twitter API, and open access geographical database like GeoNames and Google Maps API for Geocoding and Elevation.

The section 2 makes an introduction to the AR technology, by explaining the different approaches that enriched the perception of the world in different sectors like education, health, business. Also, settled the main components that compound a basic implementation of an AR application as the sensors, GPS, and camera. In addition, it is included a theoretical foundation to understand the data provided by the sensors and the way to determine location and rotation of the device, as well methods to improve the accuracy of the collected data.

Additional components have been used to improve the characteristics of the AR Browser, for example gyroscope to enhance the location in the visualization of the markers, complementary filter to avoid errors in the signal, API's to collect and transform data, and Android version 7.1 API level 25[3] to develop a configurable dashboard to project the results of search.

In the project, it is included methods to improve the accuracy of the data, due to previous version of the application presents problems respecting to the visualization of the markers. Those problems include jittery that will be solved by smoothing the data using filters and methods to fused data collected from sensors as gyroscope and accelerometer.

## 2.1 AR paradigm

AR is a new view of the world with enriched components, by complementing the reality with virtual elements that provided a new experience to the people, helping to create new experiences and forgetting all the additional technological elements on which it is supported.

Another important fact is the permanent connectivity that promote the access to the information in real time, enhancing all those technologies that complement the reality through the easy and instantaneous access to the information, provided by the current communications infrastructure.

Nowadays, AR is understood like a Mixed Reality (MR)[4], between real and virtual elements, where predominates the real information. In the Figure 1, it is illustrated the relation between those elements. From left to right, the real world is complemented with virtual elements to enriched a scenario and enhance the user experience, in the other side a virtual world is created and added real elements to create an Augmented Virtuality (AV).

Those two worlds (AR and AV) can converge in a mixed reality where Virtual and Real elements can be display in a same view, complementing each other, and decreasing the limitations that can exist in the two worlds.

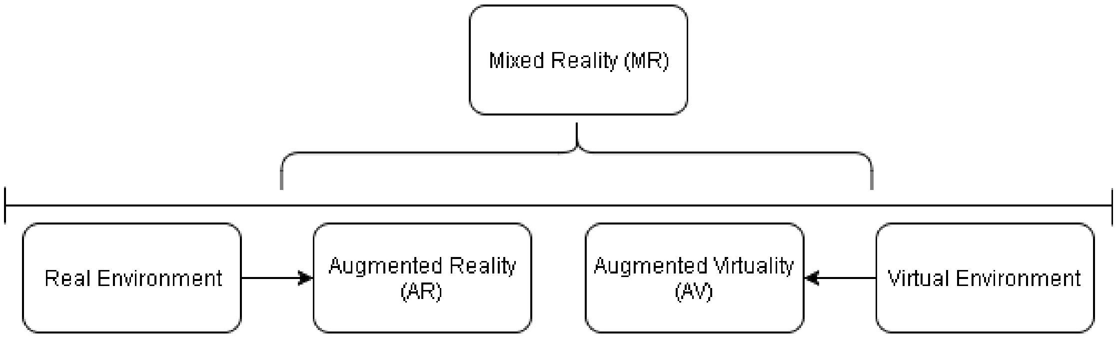


Figure 1: Representation of a Reality-Virtuality Continuum

### 2.1.1 Applications of AR

AR improves the processes of several areas by providing new software or hardware to execute regular tasks. Some representative areas of use are listed below.

- Education: To provide novel pedagogical experiences to obtain more knowledge by using tools to explore information and discover new methods to teach.
- Health: A susceptible area where the use of AR technologies demands big amounts of data, to provide a reliable context for decision-making to the professionals in this area.
- Tourism: To promote touristic places and develop guides by collecting relevant data, based on filters or user preferences.
- Marketing: Many companies used AR to highlight their products and be more attractive than the competence.
- Real time data: The use of data provided by social networks to share experiences and information.

### 2.1.2 Technological foundation

Wide range of technologies have been continuously developing to support the capabilities of AR, in relevant fields as Internet-Of-Things (IoT), Cloud Computing, or Telecommunication Services. Those technologies can be grouped and explained to understand how it works, and validate its benefits.

**Monitoring or tracking.** This kind of technologies are widely used to developed AR applications, due to the huge offer of devices to sense and record the environment. The principal tracking technologies are based on sensors, vision, and hybrid implementations.

- Based on sensors: Collecting data using sensors like Gyroscope, Accelerometer, Compass, and GPS to measure and control the environment.
- Based on vision: Two groups in this kind of technologies can be based on markers or image recognition.
- Hybrid: High advanced systems that provide fused technologies allowing to develop better AR applications, a clear example are the smartphones that integrate different sensors to enhance user experience.

**Interaction.** Technologies that offers a real interaction process between the users and the virtual world to present a significant experience by offering additional elements to make an intuitive communication.

- Based on markers: These technologies are based on the use of a camera to perceive a specific visual point, with this kind of tools real elements with special markers (QR codes, patterns) have been placed. When the AR system detects the markers, a virtual element will be displayed.
- Corporal Motion detection: By using corporal motion recognition is avoided the use of additional elements to trigger the virtual scenario, these technologies are closely related to image recognition due to the process of acknowledgment of elements like body parts (head, hands).

**Display Technologies.** Are used to present the fused information (Real/Virtual) in a screen like a smartphone, TV, and to project images in real objects (holograms, dashboards). Some examples of those technologies are listed below.

- 3D screens: Taking advantage of the increasing development of technologies in screens for wide variety of devices.
- Glasses and Lens: Another more sophisticated technologies that provides futuristic gadgets to use for developing AR applications.

### 2.1.3 Components related with an AR application

In the development of the project is defined three main components, that will be used to reach the objectives and to settled the principal characteristics of the App: Global Positioning System (GPS), sensors and camera, and data sources[5].

1. Sensors

The hardware provided by the smartphone facilitates the recognition of motion, environment, and location changes. That hardware are sensors that provide large amounts of data in form of a signal. In most of the cases, the signal captured from the sensors include certain level of

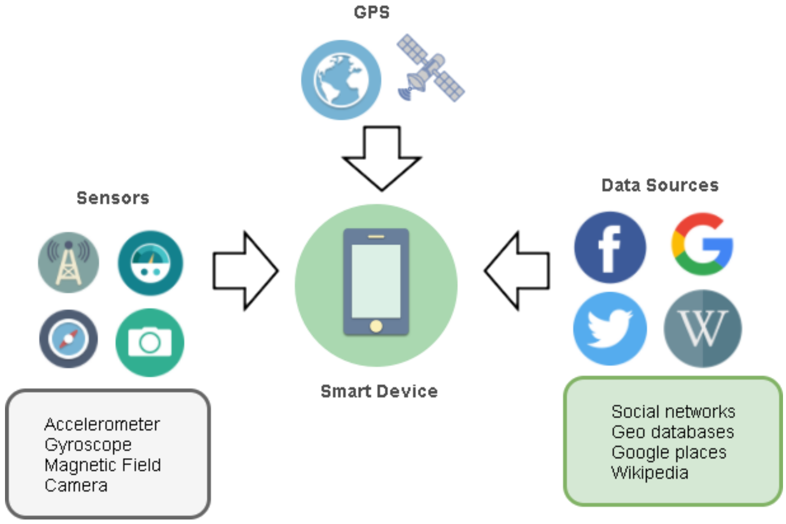


Figure 2: Main components

noise. The noise represents undesired changes that the signal can suffer in different phases of processing: sensing, storing, or transmitting.

The principal issue in the development of AR application is to present an experience using real-time data and making feel users close to a real scenario, by managing the inaccuracy presented in the data collected by the sensors.

Additional components like filters or algorithms that fuse sensor data need to be implemented to get high levels of precision and accuracy in the final results. Some sensors, like the compass, also requires frequent calibration.

The data collected from the sensors will be used to detect changes in the position and rotation of the smartphone. Then is used to correctly place the virtual elements in the camera view. This means that the application needs to know both position and orientation of the smartphone in space with a good level of accuracy. Position is obtained by using GPS localization and the orientation is based on the data provided by sensors, the use of those sensors is described below.

For this particular development, an API called Android Augment Reality Framework (ARF)[6] will be used. ARF offers the needed features to create an augmented reality browser for android devices.

ARF works by using two specific sensors: Magnetic field, used to measure the ambient geomagnetic field (common use to create a compass); and Accelerometer, used to measure the acceleration force, applied to a smartphone over the three axes (x, y and z).

To complement and improve the characteristics presented by the ARF the gyroscope (calibrated) was also used. The gyroscope measures the rotation of the device around the three axes. By using the gyroscope, the precision of the orientation measurements was improved.

2. Camera

The camera in the smartphone will be used to capture the real world, by showing the surrounding of the user and being complemented with virtual elements, more specifically markers that present point of interests (POI's) and it will be projected in an overlapping layer in the video.


Figure 3: Camera view with overlapped markers


Figure 4: Location enabled with GPS

3. Global Positioning System (GPS)

By enabling the use of GPS, it allows the application to obtain the current location and provide to the user the global positioning. Using the coordinates X and Y (longitude and latitude) is possible to show to the user nearby places by filtering based on the current location.

## 2.2 Determine orientation

This section of the report will explain the motion sensors, they describe the behaviour of the device itself, by capturing the changes in the rotation of the smartphone. Each of those values are presented in different units, and are applied in the three physical axes x, y and z. To understand the values given by the sensors is important to understand the coordinates system: Global and device coordinate system.

### 2.2.1 Device orientation and rotation (movement)

The sensor framework[7] (important package of Android API), is based on the three axes coordinates system to present the captured values. Majority of the sensors follow the coordinate system illustrated in the Figure 5 and are regulated for the screen position.

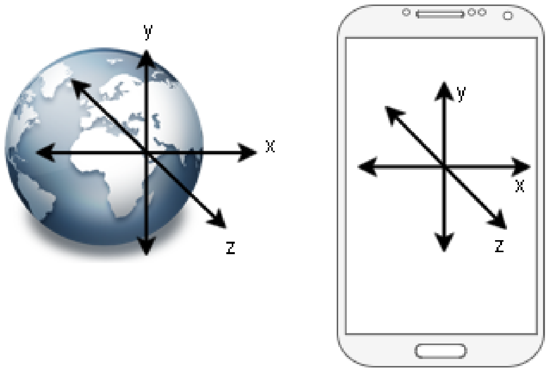


Figure 5: Global and device coordiante system

One important fact to remark is that, when is changed the orientation of the phone (portrait/-landscape) the axes remains the same, the orientation of the smartphone can change and affect the calculations of the angle view in the camera, for this project will be managed this behaviour by validating the orientation and remapping the rotation matrix.

Device coordinate: The three sensors: accelerometer, gyroscope and magnetic field that will be used, return the sensed data according to this system, where, in the default orientation of the smartphone (portrait) the information will be given with the following boundaries respect to the axes: X axis is horizontal with positive values pointing right, Y axis is vertical with positive values pointing up, and Z axis is perpendicular respect to the screen with positive values in the front direction.

Global Coordinate: Using an absolute orientation with respect to the earth (north pole), to obtain the rotation vector, mapping from the device coordinate system to the global system.

The coordinates are like this: Y axis points to the magnetic north, X axis is parallel to the surface of the earth and Z axis perpendicular to the surface of the earth.

According with the 3 values given for the accelerometer and gyroscope, in the physical 3 axes is possible to calculated the rotation angles also called: azimuth or heading, pitch and roll.

Sensors units: As previously mentioned, it is important to understand the units of the data provided by the sensors to use correctly that information. The values presented was captured in the testing of the application by logging the data sensed.

The chip (integrated sensor) reference is `MPU-6500` (six axis motion tracking, gyro + accelerometer), integrates a 3-axis accelerometer and a 3-axis gyroscope, useful for many different applications addressed for high performance, due to the low consume of battery.

MPU-6500 chip is a Microelectromechanical system (MEMS)[8]. This kind of chips provides several benefits in applying motion tracking capability by integrating multiple sensors[9]. It is one of the smallest motion tracking devices developed by InvenSense[10]. It is used in a wide variety of devices including IPhone 6 and previous versions of Samsung Galaxy smartphone.

MPU-6500 provides significant inertial sensing capabilities, it was detected in the smartphone where the application's test was executed. Check reference datasheet[1].

- Accelerometer reports all the values in $m/s^2$. The data provided by this sensor is ruled by earth gravity. Where it is a constant equal to $+9,8m/s^2$ (approximately) equivalent a 1g. The events in this sensor report the gravitational force values as follow: `value[0]` in the `X` axis , `value[1]` in the `Y` axis, `value[2]` in the `Z` axis; the normal force being applied in the smartphone without shaking strongly is in the range of $+-2g$.

  The sign (positive or negative) of the values reported by this sensor depends on the current orientation of the smartphone: Lying on a horizontal surface (screen up or down), Portrait and Landscape (normal or upside down).

- Gyroscope reports all the values in $radians/second(rad/s)$, representing the changes in the rotation of the smartphone over the time. By sensing the Coriolis force [11] applied in every axis in the device, this force just acts when the device is in motion, in other case it will report zero (or very close to zero).

  The data provided for this sensor is useless by itself to calculate orientation angles, it could be done by integrating data over the time, but the gyroscope noise can introduce undesirable errors in the calculated angles. To avoid these problems and generate better results the data provided by this sensor is integrated with the accelerometer.

- Magnetic Field reports all the values in $micro-Tesla(uT)$. Detecting earth magnetism is commonly used for compass application, it has some disadvantages depending of the environment where is located, close to pieces of metal or similar things can produce variations in the sensing.

  Chip `AK09911C` was detected in the testing smartphone, it is developed by a leading manufacturer of 3-axis electronic compass[12]. Reference datasheet[2]. Some constants associated to this sensor are `MAGNETIC_FIELD_EARTH_MAX` equals to $60uT$.

  By providing coordinates respect to the location (latitude, longitude and altitude) and a time to instantiate it is possible to use the class `GeomagneticField` from the Sensor API, then has access to methods to calculate the declination (respect of magnetic and true north) and inclination for a determined point.

### 2.2.2 Mathematical foundation

The sensors that will be used in the project are regulated under the three physical axes, where they conform a vector of size three. The information content in the vector are magnitudes like acceleration, rotation, and magnetism associated with each axis: $x$, $y$ and $z$.

The vector setup a three-dimensional cartesian coordinate system (Figure 6), the set of that 3 locations is known as Euclidean space, where is commonly represented by $R^3$.

To project the elements in the augmented view (overlap virtual markers on camera capture) is necessary to know the device orientation. The information provided by the `ACCELEROMETER` in union with the `MAGNETIC_FIELD` or relative orientation with the `GYROSCOPE` is used to calculate the angles. With the data provided by the sensors is calculated the rotation matrix and posterior uses this information to obtain the orientation.

[1]Datasheet: `https://www.invensense.com/products/motion-tracking/6-axis/mpu-6500/`
[2]Datasheet: `https://www.akm.com/akm/en/file/datasheet/AK09911C.pdf`

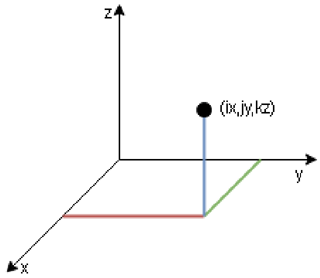


Figure 6: Cartesian coordinate system

Rotation matrix: Consist of a 3-unit vector of size three, that conform a $3x3$ matrix, the rotation matrix R express a movement in three dimensions, rotation of a body is described by a change in some of the three axes.

A basic rotation is a "rotation" about each axis $x$ ($\theta$), $y$ ($\alpha$), and $z$ ($\beta$) represent the azimuth, pitch and roll correspondingly. The rotation matrixes are presented below:

- Table 1 $Rx(\theta)$ represents counter-clockwise rotation at -90°around the x axis.
- Table 2 $Ry(\alpha)$ represents counter-clockwise rotation at -90°around the y axis.
- Table 3 $Rz(\beta)$ represents counter-clockwise rotation at -90°around the z axis.

Table 1: Rotation in axis x

| | | |
|---|---|---|
| 1 | 0 | 0 |
| 0 | cos $\theta$ | -sin $\theta$ |
| 0 | sin $\theta$ | cos $\theta$ |

Table 2: Rotation in axis y

| | | |
|---|---|---|
| cos $\alpha$ | 0 | sin $\alpha$ |
| 0 | 1 | 0 |
| -sin $\alpha$ | 0 | cos $\alpha$ |

Table 3: Rotation in axis z

| | | |
|---|---|---|
| cos $\beta$ | -sin $\beta$ | 0 |
| sin $\beta$ | cos $\beta$ | 0 |
| 0 | 0 | 1 |

Orientation vector: Once the rotation matrix is obtained, the orientation vector can be calculated. To get the value for azimuth, pitch, roll, and current angle view. The orientation vector can be obtained directly from the Android Sensor API, or it can be calculated manually by computing a matrix multiplication to get more accurate angles. Conversion from rectangular coordinates to polar coordinates (Figure 7), by computing the arc tangent in the range of -$\pi$ to $\pi$, with the values provided by the accelerometer:

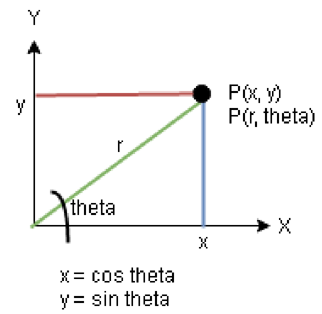


Figure 7: Conversion rectangular to polar coordiantes

$$Roll = atan2(x, z) * 180/\pi$$
$$Pitch = atan2(y, z) * 180/\pi$$
$$Azimuth = atan2(x, y) * 180/\pi$$

To project the points in the screen is used 4 components: The rotation matrix, the marker's location, the smartphone location (composed by longitude, latitude, and altitude) and the camera view.

### 2.2.3 Used sensors overview

To developed this project was used a variety of motion sensors to detect changes in the movement. One big challenge was to understand and analyse the amount of data provided by the sensors in a short range of time. All the information captured by the sensors (like the location data) was managed through Android API, where it provides a series of components (Sensor API) to make easy the use of those data, below a short explanation of those components. Sensor API provides access to the following types (relevant in this project):

- `Sensor.TYPE_ACCELEROMETER`: Provides the acceleration, gravity force in each axis x, y and z, in units of meter per square second [m/sec2]
- `Sensor.TYPE_GYROSCOPE`: Provides the angular speed around axis x, y and z, in units of radians per second (rad/sec).
- `Sensor.TYPE_MAGNETIC_FIELD`: Provides the ambient magnetic field for all the three physical axes x, y and z, in units of micro tesla (uT).

Accelerometer, magnetic field, and gyroscope are hardware based sensor, what it means is that the accuracy of the information provided depends of the quality of the hardware. In the other hand, software based sensors are a fusion between using the data from one or more physical devices and some additional calculation to provide the final data. For example, the sensor `Sensor.TYPE_ORIENTATION` (it is deprecated in the API level 8) uses the gravity and the geomagnetic sensor to get the orientation and rotation matrix.

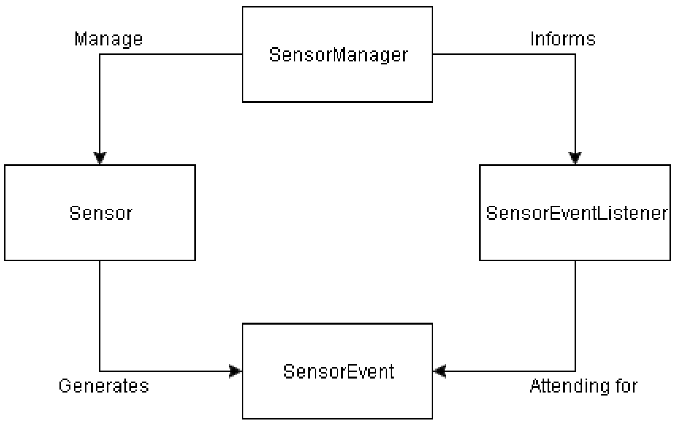


Figure 8: Sensor componentes

- SensorManager: This component provides access to the available sensors in the smartphone.
- SensorEventListener: Listener in charge of receiving new notifications when detect incoming data from the sensors, and to inform those events to the layer that implements it.
- SensorEvent: Contains the values of an event occurred in a specific time.
- Sensor: This abstraction contains all the raw data (values) provided for a specific sensor, the type of value provided for each one vary between sensors.

### 2.2.4 Calculating orientation

To obtain the orientation angles(Figure 9) in the device is necessary to use the rotation matrix. The rotation matrix is calculated from the event values of the sensed data from the accelerometer and gyroscope. The matrix is passed to a method that return the orientation vector. Vector contains the following information:

- Azimuth (also known as yaw) represents rotation over the z axis, laying the smartphone in the back and pointing in portrait mode to the north reports x equals to 0°, 90°to the East, 180°to the south and 270°to the West, fulfilling the following inequality, $0° <= Azimuth < 360°$.
- Pitch represents rotation over the x axis, returning positive values when the z axis moves in the direction of the y axis, fulfilling the following inequality, $-180° <= Pitch <= 180°$.
- Roll represents rotation over the y axis, returning positive values when the z axis moves in direction of the x axis, fulfilling the following inequality, $-90° <= Roll <= 90°$.

### 2.2.5 Calculating inclination

Geomagnetic field is the magnetism irradiated from the interior to the outside of the earth, creating a kind of shield to protect earth from solar irradiation, the inclination is the angle in radians of those magnetic field.

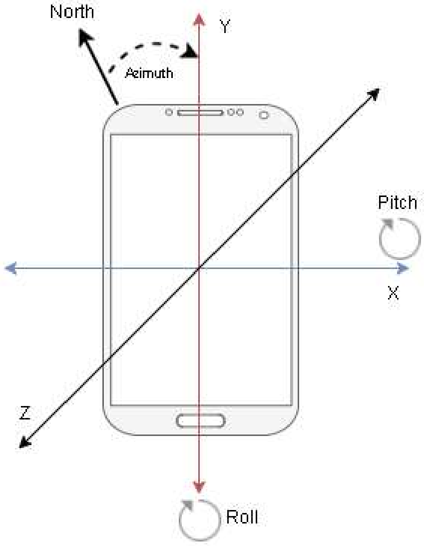


Figure 9: Three axes rotation

That shield is not regular in all the areas in the earth, to compute the geomagnetic inclination is used the vector values from the magnetometer and the rotation matrix.

#### 2.2.6 Fuse sensed data to improve calculations

Fused data is a better alternative to improve the calculations when using the information captured by the sensors, by merging the information provided by 2 or more sensors to deliver more accurate information. In our case, it provides a more precise and in real-time orientation (angles), to present a view with smoothed transitions of the projected markers when the camera is rotating.

Several vendors have developed chips (like the one installed in our testing smartphone, `MPU-6500`) integrating different sensor like gyroscope and accelerometer, additional to this they have developed special fusion algorithms to correct problems of noisy, delay and drift in signals.

Some of the algorithms include filtering (low and high pass filters to attenuate or amplify the signal), use of smoothing parameters, and average (mean) of the data in certain period.

### 2.3 Determine location

The development of the application is based on the use of Android API. It provides a series of services to access the data to get current coordinates.

Figure 10 illustrates a high-level relation between the components (java classes) that are used from the API.

- LocationManager: The principal class to manage the location in android devices, it enables to receive updates of the current location and offering additional methods to provides information about the current state like available providers.

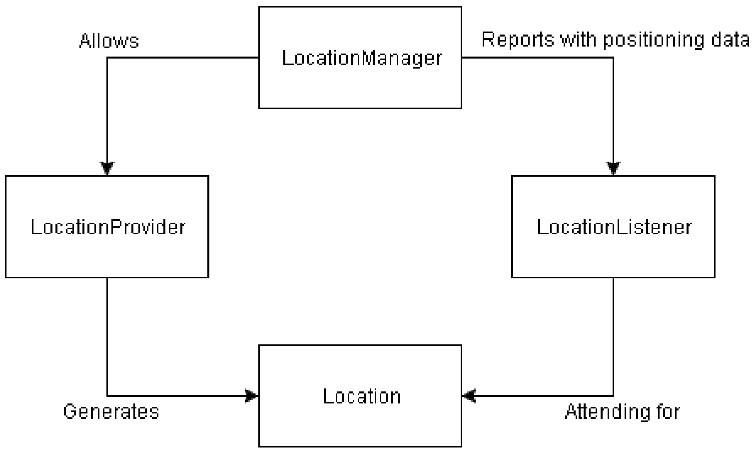


Figure 10: Location componentes in Android API

- LocationProvider: Manages the available providers to acquire the different location sources.
- LocationListener: In charge of being waiting for changes in the location and trigger the corresponding alerts.
- Location: Abstraction of current positioning of the device, containing the longitude, latitude, and altitude.

### 2.3.1 Methods to acquire location

In the development of this project to determine current location is used the information provided by mechanisms called global positioning. By using the android platform, the location is determined with two different providers: GPS and Network provider[13].

**GPS Provider** GPS Provider consists on a system composed by satellites that orbit around the earth, to send information to receivers (in this case an android device) to determine their location. The Global Positioning System commonly called GPS is also composed for some control stations that fix and adjust the information send it to the receivers.

The receivers calculate the position by determining the distance between the satellites that previously send data, by making a triangulation, to calculate the current position. In addition, receivers manage a high accurate time mechanism to have a synchronous and real time communication.

Some limitations with getting the location based on GPS signal is that it needs a clear path (way) between the satellites and the smartphone. This causes problems working indoor or outdoor situations where the sky is not clear visible (obstacles like trees in a forest or another natural location with difficult access), it could generate errors in the information and a poor location calculation.

**Network Provider** Network provider works by using or connecting to a cellular or Wi-Fi network that provides enough information to acquire the current location. Android devices can resolve locations based on this mode, by activating the Wi-Fi radio receiver or by having access to a cellular network.

The smartphone can request the location to Google Location Service by using a Wi-Fi access point. Is necessary to grant some privileges for enabling transmit and record the positioning information and have a constant update.

Like getting the location from a Wi-Fi access point, by using a cellular network the location is obtained from a cellular tower. The smartphone can use a triangulation mechanism if it has access to many towers

One of the biggest benefits by using network provider is that complements the use of the GPS when the signal is poor, and reduce the battery consumption, since the cost of use a Wi-Fi receptor is much less.

In the case of Wi-Fi networks, there may be some problems in the accuracy of the positioning if a wireless access point change the location, or in the case of cellular network, if the smartphone has not enough credit (data service) to request or receive information.

### 2.3.2 Location strategies

To present a high-level accuracy of current location and use the resources in an efficient way (battery saving, data services charge) is necessary to contemplate the following points:

- Check which provider is the most suitable Network or GPS provider.
- Validate when to start listening for updates, and when stop reads from sensor once the information is get it (register and unregister methods).
- Implement Fast fix with the last location when service is not available, by making a comparison in the accuracy between previous and current values.

## 2.4 Managing errors and signal processing

In the development of this project different sensors will be used, to select the most suitable components, for detect and correct the possible errors presented in the signal processing.

Two important concepts need to be cleared up to determine when there are errors in the data sensed: Accuracy and Precision (Figure 11).

Accuracy involves how close is the system to provide the correct value or result; in the other hand, precision is how consistent or how close are the results of different tests when we use the same methods or algorithms. The accuracy is improved by using tools or mechanisms that make the system being calibrated correctly, and the precision is increased by implementing incremental tools that require less estimation, it could be checked by repeated measurements.

### 2.4.1 Type of possible problems

Main errors (could exist more) taken into consideration to mitigate its consequences, in the development of the project are:

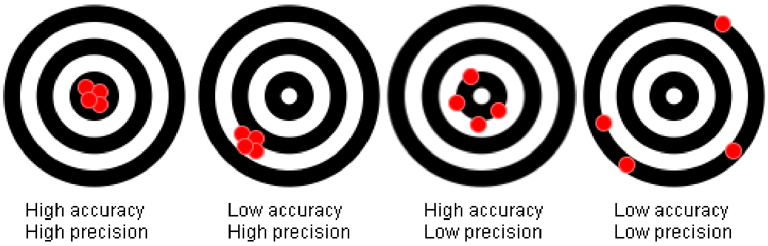


Figure 11: Precision and accuracy

- Noise: The noise in the signal indicates a random variation of the data collected by the sensors, this situation can be solved by implementing filters to eliminate all the frequencies out of the desired range. Filters can attenuate or amplify the signal by multiplying the values by a rectangular function in the same domain or an alpha parameter.

  This problem cause jittering in the view, by showing the projected elements jumping in the screen.

- Bias: The Bias error is the wrong output signal of the gyroscope when it is not applied any force in the axis of the device, when the device is not experimenting any rotation, for example when is lying on a static surface.

- Drift: Is the change of the output signal over time. For instance, when using a gyroscope to determine orientation or an accelerometer to determine position. If there is even a slight bias, this small value adds up over time to a large error, producing a wrong displacement of the signal.

- Delay: Every event produced in the sensor is related to a specific timestamp. High demand in the resources (memory, processing power) can cause some delay effect in the sensed data, then generating obsolete data with incorrect timestamps.

## 2.5 Methods to solve errors

To solve the possible errors presented in the signal, it is necessary to implement some techniques that help to process the information provided by the sensors. Some of those techniques are already slightly implemented in the Android API and others are built by open developer communities.

Below will be presented some of the highlighted methods that are characterized by smoothing or averaging the values, or another more elaborated where the data provided by two or more sensors is fused.

### 2.5.1 Filters (High and low pass filter)

The filtering is used to reduce the noise in the signal by creating a bypass where the data that we want to process is selected or filtered. The filters could be developed by defining a bypass for high or low frequencies.

**Low pass filter** To implement is by filtering out high frequencies noise and letting pass low frequencies variable changes. One easy way is by using previous value against current one, and assign a specific weight (or smooth parameter) in the range of 0 <smooth˙parameter <= 1. The smoothed value will be balanced to the current value by assigning a number closest to one (1) to the `smooth_param`.

```
smooth_value  = (1-smooth_param) *last_value + smooth_param*current_value
```

Another method is by computing a running average [14], a more accurately method to calculate a dynamic mean by subsequent values.

Initialize:

```
initial_mean = first_value
```

For subsequent values, use the recurrence formula:

```
running_avg = running_avg + (current_value - running_avg)/num_values
```

To select a correct value for the smooth˙parameter, it must be adjusted to find the most suitable value for the application. If the value is closest to 0 the calculated value will not be influenced by the new incoming values, but if the values is closest to 1 the calculated value will be controlled by the newest values.

**High pass filter** The Android API provides some virtual sensors that are the combination between the values provided by real sensors (physical hardware installed in the smartphone) with some previous processing as filtering. As an example, is the sensor `TYPE_LINEAR_ACCELERATION`. It does not show the captured raw data from the sensor, it returns data that was processed by high frequency filter.

A high frequency filter works by attenuating the low frequency components (values). The value for the attenuation depends on the design that fit more with the application. A common implementation consists in defined first a low pass filter and later adjust the incoming data with the value given by the low pass filter.

```
smoothed_value=(1- smooth_p) *last_value + smooth_p*current_value
filtered_value=current_value-smoothed_value
```

### 2.5.2 Kalman Filters

Another additional filter is the Kalman Filters, based in prediction and posterior correction of white noise in signals over the time[15]. It is used in dynamic models where the data is measured sequentially like the scenario presented in a sensor reading, by doing estimation of the current values and posteriorly applying a weighted average, given more weight to the data with higher confidence.

Important fact to highlight is that Kalman Filters can be applied to any signal provided by any sensor, and is commonly used by merging different data (sensor fusion) under the principle that the state of a system provided by different measures is better estimated than just using one kind of measurement.

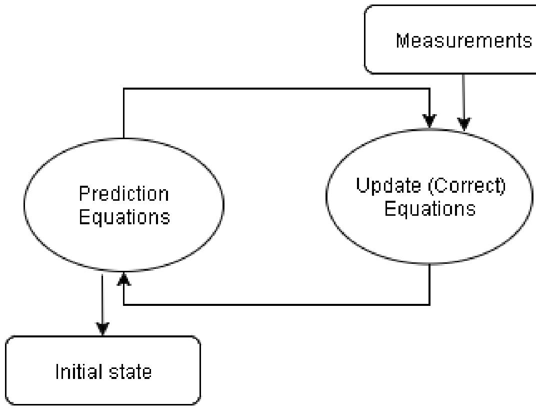


Figure 12: Kalman simple filter

#### 2.5.3 Sensor fusion

The filters are an important alternative that brings a correction/reduction of undesired values in the data provided by the sensors, but, it is not sufficient, because some sensors by itself has certain grade of uncertainly like the `ACCELEROMETER` and `MAGNETIC_FIELD`. Sensors include some natural noise, caused by external forces or the hardware itself, then causing unsuccessful results.

By implementing a fusion between the data provided by two or more sensors provides a better resolution of the actual problems presented in the signal processing.

The android API presents some synthetic sensors that are software-based:

- `TYPE_LINEAR_ACCELERATION`: Measures the acceleration force applied in the smartphone over the three physical axes x, y and z.
- `TYPE_GRAVITY`: Measures the gravitational force applied in the smartphone over the three physical axes x, y and z.
- `TYPE_ORIENTATION`: Measures the rotation in degrees around three axes x, y and z.

In the development of this project will be used a solution proposed by Shane Colton in the white paper "The Balance fIlter"[16](Figure 13) and implemented by Paul Lawitzki in the project "Sensor Fusion"[17] with the intention of used the values provided by the gyroscope fused with the data from the accelerometer and magnetic field. This with the aim to present a smooth view of the elements projected in the camera.

The complementary filter works by using the data of the three sensors, with the accelerometer and magnetic field is enough to calculate the orientation of the device, but is too inaccurate due to the noise in the signal.

To cancel the noise in the accelerometer-magnetic field orientation, it will be smoothed by passing it through a low pass filter to later be added to the orientation given by the gyroscope. The signal provided by the gyroscope (angular speed) is integrated to get the orientation angles and after is passed through the high pass filter to cancel the possible drift and noise in the signal.

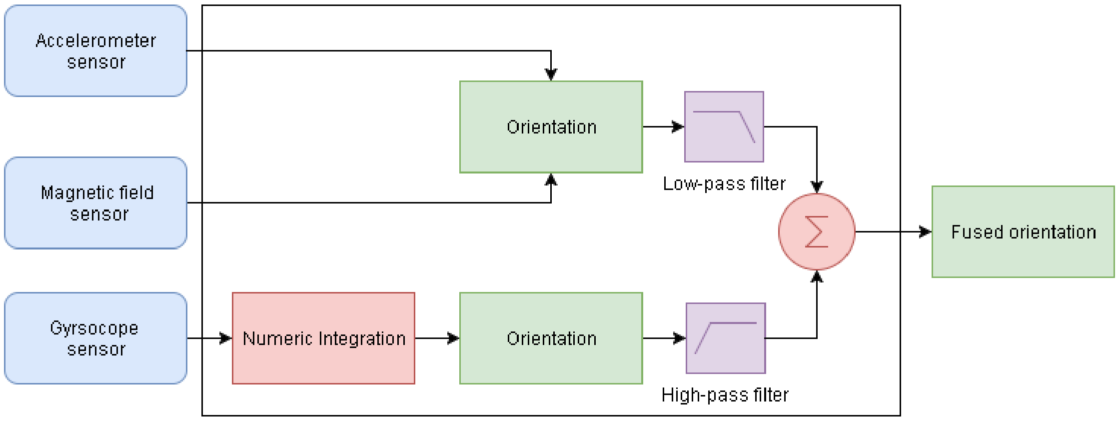


Figure 13: Complementary filter

In conclusion, the low pass filter is used to cancel short term fluctuations while the high pass filter does the opposite to preserve short term signals and filtering the constant high frequency values in the signal along the time.

Low pass filtering given by two options:

```
filtered_value = prev_value + ALPHA * (current_value - prev_value)
filtered_value = (1 - ALPHA) * prev_value + ALPHA* current_value
```

High pass filter works by replacing the high values in the orientation given by the accelerometer-magnetic field, that in fact is the final fused orientation:

```
fused_value = FILTER_P * gyroValue + (1- FILTER_P) * acc-mag_value
```

`FILTER_P` and `ALPHA` are parameters get it heuristically by testing and selecting the most suitable values for the application. The `FILTER_P` will have a value closest to 1 to give more relevance in the calculation to the gyroscope values. The ALPHA will have a value closest to 1 to give more relevance in the calculation to the current value.

## 2.6 AR Browser

An AR Browser is a search engine based on geolocation with augmented reality capabilities. Some of that capabilities are detect user location and track motion changes in the devices, to project in the screen an overlapping layer with markers referencing closest locations (local business, natural places, bus stations).

The aim of this project is to develop an application for Android smartphone, by using the capabilities offered by modern devices like GPS and motion sensors. The AR Browser are a mix of technologies with the objective of help users by improving the search places methods and giving a tool where they can interact with the found results.

### 2.6.1 Anatomy of a basic browser

To understand how works an AR Browser is necessary to know the basic parts that compound them as: Radar, Marker, Information box, Range (scroll zoom), Camera View and Map [2] (see Figure 14). They could have more, it will vary on each implementation.

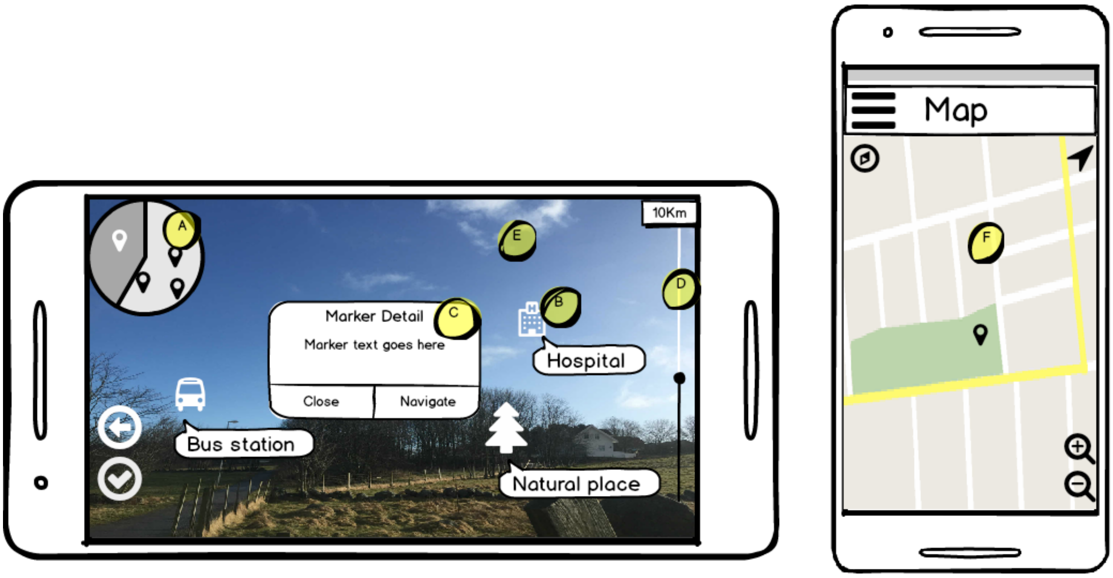


Figure 14: Basic anatomy browser

A. Radar: The radar is an element that shows points of interests (markers) included in the determined radius, by updating the sight with the near ones depending on the current angle view.

B. Marker: The points are presented as markers, which is an image associated with a name in a specific position in the screen, given by the latitude, longitude, and altitude of the real place. The information provided in each marker depends of the service that is being querying (for example, Google API returns a specific icon for each kind of place).

C. Information box: When a user select a marker in the screen by touching it, the application will show additional information concerns to the place, by presenting a box with a possible description and more data related.

D. Range: This component is associated to the ability to increase or decrease the area of search related to the current smartphone location. This component most of the times is a scroll bar, that acts like a zoom to change search area radius (distance in a circle from center to its perimeter).

E. Camera view: Is the layer where in it will be showed the image captured by the camera in addition to the elements that compound the AR capabilities. This component complement the real world with virtual objects to help user in follow a place.

F. Map: To complement the AR Browser is necessary to add a map to locate all the possible markers and present a more accuracy view of the obtained results.

# 3 Technology Review

Technologies that support the development of AR applications for smartphones was researched. There are a wide variety of development kits (SDK's), as well as open data services to set up an environment for develop projects in different platforms like IOS and Android.

Some of the most relevant technologies will be presented below, they are grouping in two classes: SDKs and APIs, the AR SDK provides relevant characteristics for geolocation searching, motion tracking, 2D and 3D models rendering, sensors and camera supporting, and native developing, specially focus in mobile platform.

In addition, an important variety of APIs public and private (some vendors provide open services with additional charges or special requirements like acknowledgment of their brand) to consume data from internet, through REST services with a previous authentication and authorization.

## 3.1 SDK

Development Kits oriented to build and test AR applications, some of that SDK are provided by means of libraries, an integrated development environment (IDE), or frameworks that offers specific services and utilities.

### 3.1.1 Wikitude

Wikitude[3] is a company that developed a SDK for augmented reality, providing features for 3D tracking, image recognition and geolocation. This SDK has an open community where it can be used a free trial license for instructive purposes only, and another commercial version with full support and additional features.

Earlier versions of this SDK provided a JAR (Java archive) with location based services to use georeferenced data. In the selection of technologies for this project, this SDK was discarded due to the licenses and additional charges to use, blinding the developer to know more about the specifics of the used services and the impossibility to modify parts of the code if need it.

Important features to highlight in this SDK are:

1. The Google Maps interoperability
2. They use an old fashion mark-up language: Keyhole Markup Language (KML) and Augmented Reality Markup Language (ARML)
3. Interoperability with web services to acquire data.

### 3.1.2 ARToolkit

ARToolkit[4] is an open source library released under GNU GPL v3[5]. It allows the development of augmented reality applications for a wide variety of mobile platforms, it includes specialized

[3]Wikitude: `https://www.wikitude.com/`
[4]AR Toolkit: `http://artoolkit.org`
[5]GNU GPL: `http://www.gnu.org/licenses/gpl.html`

framework for IOS, Android, Linux. Besides plugins for certain IDEs.

Some of the principal features of this project are related with graphics rendering, by using OpenGL specification to interact with a graphic process unit (GPU), and advanced topics for image recognition, video and sound tracking, hardware discovery and configuration. Additional to the use of native interfaces to develop or reuse C/C++ libraries

This framework is an important reference in the development of AR applications in the open source community, presenting different SDK for each platform to create virtual objects that overlaid on video in real time.

### 3.1.3 Android AR Framework

More than an SDK, this is a project[6] under the license GNU GPL v3, that proposes all the necessary elements to create an application with augmented reality features, by providing an implementation that supports connection with different data sources, an augmented reality view to display geolocated markers and interoperability with motion sensors, GPS and camera.

This is a framework to develop applications for Android Platform by using the sensors available in the smartphone like accelerometer and magnetometer, this project will be used as a baseline to understand and developed the proposed features in the current project.

The data source implementation invokes registered web services to acquire the data, also it is use an intermediate layer (or view) to register and process sensor events and draw markers in the camera view.

This project is very interesting, because it allows to understand and modify the current components (all classes are publicly available). It is possible to improve the current features by using additional algorithms or hardware that enhance the characteristics of an AR Browser. An enthusiastic (writer/developer) has followed this project to presented it like a valid reference[18].

Although, it is an old, not supported project, it will fit with the purpose of this project, being necessary add: updates in the authentication and web service invocation module, register another more reliable sensors like gyroscope, and use additional algorithms to complement the signal processing.

### 3.1.4 Tango SDK

Tango[7] is a computer vision - based platform, that provides the features of recognize current position and orientation of the devices. Tango relies in the use of the sensors and camera, by providing an accurate state of sensors (current data sensed and associated timestamps). Using tango allows to develop applications with motion tracking and depth sensing capabilities.

[6]ARF: `https://code.google.com/archive/p/android-augment-reality-framework/`

[7]Tango: `https://developers.google.com/tango/`

Some of the applications that is capable of being developed with Tango to explore physical spaces are: High precision navigation without need a GPS, creation of virtual 3D worlds, scanning of indoor spaces by recognizing elements around.

The requirements to developed with this SDK is to possess a device that support this framework, otherwise it will not be possible to run and test the application in a real device, just the development phase.

### 3.1.5 Selection of framework for AR

As presented previously, there are several SDKs for AR, however after the revision in terms of the given possibilities (Wikitude, TangoSDK, ARToolkit, Android AR Framework - ARF), it was concluded with the selection of a framework of public use, more specifically ARF, since it offers great characteristics for the creation and development of the browser. The reasons for selecting the SDK are described below.

Pros:

1. Offers services to manage the data provided by the sensors and camera integration.
2. Defines the basic elements (Section browser anatomy) of an AR Browser, to create easily an application.
3. Easy to extend model to collect data from new data sources.
4. Public use to modify and extend framework capabilities.
5. Last version adds fixes related to device orientation (portrait, landscape), marker collision (when is needed project markers in the same position).

Cons:

1. Authentication services outdated, it needs to add OAUTH integration.
2. Low accuracy and precision in the augmented view due to just uses sensors like magnetometer and magnetic field (compass) to project markers.
3. No project support, no recent updates.

The other SDKs reviewed have at least one of the following difficulties:

1. Wikitude is not totally free to use and modify it, or at least to verify and understand how it works.
2. No-commercial version of Wikitude does not have all the characteristics enabled, also presents a trial watermark in the camera view.
3. Tango SDK requires certain specific devices, if not, it is not possible to execute and test the developed applications.

4. ARToolkit does not have location integration, although it possesses an excellent integration with multiple devices and platforms, additional to the incorporation of OpenGL for graphic 2D and 3D rendering.

## 3.2 API Services

A key piece in the development of this project is to collect information from internet, to present relevant results to the searches made by the users. It becomes necessary to use Application Programming Interfaces (APIs) to manage the possible data sources, the API allows to access web services by means of invocate methods to read or write data through REST or another similar solution.

### 3.2.1 Google Places, Maps

Google provides access to several data sources through some APIs, with a previous authentication (development KEY) this APIs grant access to millions of resources like points of interest around the world, this dataset is in constantly updated by verified users.

The Google Places `API`[8] offers features to search, check details, and add places, with this functionality our project can acquire the necessary data to provide appropriate information to the AR Browser. To use this API is simple, just need to add a developer `KEY`, build a correct `URL` and resolve an `URL` through `REST` service.

An example of URL request looks like one below:

```
https://maps.googleapis.com/maps/api/place/search/json?
location=58.975440,5.718253&radius=1000&types=museum&sensor=true&key=KEY
```

- `Location`: Longitude and Latitude
- `Radius`: Distance for search area (in meters), allows a maximum distance of 50.000 meters.
- `Types`: Supported values for kind of values to filter the search in this example museum.
- `Key`: Provided developer KEY from Google.

An example of URL response looks like:


```
{
   "html_attributions" : [],
   "results" : [
      {
         "geometry" : {
            "location" : {
               "lat" : 58.9724098,
               "lng" : 5.7252117
            },
            "viewport" : {
```


[8]Google API: `https://developers.google.com/places/`

```
                "northeast" : {
                    "lat" : 58.9737587802915,
                    "lng" : 5.726560680291501
                },
                "southwest" : {
                    "lat" : 58.9710608197085,
                    "lng" : 5.723862719708498
                }
            }
        },
        "icon" : "https://maps.gstatic.com/mapfiles
/place_api/icons/museum-71.png",
        "id" : "461ee26ca900073ce1686a91899228f728432152",
        "name" : "Norwegian Canning Museum",
        "opening_hours" : {
            "open_now" : false,
            "weekday_text" : []
        }
        "place_id" : "ChIJl_cXA0w1OkYROVA2Thn-oLI",
        "rating" : 4.2,
        "reference" : "CmRSAAA...",
        "scope" : "GOOGLE",
        "types" : [ "museum", "point_of_interest", "establishment" ],
        "vicinity" : "Ovre Strandgate 88, Stavanger"
    },...
  ],
  "status" : "OK"
}
```


If the URL is correct the service will return a JSON object with results, the result contains information about the place as well as important details. The API has some restrictions about validation of a true key and number of requests per day.

A complement for places is the Google Maps API[9], providing enough tools to recreate a map view in the application. In the application is added this functionality to improve the searches and present in an interactive mode the points of interest to the user.

### 3.2.2 Open databases (Geonames)

Geonames[10] is a public database that stores geolocated data. It provides several amounts of information of different countries. This database is publicly available to download or to be accessed through web service. Geonames is mostly used by REST invocation.

To use this API is needed to create a user and resolve a well created URL, like the following example:

```
http://ws.geonames.org/findNearbyWikipediaJSON?
lat=58.975440&lng=5.718253&radius=1&maxRows=40&lang=en&username=USERNAME
```

[9]Google maps: https://developers.google.com/maps/
[10]Geonames: http://www.geonames.org/export/web-services.html

- `lat and lng`: Latitude and Longitude.
- `radius`: Distance for search area (in kilometers), maximum limit for free service is 20km.
- `lang`: Language code, default english (en).
- `feature`: The Wikipedia types available.
- `maxRows`: Amount of results to return, maximum number for free service is 500.
- `username`: To access the service is required a user (free or premium), in the development of the project was created a free user. The URL does not require password for free users.

Is a REST/JSON type web service, the response is like this:


```
{
        "geonames": [{
                "summary": "Stavanger Konserthus or Stavanger Conc (...)",
                "elevation": 20,
                "feature": "landmark",
                "lng": 5.719693055555556,
                "distance": "0.1846",
                "countryCode": "NO",
                "rank": 51,
                "lang": "en",
                "title": "Stavanger Konserthus",
                "lat": 58.976925,
                "wikipediaUrl": "en.wikipedia.org/wiki/Stavanger_Konserthus"
        }]
}
```


This API present an important open resource to access millions of POIs and the possibility to link them to the corresponding Wikipedia article, the Web Service also has some restrictions regarding of the number of requests per day.

### 3.2.3 Social Networks (Twitter, Instagram)

Nowadays many social networks offer API's to provide access to their content, the access grants permission to read and write information in the web, in addition, they provide well documented services to implement endpoints, to start requesting data from them.

Some examples are the Twitter[11] and Instagram[12] API, these API's need the creation of a previous account to get an access token, and request per permission. The use of some of that services requires an authentication based on OAuth protocol [13], once is get a successful registration and authorization, the application can request information with some limitations in the number of invocations per day.

[11] Twitter dev.: `https://dev.twitter.com/rest/public`
[12] Instagram API: `https://www.instagram.com/developer/endpoints/`
[13] OAUTH: `https://tools.ietf.org/html/draft-ietf-oauth-v2-12`

To use this APIs requires to be careful, because every query can return high amount of results and can have sensitive information, Social Networks APIs does not show the full geolocation of the information to keep safe people privacy.

Table 4: Smartphone sales by operating system

| Operating System | 2016 Units | 2016 Market Share (%) | 2015 Units | 2015 Market Share (%) |
|---|---|---|---|---|
| Android | 296,912.8 | 86.2 | 271,647.0 | 82.2 |
| iOS | 44,395.0 | 12.9 | 48,085.5 | 14.6 |
| Windows | 1,971.0 | 0.6 | 8,198.2 | 2.5 |
| Blackberry | 400.4 | 0.1 | 1,153.2 | 0.3 |
| Others | 680.6 | 0.2 | 1,229.0 | 0.4 |
| Total | 344,359.7 | 100.0 | 330,312.9 | 100.0 |

## 3.3 Platforms

Mainly AR Browser market is focused on IOS and Android devices, those platforms provides many options to design and implement easily and high quality applications. Development of this project is Android oriented, because the options for run and test this kind of applications does not require additional hardware in contrast with IOS that needs a host with their operative system to build the application.

Initially, the development of this project was intended to be in both platforms IOS and Android, by using a cross platform development software (Xamarin), but due to restrictions with the hardware (IOS), it was decided to be an Android application.

### 3.3.1 IOS

Is the second most popular operative system for mobile devices just after Android [19], provides excellent features because it is focused on a small number of different devices all made by the same company (Apple) with an exception of some public components. Source Gartner (August 2016), see table (thousands of units)4.

- Language: Objective Swift, C/C++, Java
- IDE: XCode

### 3.3.2 Android

The most used operative system for mobile devices, developed by Google, it is presented in a wide range of the devices, offering a flexible build system and a high interoperability[20].

This platform allows to develop Java projects mixed with native C/C++ libraries, those capabilities benefits the implementation of well-constructed applications by using additional libraries like OpenGL, ARToolkit and Unity, in addition with the use all the capabilities offered in the Android API.

- Language: Java, C/C++
- IDE: Android Studio

**Execution environment.** Because the application is focused on the development in mobile phone environment, the following facts rule Android projects: Limited resources, Mobile mashups, and Interchangeable applications[21].

Limited resources are related to the limitations that persist in the smartphones like battery capacity.

Mobile mashups (fusion) refer to the capacity of reuse data and current interface elements from web-based applications to provide a mobile access.

Interchangeable applications are related to the possibility of incorporating mechanisms called "Intents" that are independent of specific application implementations.

Those rules will guide the design and implementation of the future projects.

**Components of an application.** The Android architecture is defined by four basic components[21]. All Android applications will be built under this schema:

1. Activities: Represents the display screens. Most of the executable code will be write in the context of an Activity. Activity allows to interact with users and request data or other resources through another activities or services. The activities are ruled for a Lifecycle with a Start and End.
2. Services: Request for data or resources, this are executed in the background (Synchronous or asynchronous) and commonly they do not expose a user interface.
3. Broadcast and intent receivers: In charge of respond requests from another activities or applications.
4. Content providers: Share data with specific activities and services.

## 3.4 Related Work

This report documents ongoing projects by reviewing used technology and defined components, to provide the basis to design and develop an improved solution that support the objectives of the present work.

The search and study of AR Browsers, provides the possibility to understand their implementations and evaluate defined components.

### 3.4.1 Monocle Yelp

Is an additional feature in Yelp application, providing an Augmented Reality browser, where the user can search by local business and the results will be display in the camera view (Figure 15). The application was tested in IOS and Android, presenting regular performance in Android device but good performance in IOS devices.

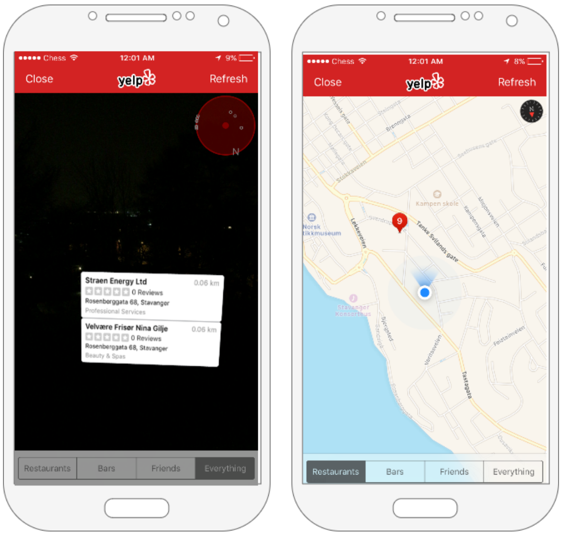


Figure 15: Interface Monocle Yelp - IOS

The problems presented in the Android version does not allow check the markers because they are jittering and bouncing all over the place. The markers are out of the radar, despite of the smartphone is facing the exact orientation of the places. In addition to some limitations on search filtering and setup of augmented reality components as sensors and map view, see Figure 16.

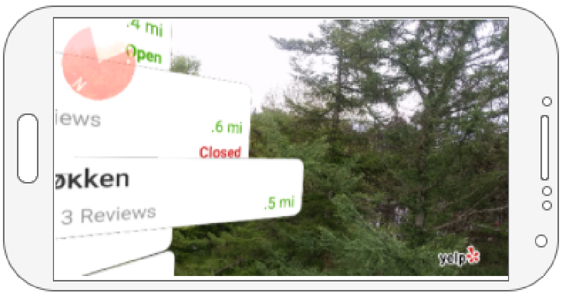


Figure 16: Interface Monocle Yelp - Android

### 3.4.2 View Ranger App

View Ranger is an application to explore outdoors offering different features for map integration and allowing to use the application without data credit or phone signal (Figure 17).

Once again, the application was tested in IOS and Android, presenting a high performance in IOS and some problems in Android. The problems presented was a slightly drift in the position, making the markers look delayed and problems when the device change position from landscape to portrait.


Figure 17: Interface View Ranger

### 3.4.3 Wikitude Browser

Wikitude is a well-known Augmented Reality browser, with several capabilities to search locations and present the results in the camera view (Figure 18). Wikitude allows to filter the searching and select the results by touching the screen to get more related information.

This browser works good in IOS and an acceptable performance in Android. When is executing a search in Android presents some wait time when is downloading information to the phone, and delay in the rendering of the new position when the device orientation changes.

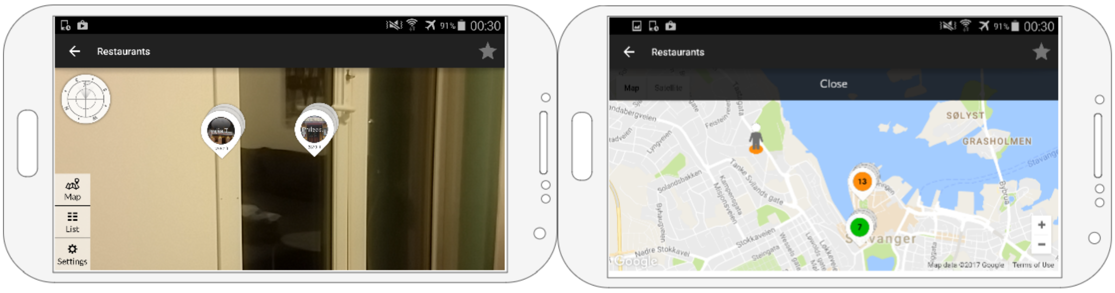


Figure 18: Interface Wikitude

# 4 Analysis

In this phase, the functional and non-functional requirements are determined, as well as the components and features.

The development of this phase established the flow process and the relation between components in the development of an AR Browser.

## 4.1 Process Model

The model (see Figure 19) describes the flow of events in the application, this flow is based on the interest of the user to find specific places, setup searching preferences, and the projection of

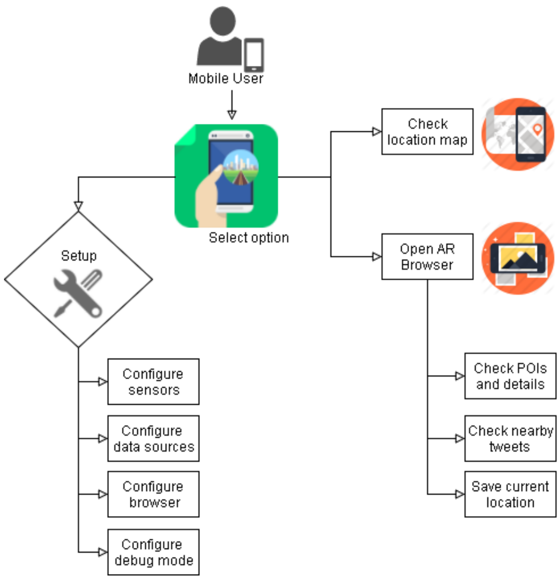


Figure 19: Interaction Process in the APP

location markers in the map or camera view, by using the sensors provided by the smartphone and the camera to show POIs in the augmented view.

## 4.2 Domain Model

In the domain model (see Figure 20) is illustrated all the entities related to the project and it is described the permissions and functionalities that an user is allowed to does.

### Glossary of terms

- User: Is the person that interacts with the smartphone, in charge of executes all the options presented in the application.
- AR Browser: Software application that helps to search information by filtering information and projecting results in an augmented view (video camera capture).
- Map: Graphic representation of a region (neighbourhood, city, country) abstracting all the elements that provides orientation by finding coordinates. In the project is used the service provided by Google maps.
- POI: Point of Interest that illustrate a real place or location.
- Location: Current geo position of the user or markers, containing Latitude and Longitude provided by the GPS or Network provider.
- Tweet: Short message shared by people in the social network tweeter. Provides certain kind of information that can be useful for decision-making (local news, comments about places, recommendations).

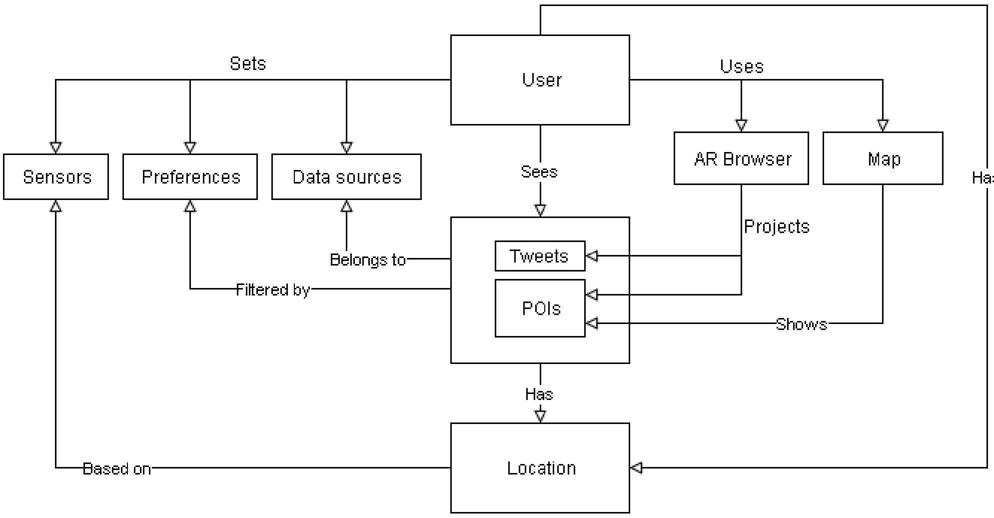


Figure 20: Project domain entities

- Data source: Source of information that provided data by accessing certain web services. The data collected presents predefined format and previous filtering (based on location or preferences).
- Preference: A preference can be a selection of options defined by the user, in terms of filtering, selected data sources or sensors. A group of preferences is a user configuration.
- Sensor: Hardware or software artefact that sense the environment and provided real time data. The used sensors depend on the specifications of each smartphone.

The features defined in this project are mentioned below, each one will be subdivided in smaller tasks:

## 4.3 System Features

Below is listed the features identified in the analysis, to achieve the objectives proposed in the project.

- Select different data sources to acquire data from internet (Google Places, Geonames, Twitter and places saved by the user).
- Validate changes in the location and orientation of the device.
- Use camera to capture real-world and project the markers in it.
- Permit to filter results and save information.
- Allow user to interact with the results by getting detailed information and integrating map view.

## 4.4 Technical details of interest

Once the functional requirements are defined, the methods and components to create an augmented reality application will be described. The selection of programming language and tools for the development of this project was made considering the following facts:

- Android is the most used mobile operative system in smartphones (see Gartner magic quadrant 2016[19]).
- The current experience of software development in Java platform.
- Android API allows to build rich featured applications in Java.
- A very important amount of free use libraries to developed like ARToolkit for android and Google API (places, maps), in addition to the wide developer community for open source projects.
- Android Studio is an excellent IDE with several features to create and build different kind of applications.
- The easiness to code and test the application without the necessity of have physical devices (virtual devices that emulate the proper functioning), in contrast with IOS development that force you to have a host with a specific operative system.

The present project is settled in the realm of augmented reality and the development of a AR Browser for Android smartphones, through the implementation of an application that converges technologies such as web services, Android API, OAUTH authentication, RESTFul services, and processing of real-time data.

## 4.5 Methodology

In addition to the selection of components, it is necessary to define a software development methodology, to define a clear and agile process to reach the objectives of the project.

The main objective of this project is not to follow a methodology, but it is necessary to carry out a clear and incremental development process.

The basic stages of software development life cycle that followed this project will be framed under an iterative process, where, it will be built incrementally by creating value in each iteration and adding new features in each cycle.

Five stages compose each iteration:

1. Prototyping the AR application feature[22], by defining the components related to build an AR Browser.
2. Definition of requirements (analysis), the output is a feature with clear defined specifications.
3. Design architecture based on feature, output is a proposed model that present the guideline to accomplish the feature, it can be a new component, an algorithm, or integration.

4. Implementation of stories, is the build stage where the feature will be developed, by using the selected tools and following the defined model.

5. Testing and evolution, in this stage will be verified the correct functioning of the implementation by validating functional and non-functional requirements. If test is passed, then it will start a new iteration, if not the feature will be reconsidered by checking possible errors or making changes in the specification.

### 4.5.1 Definition of Algorithms for sensor fusion and data filtering

For the definition of the algorithm for signal filtering and fused sensors, it was validated the selected framework for AR (ARF). Initially, ARF was tested with the given sensors Accelerometer and Compass, and after was incorporated the gyroscope.

The signal was collected and checked to validate the possible noise. In the graphs Figure 21 and Figure 22 can be appreciated the jittering in the signal captured from the accelerometer.

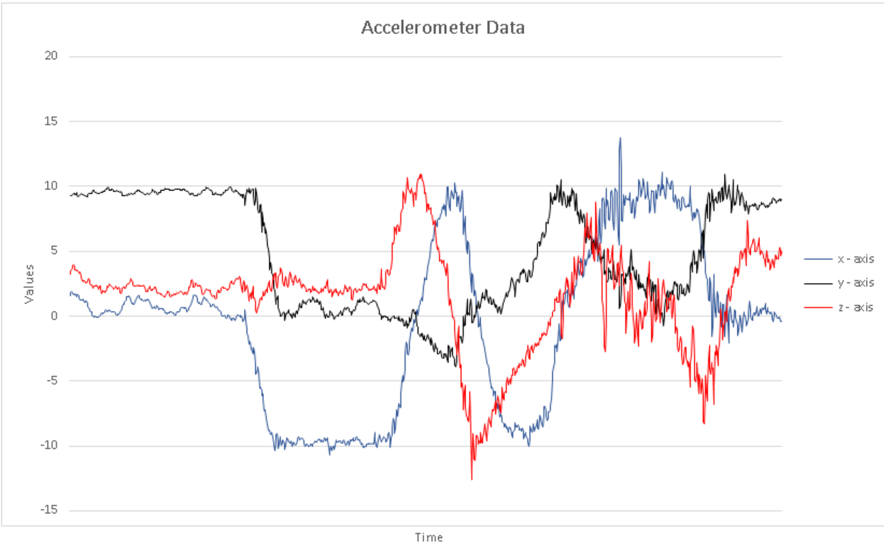


Figure 21: Graphic showing jittering in sensed acelerometer

In the data given by the gyroscope it can be illustrated a more regular signal without many abrupt changes in the data. After analysed the data, it was cleared the need to include the gyroscope to improve the accuracy in the results.

ARF already implements some strategies for low-pass filter, by reducing the amplitude of frequencies higher than the cut-off frequency signal creating a bypass that attenuates the signal. In the used strategy was defined an alpha threshold that was calculated by computing the difference between previous and current values of the signal.

The strategy selected for fused sensors was based on the white paper created by Shane Colton

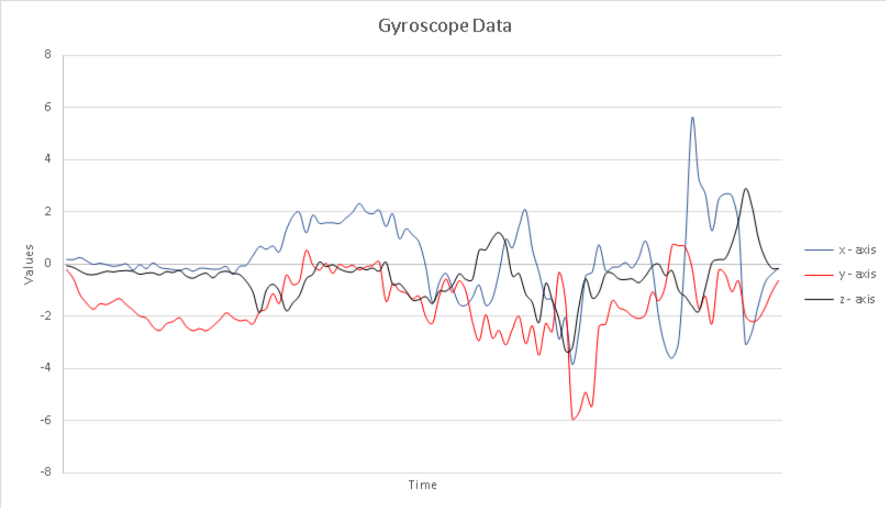


Figure 22: Gyroscope data with less jittering

about the "Complementary filter" (described in the background section 2.5.3 Sensor fusion), where the signals given by both sensors (gyroscope and accelerometer) is fused to provide a more accurate estimation of the angles. The complementary filter is easier to understand that the Kalman filters, this solution is enough for the scope of this project.

## 4.6 Functional and no functional requirements

Determine functional and non-functional needs of the system to be developed is an important task, as well as the capture of user actions and cleansing of requirements. This step is of vital importance since the requirements allow establishing the functionality and the necessary inputs for the project.

Features. This section lists and describes the operations that potential users must perform on the system, and the responses that the system must executes. The above-mentioned objective of this theme is detailed in the system design.

- Data source integration (Selection, filtering).
- Save current places (local markers).
- Twitter integration.
- Gyroscope integration.
- Google maps positioning.
- Configuration mode and calibration.

**Non-Functional requirements.** Describes how the application will works, it will be defined as an application constraints that define the quality attributes in the project. The required non-functional needs are defined in the system architecture, due to the those are architectural constraints.

- Authorization and authentication for web services.
- Acceptable levels of accuracy and precision in the data collected from sensors.
- Scalability to use new data sources.
- Consistency in the saved information: user preferences and reached places.
- High usability by implementing an easy and intuitive navigation in the application.

# 5 Design

This phase of the project defines the components that are part of the logical model of the system, which will be implemented and will support the functionality of the application.

## 5.1 Architecture Diagram

In the Figure 23 is illustrated the architectural components of the application, as well as the communication between them. The application accesses the services exposed in the APIs of Google, Twitter and Geonames through a client that executes HTTP requests. For the services of Google and Twitter is required a model of authentication and authorization, by defining a private key through the protocol OAUTH2.

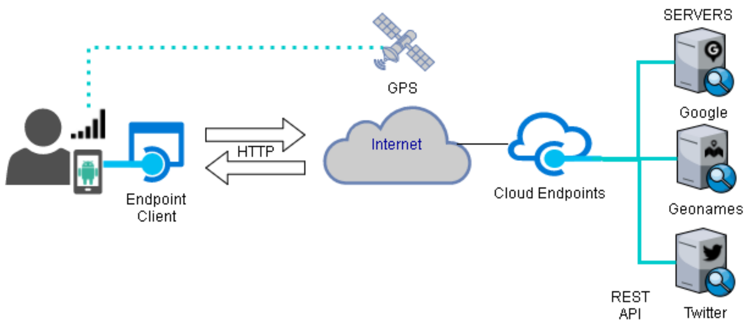


Figure 23: Architecture model

In addition, the application accesses the sensors enabled in the device as well as the GPS to obtain guidance and location respectively.
The application is created through the definition of activities that supports the system requirements, those activities have defined specific actions and allows navigability into the application.

Additionally, the preferences of the user and the sites that the user has decided to save are stored in the smartphone.

## 5.2 Interface Mockups

Based on the defined features, it is created six mockup forms, to illustrate the application look and feel, and the navigation between options. The AR Browser is composed by an initial screen, AR screen (camera view), settings screen and lateral menu to configure different options in the application. The graphic diagram is presented below.

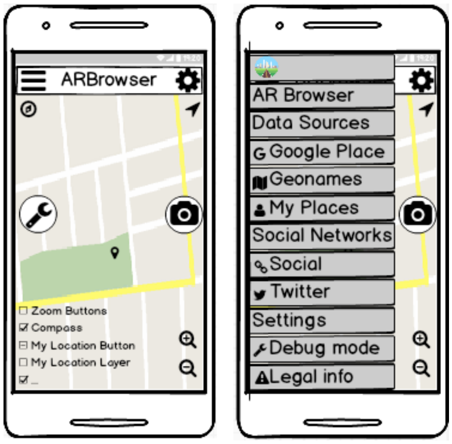


Figure 24: Initial screen and lateral menu

Once the user enters the application, the initial screen is displayed (Figure 24), it is displayed in the center the map marking the current position of the device, in the map two buttons, one is for enabled/disabled options related to the current position, the second button is to access to the AR view. Additionally, there is a top bar in which the user can setup the application and a side menu to make a more advanced configuration of the system.

The side menu is displayed by swiping the finger on the left side of the screen, once it is displayed this will show the options of configuring data sources, social networks, and developer mode, which has options for debugging and legal information of the used Projects / libraries.

The possible options in the configuration screens (Figure 25), allows to setup an advance configuration, by enabling data sources and setting the information to connect to the web services.

Last mockups (Figure 26) presents the developer options: sensor calibration and the augmented view. The augmented view illustrates the basic elements to create an augmented browser, by projecting three example markers, the radar, zoom tool and filtering options.

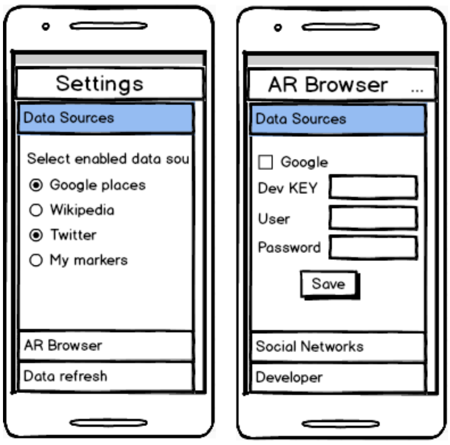


Figure 25: Setting up app with Lateral menu and Toolbar options

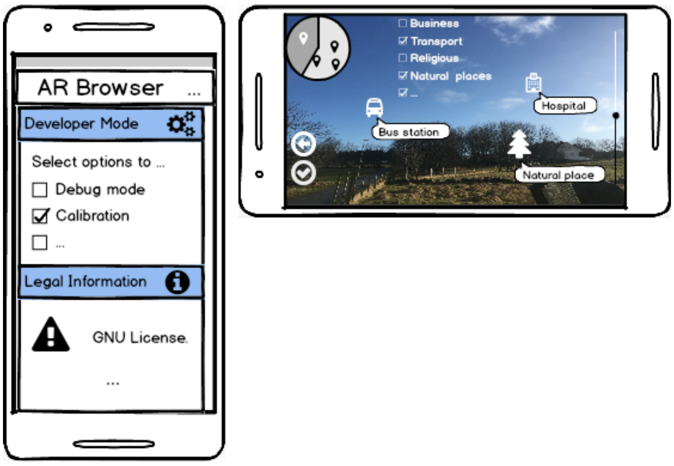


Figure 26: Developer options and augmented view

## 5.3 Navigation model

This model (Figure 27) illustrates all the activities and the navigability between them. One or more activities compose Android applications. The activities defined in this project are grouped in: User Interface (UI) activities and AR activities.

UI activities includes the splash screen, main interface where is added the map, a legal information screen, a toolbar, and a lateral bar for user settings.

- StartActivity: Once the application is running, this activity shows the initial splash screen.
- MainActivity: Main screen where is included the Google Map to insert markers, settings bars (Toolbar and lateral menu).
- TabbedActivity: Represents the lateral menu where is listed all the options to configure

datasources, calibration options, and legal information.

- InfoActivity: Important activity that show the licenses of used frameworks.
- SettingsActivity: Toolbar menu that includes the options to configure the AR browser as enabling datasources and selection of filters.

AR groups the AR Browser activities and their hierarchy, it initializes the accesses to all the available AR features, by querying data sources and projecting markers on the map and in the camera view.

- ARBrowser: This class integrates all the available data sources and user preferences. This class extends from AugmentedReality.
- AugmentedReality: This class included all the elements in the augmented view as camera screen and dashboard elements (Radar, zoom bar, filters). This class extends from SensorsActivity to capture all the changes in the sensors.
- SensorsActivity: Class in charge of detect and process all the data from the sensors and location changes.
- AugmentedView: Designed to draw all the markers in the screen and radar. The markers are composed by an icon indicator and a label to show the name.

This model represents the navigation between activities grouped in two different sets: the UI illustrated all the activities in the application and AR exemplify the additional components to generate an augmented view as sensor listener integration and camera view.

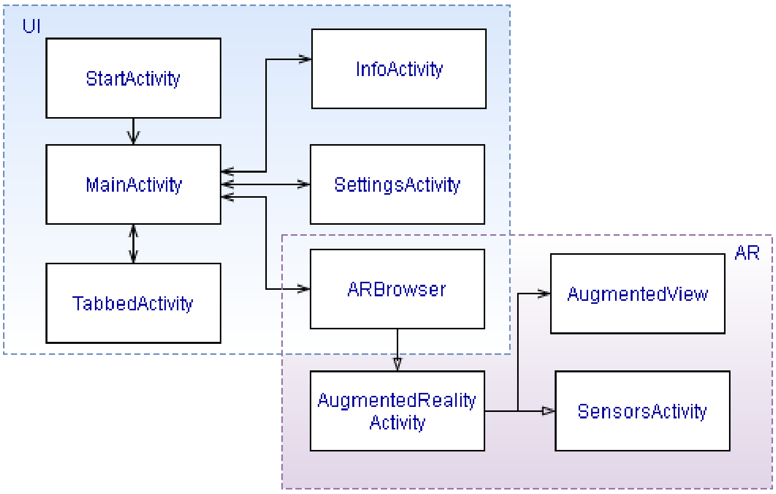


Figure 27: Design model

# 6 Implementation

The aim of this project is the design and implementation of an AR Browser to converge different technologies, the selection of technologies was made based on the current experience and choosing the most relevant software specifications in the market of IT.

In this phase, the relationships between the classes that make up the application are described, as well as the most interesting points in the implementation, regarding the handling of the sensors, GPS, and projection of elements in the increased view. Additional is presented the anatomy of the application and developed interfaces.

## 6.1 Class diagrams

The classes are grouped and presented based on the functionality, most of the classes belong to the used framework ARF. Some changes were made it to improve the application (DataSources and AR classes), in addition, some classes were included for the data fusion management (SensorFusion and activities classes).

The original packages of the framework were modified to present them in a more orderly way, in convenience with the development of this project. The authors of these classes were preserved in classes to keep credit to their developers. The classes, attributes and relationships that make up the system are described below.

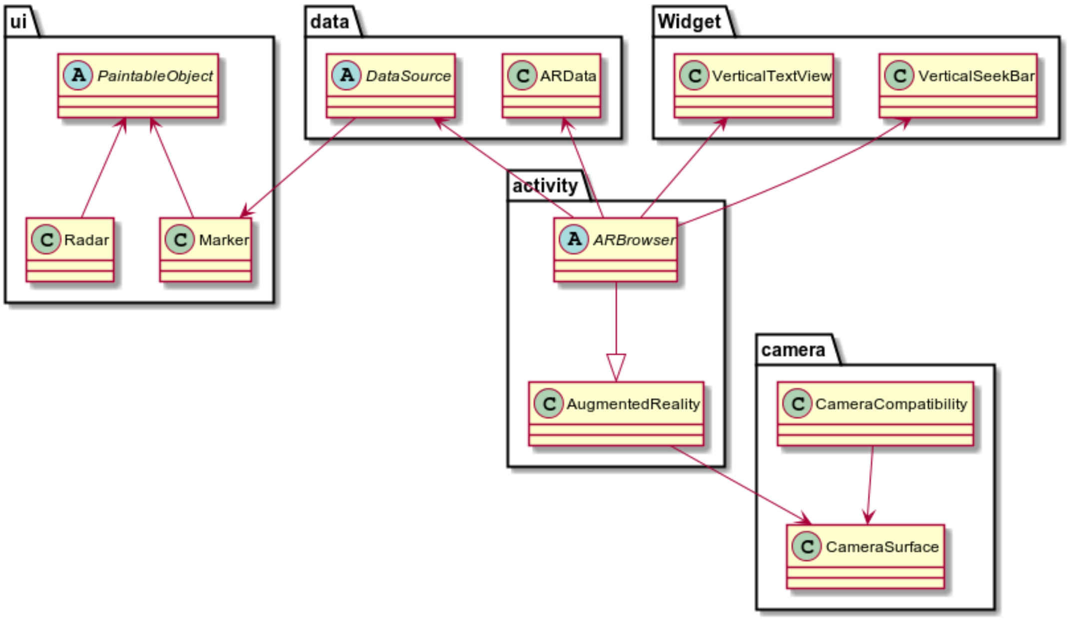


Figure 28: Main classes and relations in the application

## 6.2 Main classes diagram

The main classes in which most the actions in the application are executed is the diagram of the Figure 28, in this model the relation between 5 packages is illustrated:

- Package `activity` presents the views of the application.
- Package `data` groups the definition of data sources.
- Package `camera` contains the classes that abstract camera device.
- Package `ui` contains all the elements that abstract the markers (points of interest) and associated details, to draw them in the camera as images and texts.
- Package `widget` contains additional elements to complement the AR view, such as zooming tool and vertical text view.

The ARBrowser class is oriented to handle all the enabled data sources, by means of the validation of the current configuration. In addition, this class extends from the AugmentedReality class, which provides the necessary methods to implement and define the sensor listeners, in order to capture the changes in the orientation and location.

Another important classes contains the views of the application, those views are represented as Activities, in the Figure 29 is presented the activities class diagram.

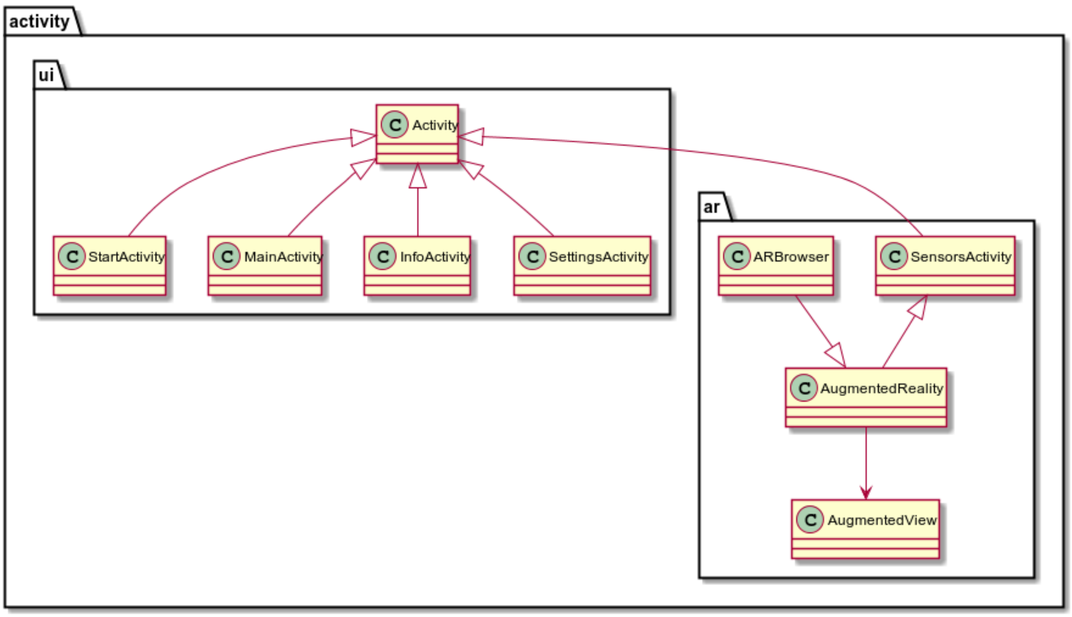


Figure 29: Activity classes

## 6.3 Collecting data from sensors

For the implementation was used the Java Android API 7.1 [14] that provides many useful features to collect data from sensors, camera integration and web services invocation.

The steps to collect data from sensors are:

- Define in the `AndroidManifest.xml` the permissions to use the sensors and additional resources (camera, location).

```
<uses-permission android:name="android.permission..." />
```

- Define a class that implement `SensorEventListener` and `LocationListener`, to be aware of changes in the device (movement, rotation) and location, in the ARF frameworks is defined the class `SensorsActivity`.

```
class SensorsActivity implements SensorEventListener,LocationListener{
...
}
```

- Check if permission is granted to access the sensors or other resources[23].

  If the AR Browser requires a permission over a specific resource, it must be verified whether it has that permission every time is executed an operation that involves that resource.

  It is important to revoke the permissions if the user request it, so even if the application have used some resources, it currently can't assume it still possess that permission.

  If is not defined the required permissions in the Manifest file, the application need to invoke the requestPermissions() method to request for it.

```
if (checkAndRequestPermissions())
{
  Snackbar.Make(layout, "Location access is required to show places.",
        Snackbar.LengthIndefinite)
  .SetAction("OK", v => requestPermissions(PermissionsLocation,
        RequestLocationId))
  .Show();
  return;
}

private boolean checkAndRequestPermissions() {
  string permission = Manifest.Permission.ACCESS_COARSE_LOCATION;
  int location = PermissionChecker.checkSelfPermission(this, permission);
  List<String> listPermissionsNeeded = new ArrayList<>();
  if (location != PackageManager.PERMISSION_GRANTED) {
    listPermissionsNeeded.add(permission);
  }
  ...// All necessary permissions
  if (!listPermissionsNeeded.isEmpty())
```

[14]Overview: `https://developer.android.com/about/versions/nougat/android-7.1.html`

```
    {
      android.support.v13.app.ActivityCompat.requestPermissions(
            this, listPermissionsNeeded.toArray(
          new String[listPermissionsNeeded.size()]),
          REQUEST_ID_MULTIPLE_PERMISSIONS);
    }
    return true;
}
```

- Register the sensor listeners to collect events in the specific sensor and specifying delay type, method `registerSensorManagerListeners`. It must be unregistered when is not more longer need it, to for save resources and avoid overheating of the device.

```
public void registerSensorManagerListeners() {
  mgr.registerListener(this, ACCELEROMETER, SENSOR_DELAY_FASTEST);
  mgr.registerListener(this, GYROSCOPE, SENSOR_DELAY_FASTEST);
  mgr.registerListener(this, TYPE_MAGNETIC_FIELD, SENSOR_DELAY_FASTEST);
}
```

- Implement the `OnChange` methods, to code the necessary validations to process the data: Collecting, filtering, and posterior transformation.

```
@Override
public void onLocationChanged(Location location) {
...
}
@Override
public void onSensorChanged(SensorEvent evt) {
...
}
```

Also, the method `onAccuracyChanged` is implemented to validate the accuracy of the sensors, and force user to calibrate them.

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
  if (sensor == null) throw new NullPointerException();
  if (sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD
                && accuracy < ARData.ACCURACY_LEVEL) {
    Log.e(TAG, "Compass data unreliable");
        String accuracyTxt = "";
        switch (accuracy){
                case SensorManager.SENSOR_STATUS_ACCURACY_HIGH:
                        accuracyTxt ="HIGH";
                        break;
                case SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM:
                        accuracyTxt ="MEDIUM";
                        break;
                case SensorManager.SENSOR_STATUS_ACCURACY_LOW:
                        accuracyTxt ="LOW";
                        break;
```

```
                    case SensorManager.SENSOR_STATUS_UNRELIABLE:
                            accuracyTxt ="UNRELIABLE";
                            break;
            }
            openDialogCalibrate(accuracyTxt);
    }
}
```

## 6.4 Orientation calculation

One of the challenges in the development of this project was to improve the precision and accuracy of the processing of the signals provided by the sensors. The sensors produce high amounts of data in a very short time, those data include significant quantities of noise that needs to be reduced to present acceptable results, regarding to the projection of the markers in the augmented view.

According with the methods illustrated in the section 2.2 Determine Location and 2.5 Methods to solve errors. Described approaches was used to filter the signals and get more consistent sensor readings (smoothed data).

In addition, the complementary filter was used to fuse the sensor data provided by the accelerometer and compass, and complement it with the gyroscope data. Those alternatives were included to enhance the data by reducing the drastic changes in the signals.

### 6.4.1 Filters

The approach was based on `low-pass` filters, `high-pass` filters, and data sensor fusion, by using the data provided by the 3 sensors: `accelerometer`, `magnetic field` (compass), and `gyroscope`, the last one included as an alternative to improve the performance of the application.

The ARF framework includes a class called `LowPassFilter` (see Figure 30), that is used to smoothed the data collected from the `accelerometer` and `magnetic field` sensors. The `LowPassFilter` works by reducing the amplitude of signals that exceed the threshold (cut-off frequency).

Once is invoked the `LowPassFilter`, it will filter the given signal against the previous values and a smoothing constant. Low-pass filter `ALPHA` where $0 \leq \alpha \leq 1$, a smaller value means extra smoothing.

```
public void onSensorChanged(SensorEvent evt) {
        switch (evt.sensor.getType()) {
                case Sensor.TYPE_ACCELEROMETER:
                case Sensor.TYPE_MAGNETIC_FIELD:
                smooth = LowPassFilter.filter(lowest_alpha, highest_alpha,
                                                        evt.values, arr);
   ...
}

public static float[] filter(float low, float high, float[] current,
                                 float[] previous) {
   float alpha = computeAlpha(low, high, current, previous);
   for (int i = 0; i < current.length; i++) {
```

```
        previous[i] = previous[i] + alpha * (current[i] - previous[i]);
    }
    return previous;
}
```

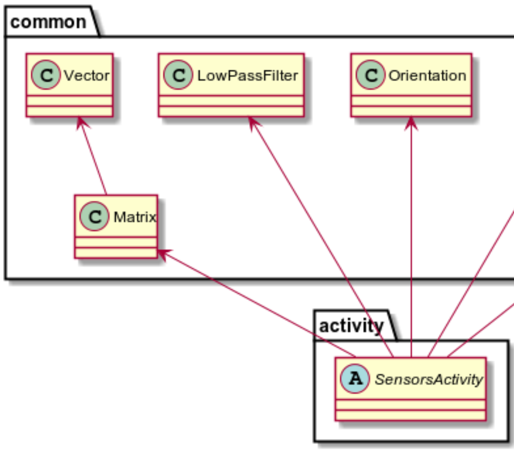


Figure 30: Sensor classes and relations

### 6.4.2 Sensor Fusion

After get data from the sensors, it will be used the class `SensorFusion` by invoking the methods `calculateAccMagOrientation` and `gyroFunction`, in charge of calculate the orientation based on `accelerometer` and `magnetic field` data, and calculated the fused orientation respectively.

```
sensorFusion.calculateAccMagOrientation();
sensorFusion.gyroFunction(gyro, evt.timestamp);
//thread to calculate fused orientation
sensorFusion.run();
```

The `gyroFunction` integrate the data from the `gyroscope` over the time, by calculating the `rotation vector` and converting the `rotation vector` in the `rotation matrix`. This method is based in the reference described in the `sensorEvent` page[15].

`SensorFusion` implements a `high-pass` filtering with the `gyroscope` data by replacing the filtered `high-frequency` values from the result of the method `calculateAccMagOrientation` with the equivalent `gyroscope` orientation values.

```
float oneMinusCoeff = 1.0f - FILTER_COEFFICIENT;
// azimuth
if (gyroOrientation[0] < -0.5 * Math.PI && accMagOrientation[0] > 0.0) {
 fusedOrientation[0] =(FILTER_COEFFICIENT*(gyroOrientation[0]+2.0*Math.PI)
```

[15]SensorEvent Reference: `https://developer.android.com/reference/android/hardware/SensorEvent.html`

```
                                   + oneMinusCoeff * accMagOrientation[0]);
}
// pitch
if (gyroOrientation[1] < -0.5 * Math.PI && accMagOrientation[1] > 0.0) {
 fusedOrientation[1] =(FILTER_COEFFICIENT*(gyroOrientation[1]+2.0*Math.PI)
                                   + oneMinusCoeff * accMagOrientation[1]);
}
// roll
if (gyroOrientation[2] < -0.5 * Math.PI && accMagOrientation[2] > 0.0) {
 fusedOrientation[2] =(FILTER_COEFFICIENT*(gyroOrientation[2]+2.0*Math.PI)
                                   + oneMinusCoeff * accMagOrientation[2]);
```

## 6.5 Location calculation

The location is calculated in the class `SensorActivity` by implementing the `LocationListener`, this implementation is quite simple, after providing the permissions is asked to the `LocationManager` for the last known location, by validating the available provider (`GPS` or `NETWORK`).

```
try {
    Location gps = locationMgr.getLastKnownLocation(GPS_PROVIDER);
    Location network = locationMgr.getLastKnownLocation(NETWORK_PROVIDER);
    if (gps != null) onLocationChanged(gps);
    else if (network != null) onLocationChanged(network);
    else onLocationChanged(ARData.hardFix);
} catch (Exception ex2) {
    onLocationChanged(ARData.hardFix);
}
```

An improvement in this implementation was done by using a method to validate the new location against a previous one, method `isBetterLocation` from location strategies page[16]. `IsBetterLocation` method validate the new value by checking if is newer than the previous one, the accuracy of the new location, and checking the confidence in the provider.

## 6.6 Camera view and markers projection

The Camera is one of the most important components in the application, because, it is the bridge between the real world and the virtual objects that create an augmented reality scenario.

Using the camera allows to capture real-time data in form of video, the configuration is simple by assigning permissions to the application and creating a surface that will contain the virtual elements. The class `AugmentedReality` creates an `CameraSurface` object that uses and initializes the camera device.

Once, the `CameraSurface` is initialized, is set as a view in the `AugmentedReality` class to add all the elements like markers, texts, and the whole dashboard with the options.

```
camScreen = new CameraSurface(this);
```

[16] Location strategy reference: https://developer.android.com/guide/topics/location/strategies.html

```
start(camScreen);
```

Projection of markers are done after the data is collected and filtered from the services. The markers are created based on the results of the invocation of a `HTTP` request. The data obtained is filtered by latitude, longitude, and altitude, additional to the filters defined for each web service as: radius and places type.

```
private void updateData(final double lat, final double lon, final double alt){
    try {
        exeService.execute(new Runnable() {
            @Override
            public void run() {
                for (NetworkDataSource source : sources.values())
                    download(source, lat, lon, alt);
            }
        });
    } catch (RejectedExecutionException rej) {
        Log.w(TAG, "Not running new download Runnable, queue is full.");
    } catch (Exception e) {
        Log.e(TAG, "Exception running download Runnable.", e);
    }
}
```

The previous method is invocated in three different moments:

1. When the application start, then it is instantiated the class `ARBrowser`.
2. When the location changes, it is verified if is a significant change in the location by validating the new location with the previous one.
3. When the zoom control change, then the `radius` of search increase or decrease its range.

The `download` function invokes the respectively service `(Google, Geonames, Local, or Twitter data source)` and execute the `override` method `createRequestURL`, after each service parse the resulted `InputStream` from the `HTTP` response to a `JSON` object, to be serialized to an array of `Markers`. Each item in the list of `Markers` is converted to a `Canvas` object to be drawn in the `CameraSurface` like an icon or a text. That list of markers also update the points in the radar.

## 6.7 Collecting information from data sources

Code below obtain the user preference to validate if Google data source is enabled (`isGoogleChk`), if the validation is affirmative, then, data source will be added to the array of sources to get the information to the browser. That validation will be executed for each one of the available data sources. This portion of code will be evaluated every time that is initialize the AR Browser.

```
Map<String, NetworkDataSource> sources = new ...
boolean isGoogleChk = sharedPref.getBoolean(KEY_PREF_DS_GOOGLE, false);

if(isGoogleChk) {
  NetworkDataSource googlePlaces = new GooglePlacesDataSource(...);
```

```
  sources.put("googlePlaces", googlePlaces);
}
```

Once the data source is added to the AR Browser, each implementation will load the settings parameters (necessary KEYS, login information and additional parameters to filter the search) and override the methods to `createRequestURL` and `parse`, to execute the search and parse the results respectively.

```
@Override
public String createRequestURL(double lat, double lon, double alt,
                                        float radius, String locale) {
  try {
     return URL + "location="+lat+","+lon+"&radius="+radius
              +"&types="+ARData.getFilters()+"&sensor=true&key="+key;
  } catch (Exception e) {
     e.printStackTrace();
     return null;
  }
}

@Override
public List<Marker> parse(String URL) {
   InputStream stream = getHttpGETInputStream(URL);
   String marks = getHttpInputString(stream);
   JSONObject json = new JSONObject(marks);
   return parse(json);
}
```

## 6.8 Authentication for data collection

The authentication and authorization of the services is provided by the definition of a developer key in the case of `Google API` and a user in the case of `Geonames API`.

In the case of `Twitter API`, it uses `OAUTH2` to provide authorized access to its API. To use this service was necessary to import the library `Twitter4j`, that provide services for creating and executing queries to the Twitter servers.

Send secure request to the Twitter API requires certain `KEYS` to granted the access.

```
query = new Query().geoCode(new GeoLocation(lat, lon),
              Math.max(radius, 1.0), "km");
query.setCount(10);
query.setSince(formattedDate);

ConfigurationBuilder cb = new ConfigurationBuilder();
cb.setDebugEnabled(true)
        .setOAuthConsumerKey(TWITTER_CONSUMER_KEY)
        .setOAuthConsumerSecret(TWITTER_SECRET_KEY)
        .setOAuthAccessToken(TWITTER_ACCESS_TOKEN)
        .setOAuthAccessTokenSecret(TWITTER_ACCESS_TOKEN_SECRET);
```

```
TwitterFactory tf = new TwitterFactory(cb.build());
Twitter twitter = tf.getInstance();
try {
    QueryResult result;
    do {
        result = twitter.search(query);
        List<Status> tweets = result.getTweets();
...
```

Once the access is granted, it can be executed `HTTP` requests to collect data from the servers. Each implementation (See Figure 31) of the class `NetworkDataSource` collects and transforms the data in `Markers`.

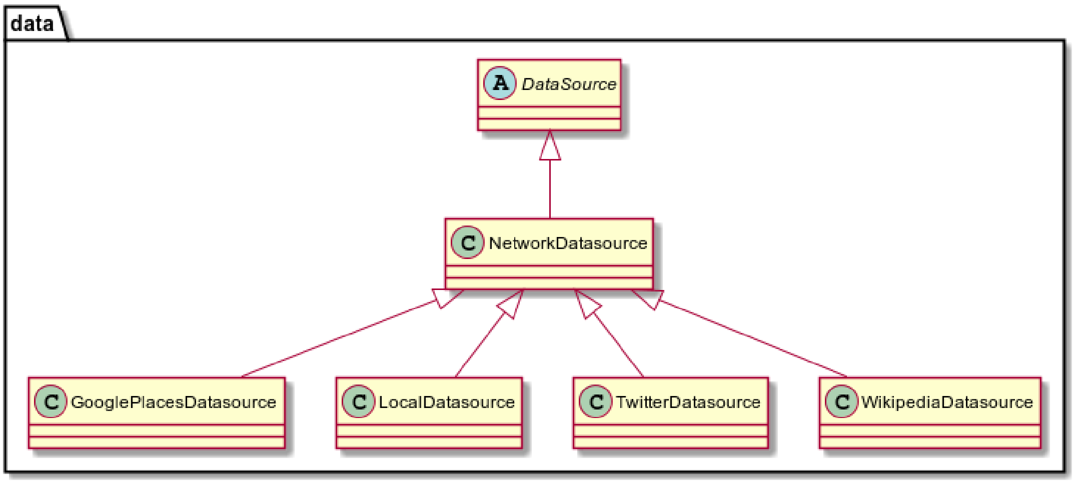


Figure 31: Datasource classes

## 6.9 User Settings and local markers

The user preferences and local markers are stored in two different ways:

User preferences are small quantities of data that can be stored as a key-value map. Android API offers an interface called `SharedPreferences` in which can be stored user settings.

```
SharedPreferences sp=PreferenceManager.getDefaultSharedPreferences(this);
boolean isGoogleChk=sp.getBoolean(SettingsActivity.KEY_PREF_GOOGLE, false);
```

User markers are stored in a basic file, because the length of the data is bigger than a setting value, this approach create a file in the storage of the phone (internal or external). Once again, this feature needs some permissions and require some level of privacy of the created files.

```
File userMarkers = new File(context.getFilesDir(), filename);
```

## 6.10 Implemented application

Following the mockups defined in the design phase, the application is composed by six screens, all those screens support the features to accomplished the objectives in this project. Below will be illustrated one by one the screens and explained how is the usability and accessibility.

Implementing this project by using the ARF framework was easy and powerful, because it provides enough features to recreate a full AR application. The application was developed taking in account the basic components that needs an AR Browser, as well as the small details that enhance the user experience by implementing an easy navigation and intuitive design.

The application starts when user select the application launcher `ARBrowser` in the smartphone. Initially the application shows a splash screen (Figure 32) with the name "Augmented Reality Browser", application takes a few seconds to start.

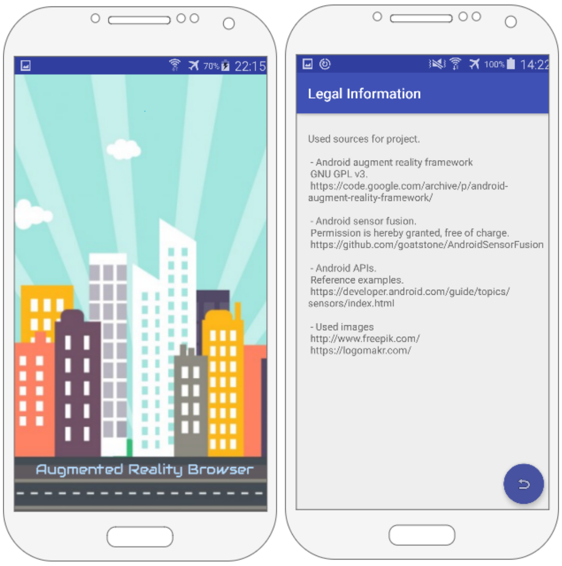


Figure 32: Initial screen and licenses information

Besides splash screen appears the information screen. Legal information is displayed in that screen, it is listed the used sources[6][17][17][18], by providing URL and license of use.

Below is illustrated the elements that belong to the augmented view (Figure 33). The image presents the components of a AR browser, it was defined based on the basic elements that need an AR browser (section 2.6.1).

### 6.10.1 Setting up application

In the main screen appears a lateral menu by swipping from left to right. The menu lists all the settings options in the application (Figure 34). All the settings is store in the preferences of

[17]Start image: `https://logomakr.com/`
[18]Another images: `http://www.freepik.com/`

Figure 33: Anatomy implemented app

the application, once it is need a prefrence is invoke by using the `PreferencesManager` with the respective `KEY`.

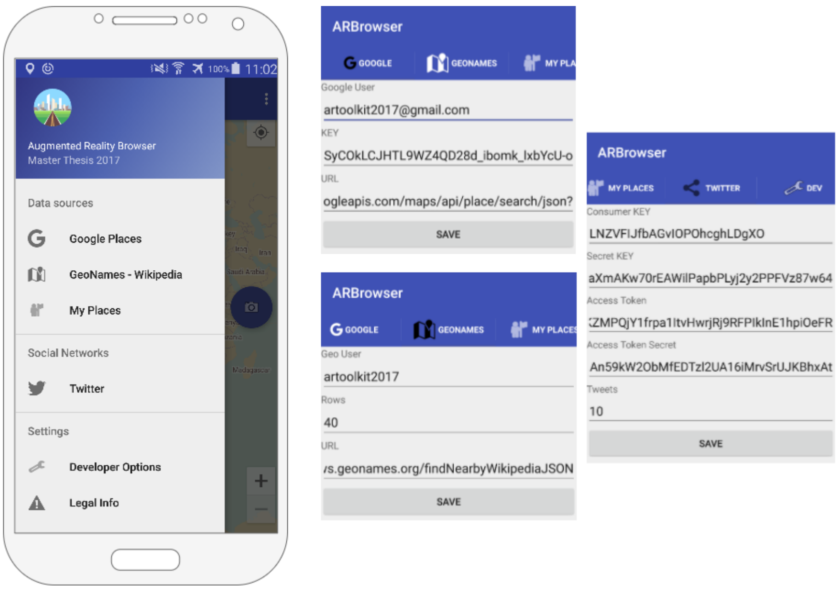

Figure 34: Application lateral menu settings

The settings include the following options to setup the APIs:

- Google: Fields to define valid registered google user, developer `KEY` and invocation `URL`.
- Geonames: Fields to define free user, number of rows to collect and invocation `URL`.

- Twitter: Requires more setting options to authenticate the user to access the twitter data. The authentication is under the`OAUTH` protocol and the `KEYs` are provided by registering in twitter developer community[19].

### 6.10.2 Selection data sources and sensors

To accesses to the configuration of data sources and sensors, the user needs to touch the toolbar and it will appear the configuration menu. The configuration is also stored in the preferences of the application, this configuration allow users to define the behaviour of the application by defining the preferred data sources, reliable sensors, and most used filters (place types[20] [21]).

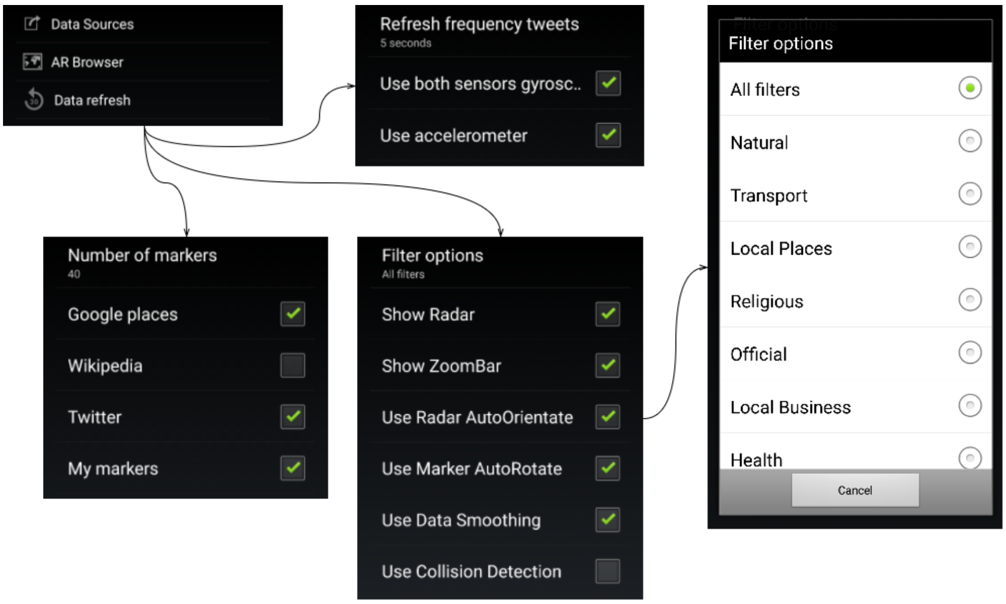


Figure 35: Configurating data sources, filters and sensors

The place types are defined in the following URL, in the application are grouped by relation between them, i.e.:

- Local places: `museum, art_gallery`.
- Local business: `cafe, bakery`.
- Natural places: `mountain, river, natural_feature`.
- Transportation: `bus_station, train_station`.
- Religious places: `mosque, church`.

[19] Twitter dev.: `https://dev.twitter.com/resources/signup`
[20] Google: `https://developers.google.com/places/supported_types`
[21] Geonames: `http://www.geonames.org/wikipedia/wikipedia_features.html`

In this menu are listed three options: Data sources works to enable/disable sources to get information, AR browser presents all the visible components of the browser and filter options, and Data refresh presents selected sensors and time to refresh tweets in the panel.

By using both available sensors gyrosscope and accelerometer, it will improve the rendering of the marks in the AR view. It was one of the most important improvements in the application, when selected both sensors the application will use the class `SensorFusion`.

### 6.10.3 Save user markers

A special improvement in the application is save markers (Figure 36), users can store their current location by touching the third button in the AR browser. When button is touched it will display a dialog box to save details of the current place, like name and a rating.

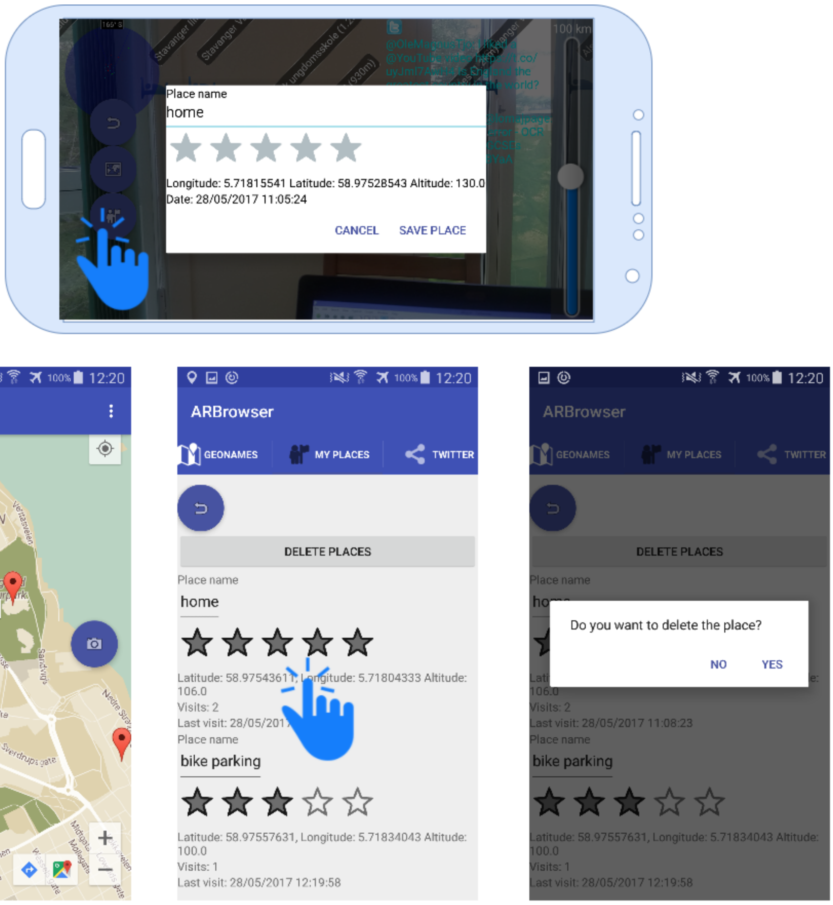


Figure 36: Saving user places

After saving current position, it will appear a new point in the radar and the map. Users can manage the markers by opening the lateral menu section "My Places". In that section user can list and delete the markers.

### 6.10.4 Calibration mode

This improvement (Figure 37) is an option that works by implementing the method `onAccuracyChange` and enabling the option in the configuration tab (Lateral menu section "Dev"). Accuracy is obtained once sensor get new values and it is validated according with the following levels: High is 3, Medium is 2, Low is 1, and Unreliable is 0.


Figure 37: Calibrating compass sensor

Once the application detects that the accuracy is less than the configured parameter, it will appear a dialog box with the instructions to calibrate the compass.

### 6.10.5 Additional browser features

Scroll list with type of places, dialog box with detailed information, and twitter panel were included in the development of the project (Figure 38). The objective of those features was to improve the user experience by helping to the interaction with the results and providing additional information in the panel with nearby tweets.

List with type of places (first red box) helps to filter the markers and improve the results, this filters works in parallel with the kind of selected data sources.

Once it is executed the search, the user can select anyone mark by touching the text with the name and it will appear detailed information (second red box). The wikipedia data source provides additional definition of the markers, while Google places provides more information about location.

Twitter panel will refresh every `x` seconds (configured by user) the tweets. The panel will be

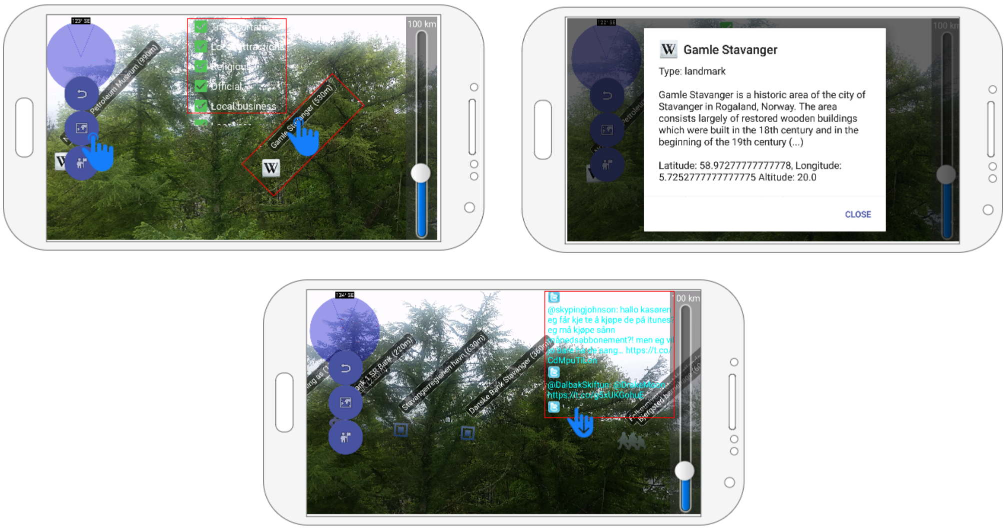


Figure 38: Additional features

visible if the user selects the Twitter data source. It will be listed by presenting the tweets and respective user.

### 6.10.6 Google map integration

Map integration (Figure 39) works by adding a `MapView` layer from the `Google maps API`. Map contains buttons to zoom, location button, and a list of options to control the map interface.

The map will show all the places from the search result, by adding a red mark with the name, those marks will be refresh every time the user executes a new search. When user select a mark, application will present two additional Google options: First option to find the route and second option to find information about place.

The location in the `SensorActivity` class was improved by using a method that validate if the current location is better that the previous one. It is best if is more accurate the location provider and if the current location is significantly new.

# 7 Evaluation

Below is presented the findings that was encountered through development of current project:

1. Detected issues in android applications.

   Android framework provides all the tools to develop an interesting AR application with significant features, but one fact to consider is that not all the devices have the same behaviour, fortunately new smartphones bring high specifications to present good performances in this kind of applications.

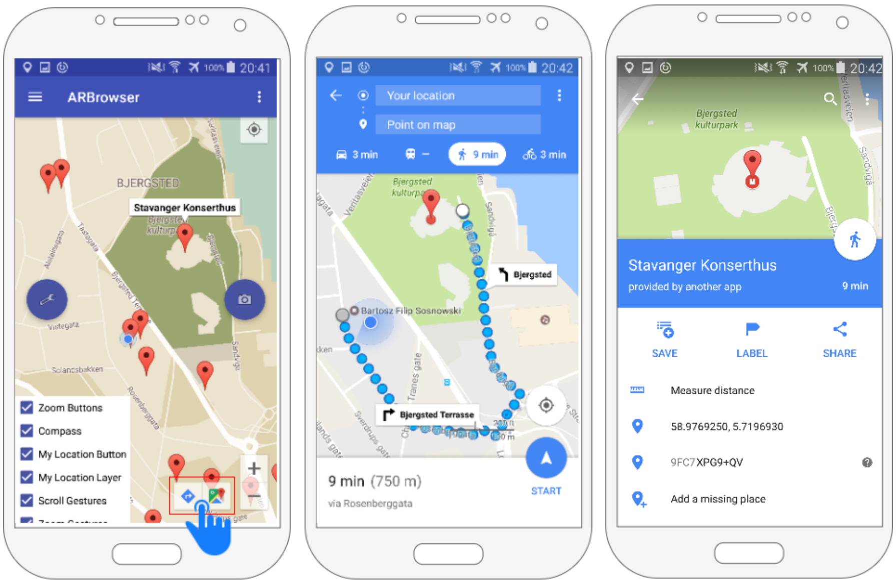


Figure 39: Google map integration

Development of Android applications are more challenging than IOS platform, due to the greater number of different devices that android supports. This wide support increases the difference between devices and the error rate in the development. Support a broad range of devices and previous versions of Android API are points to consider so that the application can be installed and executed without problems.

2. Sensor behavior.

   The present AR Browser provides better performance by adding more accurate sensor like the gyroscope. Information is provided in a very short period but needs to be processed in the same speed.

   By adding a new sensor is not enough to improve application performance, the signals include some levels of noise that needs to be cleaned. It was included different methods to process and clean the signal as filters and data fusion. By smoothing the signal with filters, it was reduced the noise and by implementing a method to fuse data between sensors the results were improved.

3. Other reliable implementations.

   Application behaviour presents an acceptable throughput, signal noise was removed but still it does not have a very good performance in the view, because the markers on some occasions present an unreliable location or position in the screen. Due to those kind of problems, other projects use simultaneous localization and mapping technologies (SLAM), to provide better AR applications[24].

   SLAM based applications can reliably manage fast motion and strong rotation[25], guaranteeing better AR experiences despite of strong changes in the values collected by the sensors in the smartphone. SLAM technologies implement specific algorithms like Kalman filters and particle filter. DroidAR[22] and Wikitude implement this approach.

4. Twitter restrictions.

   Nowadays data is provided in big amounts on internet, in some cases violating privacy of information sources. Twitter to protect privacy of users avoids delivering the location of the tweets, by allowing to create georeferenced queries but without provide exact location of the user.

5. Comparison related projects.

   The current project in comparison with previous developments[1] provides several functionalities and a more intuitive AR Browser interface to execute location-based queries. The additional capabilities enhance the user experience by allowing to select between the different data sources and available sensors in the smartphone (accelerometer or gyroscope), in addition, it has a set of options to configure the application.

   Furthermore, this project provides another interesting features like searching for nearby tweets, compass calibration, last version of ARF, store user preferences, and manage user markers.

[22] DroidAR: urlhttp://bitstars.github.io/droidar/

# 8 Conclusion

AR Browser for Android Smartphones is a powerful tool to help people for search and reach nearby places by offering AR features like projection of POI in the camera view, in addition to, GPS and sensors integration.

AR software enhances the reality by adding virtual components that users can interact with, allowing to create a mixed environment between the reality and virtuality. In this way, appears new forms of interaction in several areas.

Android API allows to build an application with significant capabilities by offering an important list of services to interact with the hardware of the devices and full interoperability with other libraries java-based.

Location-based applications is a decision-making tool due to the real-time capabilities to collect and provide information to users. Capabilities like search for nearby places based on their current position.

Sensors accuracy depends in a high measure from the quality of the hardware, in the other hand, data sensed can be enhanced by using software procedures as a low/high pass filters and fusion algorithms to improve quality of information.

Accelerometer and gyroscope produce high amount of data in a short period, due to the sensibility of the sensors.

An incorrectly accesses to the sensors produces an overheat and high-speed decrease of battery. It needs a well-defined use of register and unregister listener methods.

Exists many free-use web services for collecting information to produce knowledge, like Google Places, Geonames, and real-time data from social networks.

Location-based services can be improved by skipping small changes in the position and timestamps of current location, also by validating the accuracy of the available providers.

AR applications can be implemented for different areas due to the several features provided like image recognition and motion detection. Those areas cover from educational purposes to health sector.

## References


[1] Jose Carlos Tejada Saracho. *Mountain view application for smartphones.* University of Stavanger, 2013.

[2] Lester Madden. *Professional Augmented Reality Browsers for Smartphones: Programming for Junaio, Layar and Wikitude.* Wiley Publishing, 1st edition, 2011.

[3] Google. Android Platform, version 7.1, API level 25, package reference. `https://developer.android.com/reference/packages.html`, 2016.

[4] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. pages 282–292, 1994.

[5] Jens Grubert. *Augmented Reality for Android Application Development.* Packt Publishing, 1st edition, 2013.

[6] Justin Wetherell. ARF - Android augment reality framework. `https://code.google.com/archive/p/android-augment-reality-framework/`, 2015.

[7] Ronan Schwarz, Phil Dutson, James Steele, and Nelson To. *The Android Developer's Cookbook: Building Applications with the Android SDK.* Addison-Wesley Professional, 2nd edition, 2013.

[8] D. K. Shaeffer. MEMS inertial sensors: A tutorial overview. *Communications Magazine, IEEE*, 51(4):100–109, April 2013.

[9] Diego Emilio Serrano and Farrokh Ayazi. MEMS inertial sensors. pages 327–353, 2015.

[10] EETAsia. Smartphone basics: Connectivity and sensors. `http://archive.eetasia.com/www.eetasia.com/ART_8800701579_480400_TA_e71be411.htm`, 2014.

[11] Henry M. Stommel and Dennis W. Moore. *An Introduction to the Coriolis Force.* Columbia University Press, New York, USA, 1989.

[12] AKM. Electronic Compass. `http://www.akm.com/akm/en/product/detail/0001/`, 2016.

[13] Greg Milette and Adam Stroud. *Professional Android Sensor Programming.* Wrox Press Ltd., Birmingham, UK, UK, 1st edition, 2012.

[14] Donald E. Knuth. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[15] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter.* University of North Carolina, Chapel Hill, NC 27599-3175, 1995.

[16] Shane Colton. The Balance Filter. `http://d1.amobbs.com/bbs_upload782111/files_44/ourdev_665531S2JZG6.pdf`, 2007.

[17] Paul Lawitzki. Android sensor fusion tutorial. `http://www.thousand-thoughts.com/articles/sensorfusion/`, 2014.

[18] Raghav Sood. *Pro Android Augmented Reality.* Apress, 1st edition, 2012.

[19] Gartner. Gartner says five of top 10 worldwide mobile phone vendors increased sales in second quarter of 2016. `http://www.gartner.com/newsroom/id/3415117`, 2016.

[20] James Steele and Nelson To. *The Android Developer's Cookbook: Building Applications with the Android SDK.* Addison-Wesley Professional, 1st edition, 2010.

[21] Rick Rogers, John Lombardo, Zigurd Mednieks, and Blake Meike. *Android Application Development: Programming with the Google SDK.* O'Reilly Media, Inc., 1st edition, 2009.

[22] Tony Mullen. *Prototyping Augmented Reality.* SYBEX Inc., Alameda, CA, USA, 1st edition, 2011.

[23] Google Android. Requesting Permissions at Run Time. `https://developer.android.com/training/permissions/requesting.html`, 2015.

[24] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067, June 2007.

[25] Liu Haomin, Zhang Guofeng, and Bao1 Hujun. Robust Keyframe-based Monocular SLAM for Augmented Reality. *IEEE International Symposium on Mixed and Augmented Reality*, 2016.

# A Installing the AR Browser

## Before start

Before beginning installation, it is needed to know the principal folders of the application and license keys. The keys are provided by Google Developers, Twitter and Geonames.

The sources of the project are attached to the PDF. Once the project is imported three folders compose the app: manifest, java, and res. The manifest folder has the XML file where is provided important information about the project like permissions and components.

Java folder groups all the implemented classes and res folder contains all the used resources like images and layouts.

## System Requirements

- Android Studio IDE to import the project.
- Android version 7.1 API level 25.
- Use embedded JDK.

## Installation Steps

- Step 1: Clean and build the application: Android Studio offers the possibility of clean and build the app to validate the source code and check the possible integration problems like missing or deprecated libraries.
- Step 2: Generate APK: Once the app has been build, it is generated the APK to install in the smartphone.
- Step 3: Installing the APK: Two options are present for the installation of the app, it must be selected the Deployment target: Connected Devices or Virtual Devices. The Connected Device was a Samsung S5 Active with Gyroscope, Accelerometer, and Magnetic field sensors.
- Step 4: Test/Debug the APK (Optional): The last step is to test the app in the smartphone. When the AR Browser is installed, it can be tested (Run or Debug mode) to validate all the features.