



Guilherme Jorge Birra Seabra

Bachelor in Computer Science

**Improving Mobile GIS applications through the
identification of Geographic Context**

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Armanda Rodrigues, Assistant Professor,
NOVA University of Lisbon

Examination Committee



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

March, 2019

Improving Mobile GIS applications through the identification of Geographic Context

Copyright © Guilherme Jorge Birra Seabra, Faculty of Sciences and Technology, NOVA University of Lisbon.

The Faculty of Sciences and Technology and the NOVA University of Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

In memory of Carlos de Brito.

ACKNOWLEDGEMENTS

First I would like to express my gratitude to my advisor Armanda Rodrigues for her tireless guidance, support and fundamental feedback throughout this dissertation.

I would also like to thank my friends and colleagues, who helped me throughout this journey, for all the motivation and all the shared sweat and tears. More specifically, I'd like to thank Alberto, Pedro Lopes, Ana Henriques and the "Oracle's Squad" for their incredible support throughout all these years.

Last but not least, I would like to thank all my family, especially my parents, who have always helped me reach my personal goals. Finally, I would like to thank Joana Pereira for being an incredible friend, teacher and unconditional supporter.

I will never forget any of you, this would not have been possible without your help.

*“Se soubesse que amanhã morria
E a Primavera era depois de amanhã,
Morreria contente, porque ela era depois de amanhã”*

Alberto Caeiro, “Quando vier a Primavera”

ABSTRACT

Mobile devices are becoming increasingly popular. Their functionalities have become more than just making phone calls, due to regular improvements to these devices. Thus, with their notable increase in computational power, these devices have become able to support applications based on georeferenced data. By allowing the manipulation, visualisation and sharing of such data, these applications (supported by, for example, Google Maps or OpenStreetMap) have also shown an increasingly higher popularity.

In this dissertation, we developed an adaptive Geographic Information System for Android devices, which displays relevant information to the user, based on the detected geographic context. The platform is supported by the concept of a context adaptation model, which enables the identification of particular situations in the context of the user, and the consequent adaptation of the application's interface to the identified event. The context of the user is composed of the information collected by the sensors present in most mobile devices, which also enables the system to automatically adapt its content and thus become more relevant (according to the detected conditions), contributing for a better user experience. The relevant events to be listened to and the actions to be taken accordingly are managed in an administrator online tool, allowing for simplified software maintenance. By defining adaptation rules on a Web platform, the administrators are able to configure the Android application's behaviour without having to change the existing code.

Finally, the developed platform was tested on a prototype of a Tourism application. The system was evaluated in two distinct parts - Web platform and Android application - by several participants, who agreed that the second is easy to use, while the first requires some previous learning.

Keywords: Geographic Information Systems, Adaptation, Context, Context-Awareness, Georeferencing, Android

RESUMO

Os dispositivos móveis são cada vez mais populares. Devido a melhorias constantes destes dispositivos, as suas funcionalidades tornaram-se muito mais do que simplesmente efetuar chamadas. Assim, com a notável evolução do seu poder computacional, estes dispositivos passaram a ser capazes de suportar aplicações baseadas em dados georreferenciados. Ao possibilitar a manipulação, visualização e partilha de tais dados, estas aplicações (suportadas, por exemplo, por Google Maps ou OpenStreetMap) têm também apresentado um crescimento e popularidade cada vez maior.

Nesta dissertação, foi desenvolvido um Sistema de Informação Geográfica adaptativo para dispositivos Android, que mostra informações relevantes ao utilizador, com base no contexto geográfico detetado. A plataforma é apoiada por um modelo de adaptação contextual, que permite a identificação de situações particulares no contexto do utilizador, e a conseqüente adaptação da interface da aplicação ao evento identificado. O contexto do utilizador é composto pelas informações recolhidas pelos sensores presentes na maioria dos dispositivos móveis, que também permite ao sistema adaptar automaticamente o seu conteúdo e, assim, tornar-se mais relevante (de acordo com as condições detectadas), contribuindo para uma melhor experiência de utilizador. Os eventos relevantes a serem tidos em conta e as ações a serem tomadas em conformidade são geridos através de uma ferramenta online de administrador, permitindo uma manutenção de software simples. Ao definir regras de adaptação numa plataforma Web, os administradores podem configurar o comportamento da aplicação Android sem precisar de alterar o código existente.

Finalmente, a plataforma desenvolvida foi testada num protótipo de uma aplicação de Turismo. O sistema foi avaliado em duas partes - plataforma Web e aplicação Android - por vários participantes, que concordaram que a segunda é fácil de utilizar, enquanto a primeira requer alguma aprendizagem.

Palavras-chave: Sistemas de Informação Geográfica, Adaptação, Contexto, Georreferenciação, Android

CONTENTS

List of Figures	xvii
List of Tables	xix
Listings	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Objectives	3
1.4 Contributions of the Dissertation	3
1.5 Document Structure	4
2 Background	5
2.1 Geographic Information Systems	5
2.1.1 Georeferencing	7
2.1.2 Web GIS	8
2.2 Context	10
2.2.1 Adaptation	12
2.3 Conclusion	17
3 Related Work	19
3.1 Modelling Context with Trigger/Action Rules	19
3.2 Web GIS Examples	23
3.3 Platform for Coastal Structure analysis	24
3.4 Automatic Activity Detection	25
3.5 LiveTeams - Emergency Teams Management	26
3.6 Summary	26
4 Context Adaptation Model	29
5 Implementation	33
5.1 System Modelling	33
5.1.1 Technologies	33

CONTENTS

5.1.2	Architecture	36
5.2	Points of Interest Server	38
5.3	Configuration Platform	39
5.3.1	Entities, Attributes, Values, Comparison Operators	40
5.3.2	Adaptation Rule Creation	40
5.3.3	Database link	43
5.3.4	Initial Settings	44
5.3.5	NoSQL Database	45
5.3.6	RESTful API	50
5.4	Android application	51
5.4.1	Geographic Interface	51
5.4.2	Automatic Adaptation	53
5.5	Conclusion	60
6	Evaluation	63
6.1	Methodology	63
6.2	Summative Assessment	66
6.3	Formative Assessment	69
6.3.1	Configuration Platform	70
6.3.2	Android Application	76
6.4	Conclusion	83
7	Conclusions and Future Work	85
7.1	Conclusions	85
7.2	Future Work	86
	Bibliography	89
A	Given tasks	93
B	Evaluation Questionnaire	99
C	User Profiles and Information	111

LIST OF FIGURES

2.1	Different types of projections. Source: [32]	8
2.2	Web GIS composition. Source: [16]	9
2.3	Client-server systems architectures. Source: [33]	10
2.4	Relationship between location-aware, context-aware, pervasive and mobile computing. Source: [33].	11
2.5	Five categories of Context.	12
2.6	Adaptive Maps. Source: [30].	14
2.7	Adaptive Visualisation in GIS. Source: [29].	16
3.1	Authoring tool’s architecture.	21
3.2	List of all elements managed by the three environments collectively. Source: [26].	22
3.3	User’s current position and nearby bus stops, respectively. Source: [15].	24
3.4	Comparison between the mentioned projects’ ([9, 21, 27]) adaptation features.	28
4.1	Visual representation of an adaptation rule example.	30
4.2	Adaptation Rule Entity Model.	31
5.1	System architecture	37
5.2	Entity and Attribute creation page.	41
5.3	Rule Creation on the Configuration Platform.	42
5.4	Database Link page.	44
5.5	Database Link page.	45
5.6	Android application interface.	52
5.7	Android application details.	53
6.1	Results from the SUS Questionnaire (Configuration Platform).	67
6.2	Results from the SUS Questionnaire (Android application).	68
6.3	Comparison between adjective ratings and SUS scores.	69
6.4	Question 1: How would you classify your experience while using the platform?	70
6.5	Statement 1: I think it was easy to understand the relation between rules, triggers, actions, entities, attributes, values and comparison operators.	71
6.6	Statement 2: I think it was easy to create adaptation rules.	71

6.7	Statement 3: It is easy to verify which rules have been created.	72
6.8	Statement 4: It is easy to understand what each rule will perform.	72
6.9	Statement 5: The feature providing a description about entities and attributes is useful.	73
6.10	Statement 6: The feature concerning rule priorities is useful.	74
6.11	Statement 7: The platform's style adaptation to mobile devices is useful. . . .	74
6.12	Statement 8: The goals of the configuration tool are useful.	75
6.13	Statement 9: It makes sense to define the application's adaptation through the platform.	75
6.14	Question 2: Did you have any difficulty while using the platform?	76
6.15	Question 1: How would you classify your experience while using the application's interface?	77
6.16	Statement 1: It is useful that the application adapts itself automatically according to what was specified in the configuration platform.	77
6.17	Statement 2: I think it was easy to find the option to turn off the automatic adaptation.	78
6.18	Statement 3: I noticed that notifications pop up telling me when I get close to a POI.	79
6.19	Statement 4: It is useful when the application automatically opens the information of the closest POI.	79
6.20	Statement 5: It is useful when the application tells me I am getting near a new POI, even if it is not the closest.	80
6.21	Statement 6: It is useful that the application is able to change the proximity distance according to the movement speed.	81
6.22	Statement 7: I felt that the proximity distance when riding a car was adequate. .	81
6.23	Statement 8: I felt that the proximity distance while walking was adequate. . .	82
6.24	Statement 9: The fact that the map solely centred itself on the user when driving a car is useful.	82
6.25	Statement 10: I enjoyed the application better when the automatic adaptation was turned off.	83
6.26	Question 2: Did you have any difficulty while using the application?	84
C.1	Participants age.	111
C.2	Participants graduation.	112
C.3	Participants gender.	112

LIST OF TABLES

3.1	Decision Table Example.	22
5.1	RESTful API description.	39
5.2	Configuration platform's RESTful API description.	51
6.1	Results of the SUS Questionnaire (Configuration Platform).	67
6.2	Results of SUS Questionnaire (Android application).	68

LISTINGS

5.1	“settings” document structure.	45
5.2	“database” document structure.	46
5.3	“entities” document structure.	46
5.4	“rules” document structure.	49
5.5	“RuleController” trigger checking.	54

INTRODUCTION

This chapter describes how the subject of this dissertation emerged, as well as the general problem it addresses and the intended contributions.

As the name suggests, section 1.1 describes the main motivations for this work and why adaptation plays such an important role in a system. Furthermore, it explains why this is also interesting when applied to Geographic Information Systems.

Section 1.2 presents the general problems that this dissertation aims to address.

The main objectives of this dissertation are described in section 1.3 while the contributions are presented in section 1.4.

Finally, section 1.5 presents the general document structure.

1.1 Motivation

Mobile technologies are present in everyone's lives, in one way or another. Smartphones are the most popular of them, and are expected to reach 2.5 billion users in 2019 [31]. Furthermore, every year mobile devices feature higher computing resources, screen resolution, better network connection capabilities, etc. Additionally, more and more mobile data traffic is transmitted every year and this trend is expected to continue in the coming years [8]. Furthermore, mobile devices almost always include a wide array of sensors, capable of measuring multiple conditions such as ambient light and temperature, device rotation, etc. This characteristic makes mobile technologies often more desirable than regular computers like desktops and laptops, since they generally do not feature such sensors due to their lack of mobility.

GPS (Global Positioning System) is a technology that is often associated with mobile devices. Location-based services take advantage of an array of sensors provided by each mobile device (GPS being a popular choice) to detect the user's geographical location.

Furthermore, these applications are able to obtain contextual information from, for example, the device's sensors and the user's activity history, and use it to enhance the user's experience.

The development of sensor technologies in mobile devices offers new possibilities related to context. However, very frequently these applications do not take full advantage of this potential. Very often they only take into consideration a simple context aspect, such as time of the day or WiFi connection. Consequently, mobile applications still have room for improvement when it comes to adapting themselves to information coming from "outside" the device. There are four domains of context (the user, the devices, the environment, and social relationships)[18] and each of them is a great potential source of information. When these are taken into account, the system's adaptation capabilities can be largely increased.

As previously mentioned, the usage of GPS in mobile devices is very popular. Applications that use this technology typically include a map in their interface, to show the user's location as it is easy to use and understand. For example, mobile geographic information system applications can obtain the user's location through GPS so it can be shown on a map.

However, there is a frequent need for users to obtain relevant geographical information depending on the context. For example, when the user is using the device late at night, showing points of interest that are only enjoyable during the day is not very useful. If the user is not familiar with the city or village he is in, this issue becomes even more crucial, as he will not know what information is useful. Hence, in tourism applications there is a greater need for context-dependent adaptation of its features and interface. Although some applications feature adaptive possibilities (some of these will be mentioned later in Chapter 3) and technology grants access to many spatial context dimensions, Geographic Information Systems (GIS) adaptations are still lacking. For example, if the user is moving quickly, the information should also change accordingly (e.g. by displaying POIs further away from the user or decreasing the level of detail).

It is also important to improve the user's general experience when using a GIS, by systematically adapting the platform's information and interface according to the detected context conditions. Furthermore, better results can be achieved by allowing users to manually change elements of the interface and map to their liking.

As Paternò *et al* have suggested in [17–20], these adaptations may be modelled through a framework for storing and dealing with spatial context data. Lastly, this framework can be used for supporting a mechanism which reacts to context alterations, through configuration-defined actions.

1.2 Problem Statement

Taking into account what has been previously mentioned in section 1.1, the general problems that this dissertation aims to address can be summarised as:

- Is it possible to improve the mobile application user experience, by taking advantage of spatial context information provided by modern devices?

This is a general problem that is going to be addressed in the tourism field of mobile applications, according to the following three questions:

1. How to capture the spatial behaviour of a tourist through the mobile device's sensors?
2. How to model the behaviour and the tourist's spatial context, independently from the rest of the application's functionalities?
3. How to take advantage of this information to improve the tourist's experience when using the application?

1.3 Objectives

The definition of the problem supports thus the main objectives of this dissertation, which are the following:

- Developing a context-based adaptation model for mobile applications (more specifically, to tourism applications);
- Developing a set of automatic adaptations based on certain context characteristics for this application;
- Developing an application (prototype) as a proof of concept of the proposed model. This application shall be supported by an array of POIs on an interactive map, and will adapt to the situation identified by the model. This system shall also have a context management interface to be managed by a knowledgeable administrator responsible for it.

Thus, this dissertation focuses on the development of an interactive mobile tourism GIS that takes into account multiple context factors during runtime, measured using the device's sensors. This application systematically adapts itself according to the automatic adaptations designed by knowledgeable administrators and by performing these according to periodic measurements.

1.4 Contributions of the Dissertation

The main contributions of this dissertation are related to the developed system due to the following reasons:

- The generic context-based automatic adaptation model, validated for tourism applications;

- The configuration platform which will provide a way of designing context-based automatic adaptations meant to be used by an external application;
- The Android application (proof of concept) which will adapt itself according to the detected context and what was configured by the knowledgeable administrators in the external platform;
- Facilitating and improving the tourist experience, by enabling adaptation to their movements and actions as needed.

1.5 Document Structure

The remainder of this document is structured as follows:

- Chapter 2 presents important concepts needed to better understand the work developed in this dissertation as well as their respective state of the art.
- Chapter 3 presents the related work where practical applications of trigger/action rules and some context-aware geographical information systems are described.
- Chapter 4 describes the context adaptation model which was developed to support the dissertation's system automatic adaptations.
- Chapter 5 presents the technical description of this dissertation's project. This chapter also presents the architecture of the system, along with the technologies that were used during its development. Finally, the developed system components are also presented and shown in this chapter.
- Chapter 6 describes the methodology and presents the results of the evaluation of both the developed mobile application and configuration platform.
- Chapter 7 draws conclusions about the outcome of this dissertation and also presents further developments as future work.

BACKGROUND

This chapter describes the fundamental concepts needed to better understand this dissertation.

Geographic Information Systems are presented in section 2.1. This includes an explanation of common functionalities supported by GI systems, as well as an insight into georeferencing. Furthermore, an explanation on map projections will also be provided. Finally, Web GI systems will also be presented, as they are an important part of this dissertation.

In section 2.2, concepts such as context and context-aware are detailed. This section also includes an overview on flexible systems and their categories. Finally, the description of computing types such as location-aware and pervasive is provided.

Lastly, section 2.3 summarises the information present in this chapter and gives an overview on why it is important for this dissertation.

2.1 Geographic Information Systems

Geographic Information (GI) has always been an important factor in people's lives. According to Longley in [25] (section 1.1), geographic information records "where", "what" and sometimes also "when". When using Geographic Information on a computer system, it is common to refer to it as a *GIS (Geographic Information System)*.

A GIS is described by Worboys in [33]'s first chapter as "a special type of computer-based information system tailored to store, process and manipulate geospatial data", or, in a similar analysis by Longley in section 1.3 of [25], it is described by a set of tools for collecting, storing, processing, analysing and visualising geographic information.

What differentiates a general information system from a geographic information system is the geographic data. GI systems are concerned with associating regular data with

a spatial and temporal dimension. Hence, this allows the system to feature particular functionalities such as planning a road trip, in a faster, easier to understand way when compared to general information systems. However, the specificity of geographic data storage and collection must be taken into account to get the most from this type of application.

The central part of a GIS is the database [33] (chapter 1). The structuring of data is important because the volume of data is large, since many observations are often made of the phenomena to be recorded. Furthermore, the functionalities featured by the GIS may also be more or less efficient, depending on this structuring. Hence, the data structuring is also going to depend on the necessary algorithms. However, given the visualisation needs, the interface component must also not be neglected.

GI systems are many times incorrectly associated with SIS (*Spatial Information Systems*). While they are often associated with each other, this may not happen at all times. This is because a SIS may handle data on a much larger range than a GIS [33]. For example, a SIS may be used for visualising data on molecular configurations. Another misconception is referring to a geographic database as an image database. While the first provides the database functionality of a GIS, the latter can be described as a collection of images. In other words, an image database does not take into account structural interrelationships or topological features of the stored objects. An example of an image database is a database of a basketball's league players, which has no spatial or geographic information associated with the objects. Summarising, computer graphics are responsible for displaying information of a visual nature, while a GIS provides more than just graphics processing [33] (Chapter 1).

GIS is able to reveal omitted inter-relationships between information and patterns, allowing the users to solve spatial problems and make decisions based on what is displayed by the GIS. Among many functionalities supported by GI systems, some of the most common are (as described by Worboys and Duckham [33]):

Resources inventory

Allows for the combination of several sources of information into one view.

Network analysis

Drawing itineraries based on route information, which can be interpreted as a graph, internally.

Terrain analysis

Analysis and generation of new information based on topographic data.

Layer-based analysis

Combination and data analysis of data from a variety of sources.

A particular characteristic of a GIS is the integration in a reference system. It allows for locating one object relative to another, integrating multiple spatial data sets into one, and minimising collection and treatment errors.

2.1.1 Georeferencing

Spatially referencing a location is widely needed in everyone's lives. It may be as simple as saying that a certain object is situated five centimetres away from the walls of the southern corner of a room. A coordinate system can also be used, which would facilitate spatially referencing objects.

A coordinate system is an agreed upon way of representing locations. It requires two axes (for two-dimensional referencing), an origin and an unit size [14, 23]. Without these definitions, example coordinates such as (3, 7) and (2.8, 2) are meaningless, as it would be impossible to know where they would be located.

One of the coordinate systems used to specify locations on the entire Earth is the Geocentric Coordinate System. In this system, it is possible to specify locations on Earth by using just two angles - latitude and longitude. Latitude is defined by the angle formed between the a specific location and the Equator line. This angle is measured between 90° north and 90° south. In other words, latitude is used to determine how far above (or below) the equator an object is. On the other hand, longitude is characterised by the angle formed between a specific location and the Greenwich meridian. Simply put, longitude measures how far around the Earth an object is, compared to the prime meridian (Greenwich meridian). This angle has a minimum value of 180° east and 180° west, where a location featuring a 0° longitude is said to be located along the Greenwich meridian [3, 12, 13, 23].

This geocentric system is able to specify locations by using just two angles because it assumes that Earth is a perfect sphere. However, this is not true - Earth has a very irregular shape, that better resembles an ellipsoid. Due to this fact, there are variations to the geocentric coordinate system that use latitude, longitude and altitude to specify locations on an ellipsoid.

An ellipsoid that is close to the shape of the Earth in a determined site is called a *datum*. Datums are used because it is too complex to model Earth's actual, ever changing shape [2]. While there is no such thing as a perfect datum, WGS84¹ is a reference system whose datum is widely used in global systems, including GPS and Google Earth².

2.1.1.1 Projecting the Earth

Due to reasons like affordability, easy storing, ease of use and transportation, two dimensional maps are still largely used. A projection is a transformation from geographic coordinates (from a spherical surface) to a planar representation, through a mathematical

¹<https://confluence.qps.nl/qinsy/en/world-geodetic-system-1984-wgs84-29855173.html>

²<https://www.google.com/earth/>

formula[4]. However, it is impossible to use a flat representation of the Earth without distortions. Frequent distortion types are angular distortions, distance distortions and area distortions. However, projections can still preserve certain properties, such as *conformal* or *equal area* properties. Yet, a projection is only able to preserve at most one of these properties, not both [25] (section 4.8). Thus, when projecting a region in the plan, we have to decide which projection to use, taking into account the objectives of the representation. If the purpose is navigation, then the directions have to be preserved. If the objective is to maintain the relationship between the area of the region on the map and the area in reality, then an equivalent projection should be used, which maintains the areas, but may distort the shape of the region. Finally, it is necessary to take into account that when the focus falls on a particular region, the error is minimised there, but may be increased elsewhere.

Conformal property

The conformal property ensures that the scales of projection in the x and y directions are always equal. In other words, the shapes of small features on the Earth’s surface are preserved. A projection that respects this property is the conformal projection, represented in Figure 2.1b.

Equal area property

According to Longley et al in [25]’s section 4.8, this property “ensures that areas measured on the map are always the same proportion to areas measured on the Earth’s surface”. A projection that respects this property is the equivalent projection, represented in Figure 2.1a.

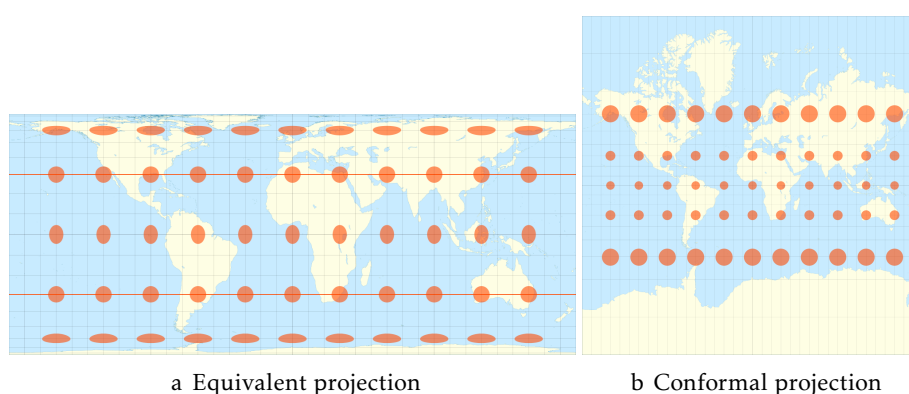


Figure 2.1: Different types of projections. Source: [32]

2.1.2 Web GIS

Web GIS is a GIS where the communication between server and client is done through web technologies, making it a type of distributed information system. A Web GIS features

at least one server and one client. The server is a Web application server, while the client may be any device using the Web technology to communicate with the server. The client sends requests to the server via HTTP (*Hypertext Transfer Protocol*). The server performs the GIS operation requested by the user and sends a response, via HTTP [16] (Chapter 1.3). A common misconception is to think of a Web GIS as a GIS running in a web browser. As shown in Figure 2.2a, this is not true, as a desktop or even mobile client can use Web technology to communicate with the Web GIS server.

The Internet supports many services, Web being one of them. A GIS could potentially be provided to a client by FTP (*File Transfer Protocol*), even if it is not so natural. In this case, the system would be called an Internet GIS. Thus, Internet GIS is broader than Web GIS - Web GIS is a sub-type of Internet GIS, as shown in Figure 2.2b. Even though Web GIS is merely a part of Internet GIS, the Web is the most commonly used Internet service. Hence, this makes Web GIS a very popular form of Internet GIS [16](Chapter 1.3), as users typically consider a Web GIS application a website application like many others.

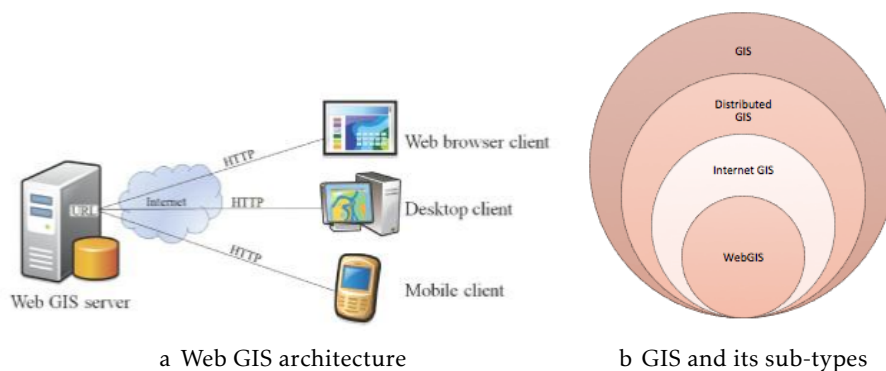


Figure 2.2: Web GIS composition. Source: [16]

2.1.2.1 Web GIS Architecture

A computer-based information system architecture has two distinguishing characteristics: *interoperability* and *modularity* [33](Chapter 7). Interoperability is the ability of multiple information systems to share data or processing capabilities. This is particularly important to GI systems, as they often need to gather and/or integrate data from multiple sources. Modularity on the other hand, is the extent to which an information system can be constructed from independent units with clearly defined functions. This characteristic is also important because it makes complex GIS software easier to develop, maintain and adapt to meet the specified requirements [33](Chapter 7).

Furthermore, Figure 2.2a is a representation of a typical *client-server system*. This type of system provides a delineation between the functions and responsibilities of different systems within the application. While a *server* offers a particular service to other systems on the network, a *client* is an information system that consumes these services, as shown

in Figure 2.3a. Communication between server and client is done through a *protocol*, which is, simply put, a standard format for communication (e.g. HTTP). However, Web GI systems commonly use a *multi-tier* client-server system. These systems are essentially composed of chains of regular client-server systems (as represented in Figure 2.3b), where the intermediate "middle-tier" acts as a server to the client and as a client to the server. This way, modularity is increased, with specific functions allocated to particular tiers. When a client sends a request for an action on a Web GIS, it is typically propagated through the necessary servers for completing the task.

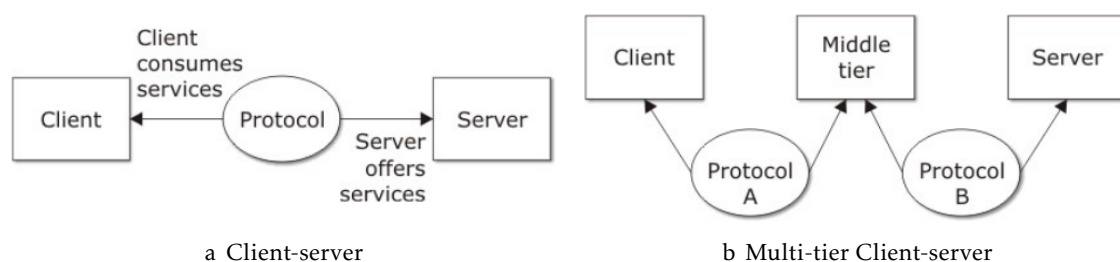


Figure 2.3: Client-server systems architectures. Source: [33]

2.2 Context

Today's applications have functionalities that satisfy most of the users. Furthermore, these applications also feature great usability and intuitive ways of interacting with their users. However, context information, which is accessible thanks to technological developments, can be harnessed to improve the user experience.

Through a rich language and common knowledge of everyday situations, humans are able to convey their ideas effectively. Furthermore, it is easier to share information when both participants share the same location. This is because their *context* is identical and there is a set of implicit circumstantial information that both humans understand and refer to. On a remote conversation (e.g. phone call), the *context* is not the same, making it harder to convey information. Thus, taking *context* into consideration is advantageous, as it increases the information relevancy. For example, when trying to inform the user on open restaurants for dinner, the results will be more relevant if the system knows the current time, so it does not suggest closed places.

Context is defined by Dey [10] as "any information that can be used to characterize the situation of an entity". It can be considered a powerful resource, which can be used to increase the richness of communication between user and application. By utilising what surrounds their interaction (e.g. room temperature, user's mood), it is possible to obtain more useful computational services. As mentioned previously, spatial context captured by a device's sensors can be used to improve the user experience. However, some applications are prepared to take into account the user's context during runtime,

which makes them *context-aware* applications (some will be mentioned in Chapter 3).

According to Dey [10], “a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task”. In a similar approach, Worboys and Duckham define context-aware computing as “the use of sensors and other sources of information about a user’s context to provide more relevant information and services” in [33](Chapter 7.5).

Despite being often used with the same meaning, *pervasive*, *mobile*, *context-aware* and *location-aware* computing are not equivalents. As shown in Figure 2.4, there are applications that feature all of these concepts, but they are distinct [33]:

Location-aware computing By utilising information about the user’s location, it provides more relevant information and services to the same user.

Pervasive computing Also called *ubiquitous computing*, pervasive computing describes the idea that everyday objects can become unseen personal assistants.

Mobile computing Mobile computing is concerned with information systems that can move around with the user. This term “mobile” may refer to at least three types of mobility, relevant to information systems - software, device and user mobility.

- Software mobility: migration of software applications or agents between different computing devices or platforms;
- Device mobility: ability of computing devices to move around, typically while retaining a network connection;
- User mobility: refers to the mobility of the user through an environment.

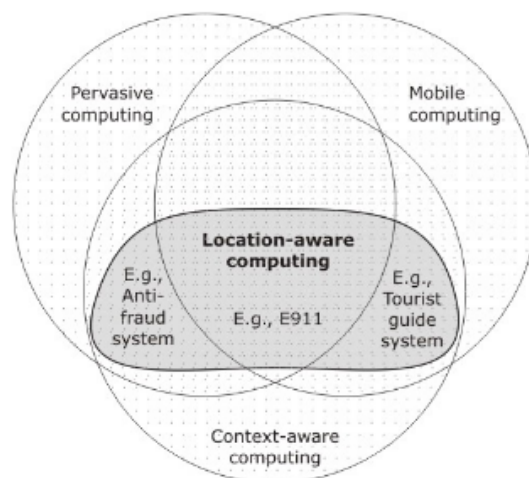


Figure 2.4: Relationship between location-aware, context-aware, pervasive and mobile computing. Source: [33].

2.2.1 Adaptation

Applications and their interfaces are typically designed to fit the system to the basis of the requirements analysis. However, this approach aims for a general type of user, which is not ideal, as some users may not relate to this “general user” at all. A flexible system improves the correspondence between the user, task and system characteristics. According to Sarjakoski and Sarjakoski in [30], there are two variations of a flexible system: adaptive and adaptable.

Adaptable systems An adaptable system provides the user with tools for system modification. This means that each user may manually adapt the system to their taste or context, typically through the application settings. For example, a user may explicitly change the application’s background to a dark colour through the system settings.

Adaptive systems An adaptive system is able to change its characteristics based on the user’s needs, hence why it may also be called a self-adapting system. According to the same authors, an adaptive system stores and uses knowledge about the system, its interface, tasks and user. This way, the application must be able to match particular system responses with the usage profiles.

Furthermore, context can be categorised into five types, as represented in Figure 2.5: computing context, such as network status, connected resources (e.g. printers, input devices); user context, like the user’s location, social status or friends; physical context, which is associated with conditions like lighting, noise level, traffic conditions, precipitation; time context, for example, time of day, week, month, year and season of the year; context history, which registers and returns previously detected contexts [30].

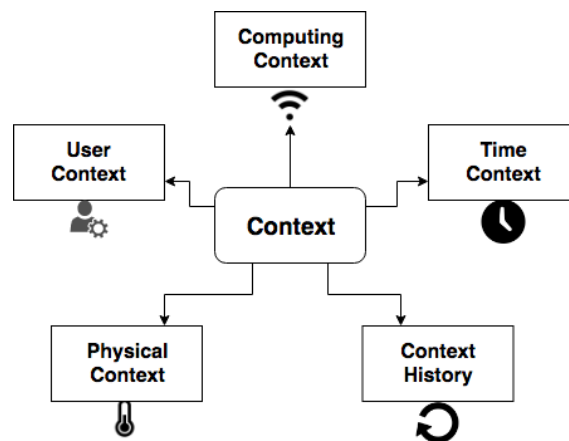


Figure 2.5: Five categories of Context.

Desktop computers typically have access to most of these types of context. However, the physical context detection is still lacking. This is because a regular, personal computer does not include sensors for this goal. Although it includes a temperature sensor for components like the processor and graphics card, it cannot correctly measure the ambient

temperature, which would be a relevant aspect for physical context. On the other hand, mobile phones include a wide variety of sensors. This means that they can measure, for example, ambient temperature, relative humidity, light level, etc. Hence, it is possible to detect more accurately the physical context. Due to these factors and also because it is easy for a mobile phone or tablet to detect the other four types of context, they are generally a better option for running context-aware applications. Thus, as the ability to detect context is greater, the potential of automatically adapting a mobile application according to the detected data also increases.

2.2.1.1 Adaptation in Mobile Applications

Ever since mobile phones became a reality, they have proved to be a relevant way of connecting people. It is easy to send information to almost anywhere in the world by calling or sending a text message. Furthermore, when mobile phones started to become smartphones, their ability to transmit information has greatly increased. With the introduction of smartphones and tablets, mobile applications soon came into existence. Nowadays, there is a smartphone application for almost everything. From city guides, to football live updates and even applications to help people exercise, smartphone applications are an important factor in everyone's lives.

Today's mobile phones are, simply put, miniaturised computers. However, there are some differences between a mobile phone and a computer that make smartphones an interesting platform for running software. Some of these differences are:

- A mobile phone is almost always on;
- Users carry their smartphones with them most of the time, making them a very personal gadget;
- Smartphones typically have access to a relevant amount of sensors.

For all these reasons, detection of context in mobile phones is facilitated. Thus, smartphones are a tempting platform for building context-aware applications. As mentioned earlier, in section 2.2, context-aware applications are effective at providing relevant information or services, based on a context data feed. An useful way to adapt an application to the context is by changing its User Interface (UI). An important User Interface principle is *consistency*. Consistency dictates that when users know that the same function will always have the same effect, they feel more confident in using the system [28, 30]. Hence, in a context-aware, adaptive application, consistency is attained when the system adapts itself systematically the same way taking context into account. Furthermore, users expect highly personalised information, adapted to their profiles and preferences, as well as their current situation (context) [22]. For example, an application can be made to change its appearance to a lighter theme when it is in a brighter environment to avoid screen reflection.

Context-aware applications (mobile applications included) can adapt themselves, or be adapted by the user, in many different ways. According to Höpken in [22], these adaptations can be put together into different categories:

Device Adaptation There are thousands of different types of mobile devices and different capabilities, such as display size. Mobile applications featuring device adaptation can change according to the device they are running on. For example, an application may hide some elements on smaller screen devices, in order not to occupy too much screen space.

Multimodality Multimodal systems are able to provide different types of interaction, depending on the user context. For example, if the user is hearing impaired, a multimodal application may use vibration instead of sound when interacting with him/her. Simply put, a multimodal system features different interaction interfaces (hand gestures, speech recognition, etc).

User Adaptation (Personalisation) Personalisation is responsible for adapting content, presentation and behaviour of an application according to the user profile and history. For example, an application implementing this kind of adaptation may present the artefacts of a museum in different ways, depending on age. While texts and images could be shown to an adult audience, entertaining videos would be shown to younger users [18].

Location Adaptation Mobile applications can typically access location-based services as today's devices have access to *Global Positioning System* (GPS) most of the time. A mobile application featuring location adaptation provides different information according to the user's current location. For example, the application may show the user's current position on a map, along with nearby points of interest.

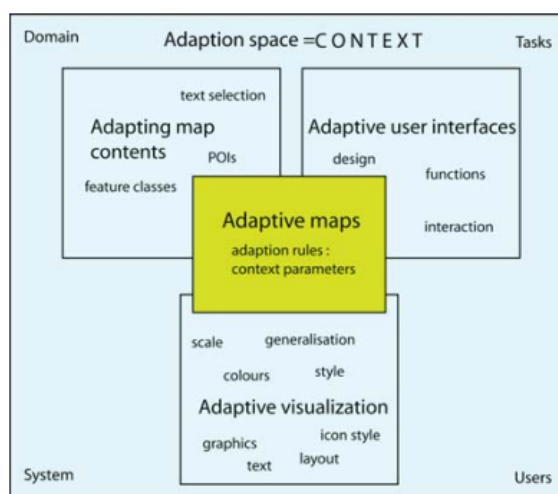


Figure 2.6: Adaptive Maps. Source: [30].

By combining GIS, context, mobile applications and adaptation, it is possible to create *Adaptive maps*. As represented in Figure 2.6, adaptive maps are the result of combining adapting map contents, adaptive user interfaces and adaptive visualisation. The most obvious map adaptation is location, but other contexts are also useful to mobile map usage, such as: orientation of the map/device, time, navigation history, social and cultural situation, physical surroundings, etc.

2.2.1.2 Adaptation in GIS

Adaptation in GIS is generally achieved by adjusting the visualisation of geographic information and associated parts in the visualisation process such as the interface, information content, etc. The reason for adaptation in GIS comes from the need for improving the usability of the geospatial information access, resource limitations and the desire to enhance the relevance of the presented information [29].

In order to be able to change the visualisation, the geographic information and/or interface must feature adaptability. As digital representations of geographic information separate the storage and the display components of the information, adaptation can be easily achieved. Paper maps store and display information, making adaptation impossible, as they are not adaptable and hence not susceptible to adaptation [29].

As mentioned in section 2.2, there are two types of flexible systems: adaptable and adaptive systems. In GI systems, this is also valid. Adaptability can be featured in the system, for example, by allowing the user to move the map around or by featuring zoom controls. Yet, adaptive GI systems typically involve more resources, like the user's context, hence being harder to implement. Furthermore, while most GI systems feature adaptability, there are fewer adaptive systems [29, 30]. One form of automatic adaptation in GIS (adaptive) is the automatic generalisation implemented by interactive maps. When we zoom in, the map window will contain a smaller area, but with more detail, and the map visualisation will show more relevant information for that area (for example the names of the stores in a street).

Mobile devices facilitate context detection (see section 2.2.1.1). Spatial awareness is one of the simplest and most evident implementations of adaptive GI systems. *Location-Based Systems* (LBS) implement this kind of adaptivity, where the information is adapted to a specific location, in most cases the user's current position. For example, automatically panning the map to the user's current position (obtained through the mobile device's GPS receiver) is a simple way of applying spatial awareness to the visualisation of geographic information [29].

According to Reichenbacher in [29] and Sarjakoski and Sarjakoski in [30], **adaptation objects** are the contents that can be changed by visualisation, either by the user or the system. On the other hand, an **adaptation target** is the referential source of information to which the adaptation is directed to, for example, the mobile device. Figure 2.7 summarises adaptation objects using the example of a mobile map. Although visualisation

is the central object of adaptation, the user interface and geospatial information components can also be treated as separate adaptation objects in adaptive visualisation. For example, by interacting with the map (user interface component), it is possible to trigger a visualisation change (visualisation component).

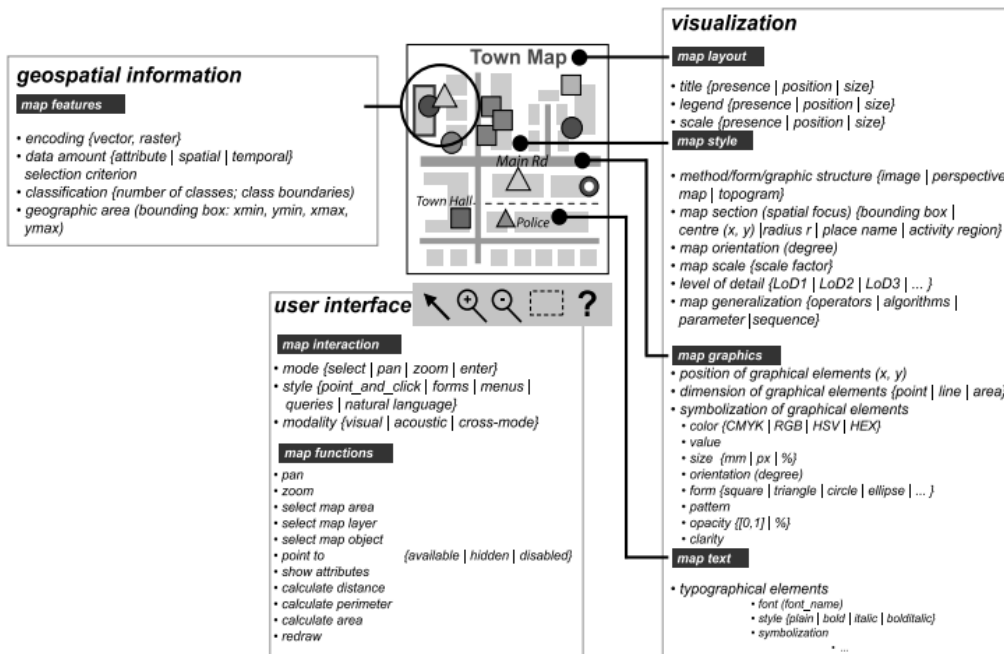


Figure 2.7: Adaptive Visualisation in GIS. Source: [29].

Although Internet users generally have had some contact with GI systems (like Google Maps³ or a GPS device), they may not necessarily have much knowledge of using and interpreting digital maps. Hence, the geospatial information must often be generalised and adapted to the specific usage situation, so that the user is not overloaded with information. The most fundamental issue is to select the relevant information to be shown (to avoid information overload). Thus, *selection* methods are typically applied for the information to be displayed. In other words, adaptation can be used to present only the relevant geospatial information, according to the current user's needs [30]:

- Selection of the topographic feature classes or layers to be shown on the map (roads, buildings, etc);
- Selection of the points of interest to be shown on top of the topographic data;
- Selection of relevant text on the current map (place names, road names, etc).

This selection of topographic feature classes depends on the database schema supported by the map service. The selection is affected by the usage situation of the user, for

³<https://maps.google.com>

example, a mobile user is by default interested only in updated map information relevant to the current task [30].

Map adaptation may also be controlled by context parameters, leading to a context-aware GIS. Information supplied to the context-aware Location-Based System often relates to the user's immediate vicinity, so the user is the most prominent context parameter for mobile usage situations. Besides the location, there are other context relevant attributes to facilitate mobile map usage: orientation of the map, time, navigation history, physical surroundings, etc [30].

2.3 Conclusion

This chapter presented important concepts for better understanding this dissertation. GIS are mainly a combination of the actions of storage, analysis and visualisation of geographic information in a single applied system to computational devices that contain maps as reference objects. Furthermore, being able to specify where an object is located is important when sharing information. Hence, georeferencing comes into play. By using coordinate systems, it is possible to pinpoint spatial locations effectively. Furthermore, it is also possible to convert the position of the object from one system to another. As Web technologies are widely accessible to Internet users, Web GIS is probably the most known form of GIS. Thus, this chapter introduces this technology, along with typical architectures.

As mentioned in section 2.2, context has the potential to increase a system's information relevancy and user experience. By taking into account the device's context, it is possible to better understand what information is more relevant under certain conditions. Thus, a system may take this opportunity to adapt itself according to its surroundings and outside conditions, so more relevant information is shown. Section 2.2.1 provides a detailed description on types of adapting systems. Furthermore, due to the amount of sensors typically present in mobile devices, adaptation according to context is even more tempting in these machines. Finally, as mentioned in section 2.2.1.2, GIS can also be adapted according to context. Geographic information becomes even more relevant in context-aware systems, potentially reducing the amount of noise (i.e. irrelevant information under the device's current conditions) shown on the screen.

Many systems take advantage of the concepts and technologies presented throughout this chapter. Next chapter presents some of these systems, which are relevant for this dissertation.

RELATED WORK

Automatically adapting applications contribute for a better user experience. This chapter presents and describes some projects related to this dissertation. It contains projects featuring context-awareness, adaptivity and/or Web GIS technologies.

This chapter starts by describing, in section 3.1, the concept of trigger/action rules developed for context-based applications, which is a fundamental aspect of this dissertation, as well as some applications that make use of them.

Section 3.2 describes two systems that make use of Web GIS technologies as well as adaptivity, to some extent.

Section 3.3 presents a platform for analysing coastal structures, which takes advantage of context, adaptation and georeferencing. Similarly, section 3.4 describes a system featuring automatic activity detection based on the user's spatial context and adapting to it. Lastly, section 3.5 presents an emergency team management application that also takes into account spatial context and georeferencing during runtime, however, as these systems are limited in terms of adaptations, as the latter are hard coded.

Finally, section 3.6 presents a summary of this chapter, where it is explained how all these works will contribute for the development of this dissertation.

3.1 Modelling Context with Trigger/Action Rules

Ghiani *et al* (2015) have developed several software solutions related to context [17–20]. Not only did these authors develop software capable of detecting context and communicating observations to other components, but they also developed practical applications using their solution. Their work is highly based on trigger/action rules which enable adaptation. An adaptation rule is composed by a trigger and an action. The application

executes the adaptation action if the user's context activates the condition (trigger) chosen when defining the rule. A simplified version of a trigger/action rule is, for example, "when the user starts moving (trigger), set font size of element "title_bar" to 16 (action)".

In [18], the authors present an authoring tool designed for supporting the development of context-dependent user interfaces by defining rules for the adaptation and distribution of the application. With this tool, it is possible for developers to easily integrate interface adaptations to context, which is structured along four main dimensions: the user (tasks, preferences, emotional state, etc), the device (connectivity, multimedia support, battery level, etc), the environment (noise, light, temperature, location, etc), and social relationships (friendships, groups, etc). In order to correctly execute the applications according to the adaptation rules specified, this tool features three components:

- The **context model manager** is composed of a context server and a set of modules delegated to monitor relevant parameters of the context of use (e.g. environmental noise, device coordinates, etc). This component's purpose is to detect context alterations and inform the relevant modules, so further actions can be carried on by them;
- The **distribution manager**, which is responsible for managing user interfaces across multiple devices;
- The **adaptation engine**, which stores and manages the adaptation rules.

Figure 3.1a shows how these components interact with each other in run-time. The adaptation engine subscribes to the context model manager in order to be informed of the occurrence of the events which are relevant for the rules associated with the application. When one or more of such events occur, the adaptation engine sends the actions to the Web application in order to perform the corresponding adaptation.

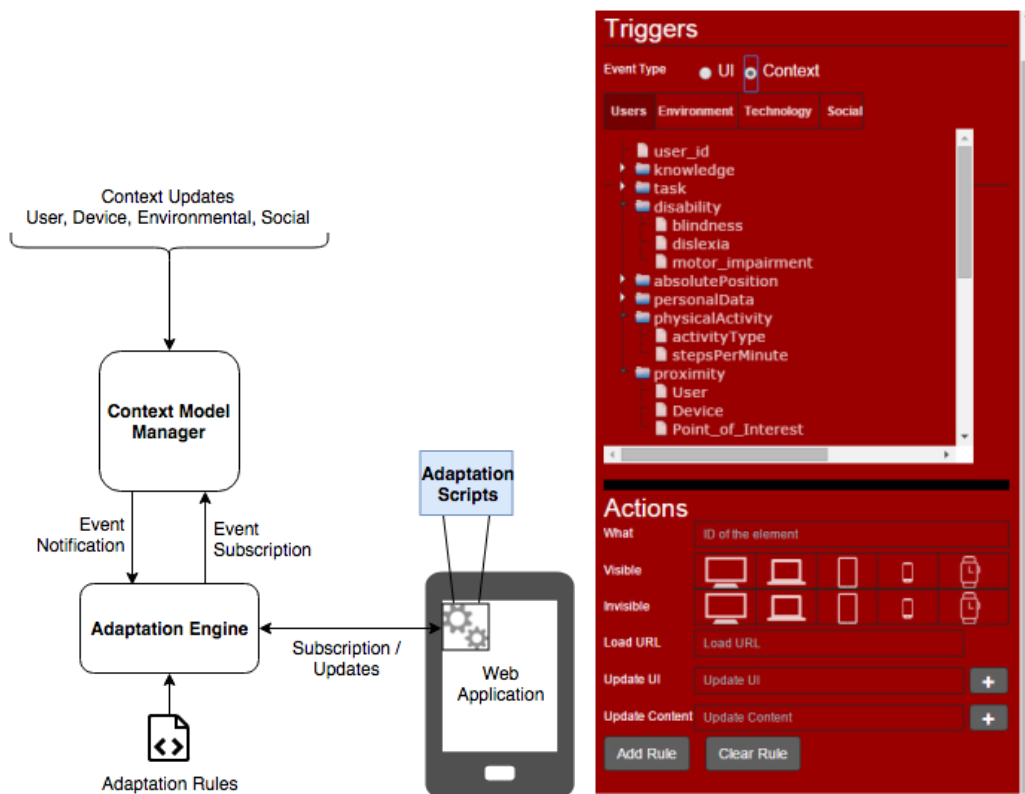
By using this tool, developers can easily create context-aware software through the creation of trigger/action rules. The developer defines the adaptation trigger by firstly selecting an attribute from the contextual aspects tree. Afterwards, s/he needs to specify the change (e.g. "font-weight: bold") that is going to be applied to the chosen element (see Figure 3.1b).

Furthermore, in [26], Lucci *et al* study smartphone environments that allow non-professional developers to create context-dependent applications. Their study focuses on three Android apps: Tasker¹, Locale² and Atooma.

These applications allow the end user to personalise many aspects of their smartphones. By creating certain conditions and actions (similar to what is described earlier in this section), end users can customise their smartphone's behaviour. These applications are not only able to manage a large quantity of elements but they can also express a

¹<https://play.google.com/store/apps/details?id=net.dinglish.android.taskerm>

²<https://www.twofortyfouram.com/>



a Run-time support architecture of the mechanism used for automatic adaptation rule activation.

b Trigger/Action rule definition through a simple Web interface containing multiple context elements. Source: [18]

Figure 3.1: Authoring tool's architecture.

significant amount of action types. Figure 3.2 shows which application supports each element by adding their initials (A for Atooma, L for Locale and T for Tasker).

Furthermore, with these applications, end users can create profiles in Tasker, condition/actions in Atooma or situations in Locale, which are similar to the concept of *rule* mentioned earlier in this section. Consequently, it is possible to define rules like “If battery is below 20% then decrease the display’s brightness”.

In [1], Alnanih *et al* propose a context-based and rule-based solution to change the application interface. The application changes the user interface based on predefined context conditions, which consist of location and time (e.g. patient’s room during the evening shift), ambient conditions (light level) and noise level. When one of these conditions changes, the application automatically checks what actions should be taken according to a *Decision Table*. Decision tables provide a schematic view of the inference process in decision making. They provide a compact visual presentation, contributing to a better understanding of the selection problem.

As shown in Table 3.1, decision tables include conditions, rules and actions. When one aspect of the detected context changes, the application evaluates the Boolean value for each condition. Consequently, all the actions under the rule that represents the Boolean

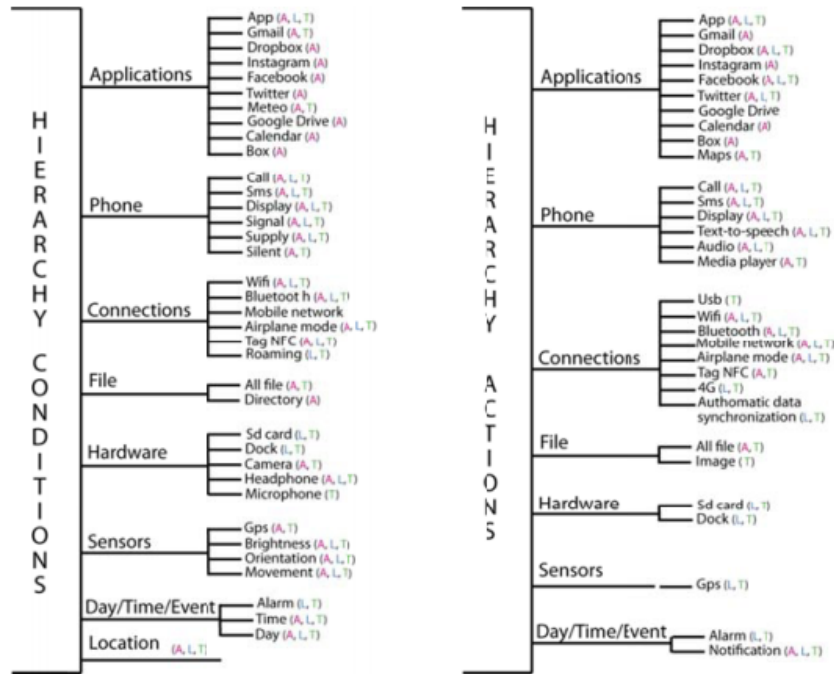


Figure 3.2: List of all elements managed by the three environments collectively. Source: [26].

Table 3.1: Decision Table Example.

Conditions		Rules			
		1	2	3	4
C1	Doctor is in the patient's room during the morning shift	-	-	-	-
C2	Level of light in the room is bright	Y	Y	N	N
C3	Level of noise in the room is low	Y	N	Y	N
Actions					
A1	Adjust information's font size to "user default"	X	X		
A2	Adjust information's font size to "large"			X	X
A3	Adjust brightness to "user default"	X	X		
A4	Adjust brightness to "high"			X	X
A5	Receive alerts by "ring tone"	X	X		
A6	Receive alerts by "vibration"			X	X
A7	Receive information via a headset	X	X		

combination of the conditions are taken. For example, according to the decision table in Table 3.1, when the "Level of light in the room is bright"(C2), but C3 is false (level of noise in the room is not low), actions A1, A3 and A5 are executed.

This type of application allows the use of the device's sensory capabilities as sources of information related to the context of the user. Applications and interfaces can be adapted according to changes in context. These context models can be extended taking into account objectives of adaptation to the user's spatial context. We do not know of works where this is done in a systematic way, integrated in a context model. However, we do know of Web GIS applications where the explicit knowledge of spatial context conditions influence the interface and the available functionalities in the applications.

3.2 Web GIS Examples

In this section several works related to Web GI systems will be presented. Although there are many works including geographic information system automatic adaptations, the following projects were chosen as they were developed in NOVA LINCS' Multimodal Systems group, similarly to this dissertation.

In [15], Ferreira *et al* present a GIS-T³ solution called XTraN App, whose goal is to provide real-time accurate data about buses, stops and routes to their users. For example, a user wanting to check when his bus is arriving does not need to consult the extensive list of all buses provided by the operator. Among many functionalities featured in this work, some of the most important are:

- Static consultation of timetables and estimated times of arrival of desired buses;
- Dynamic contextualised consultation. The user is able to know at any time which bus stops are closer to his current position, the respective routes and estimated times of arrival;
- Personalised information. The application only shows relevant information to the user based on the current context;
- Personalised bus stop status. The app provides notifications in the morning with the current status for the user's morning commute bus stop and route.

Furthermore, the user is also able to check his current position on the map, as well as knowing nearby bus stops, as shown in Figure 3.3.

This work uses the OpenStreetMap library⁴ for Android instead of Google Maps (which is included in Android's libraries) because OSM⁵ provides complete overlay customisation for drawing objects on the map.

³Geographic Information System for Transportation

⁴<https://github.com/osmdroid/osmdroid>

⁵OpenStreetMap

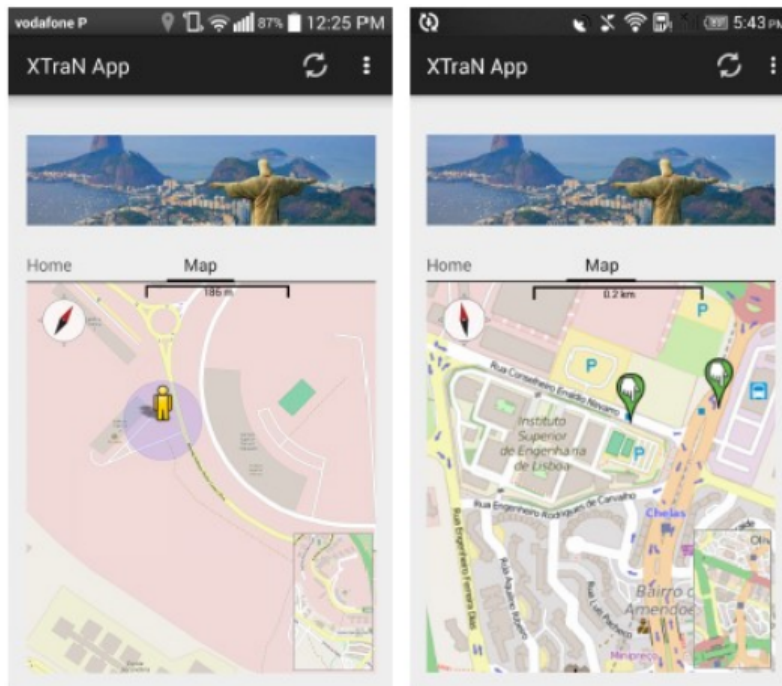


Figure 3.3: User's current position and nearby bus stops, respectively. Source: [15].

Di Martino *et al* propose an approach to automatically generate Web GIS applications able to recommend content to users and adapt the interface of a web-based spatial map in [11]. The approach is based on an extension of WebML⁶, which is a modeling language suited to support users in designing data-intensive Web applications by providing them with a set of visual notions to model the content, data and structure of a Web application. In this solution, adaptivity is achieved implicitly through the addition of modules to record user interactions with the map, analysing the data and producing recommendations. For example, by analysing which regions the user tends to spend more time looking at, the application can automatically create a custom filter that focuses those regions by default when the user accesses the map page.

The following platforms provide further examples of automatic adaptations in GIS-related systems. As previously mentioned, these platforms were chosen because they were developed in NOVA LINCS' Multimodal Systems group, similarly to this dissertation.

3.3 Platform for Coastal Structure analysis

The information described in this section was taken from [27], by Maia. This work presents a platform for coastal structure analysis, available in the form of a Web GIS. These structures (breakwaters and jetties) are subject to intense erosion caused by waves and tides. Therefore, this application aims to provide a simple way of identifying points of the structure that must be carefully analysed and associate information, including

⁶Web Modeling Language

media, to them. Hence, it becomes easier for the responsible organisations to detect what structures need maintenance.

Georeferencing is done by marking each part of a coastal structure by drawing a polygon around it, for easy identification. Furthermore, each important observation point is represented by a yellow marker. When an area is clicked, the user is able to see the name of the structure part, as well as the information associated with it, including media content.

Additionally, the system also features a red marker on the map. This marker represents the user's current location, which is captured by the *Geolocation* API present in HTML5, with intervals of 10 seconds. Furthermore, the application detects the structure part where the device is located, which is useful when adding information, as it is automatically associated to the correct structure part. This feature takes into account the context to provide a better experience for the user, making this a context-aware application.

As mentioned earlier, when analysing the structure at particular observation points, users have the option to upload pictures depicting the current state of the coastal structure. However, the application stores not only this media content, but it can also store some metadata, like altitude, orientation, latitude, longitude and the photo capture date. Furthermore, this information is then represented in the Web App. Additionally, a red line represents the photo capture orientation with a blue triangle drawn around it, to represent what is visible in the picture.

3.4 Automatic Activity Detection

This section describes information present in [9], by Antunes. The system developed and described in this work aims to automatically detect regular patterns so it is possible to identify the various agronomic activities that a technician performs. By taking into account the technician's spatial context captured by GPS, the application determines and records their activities for the day.

When starting the application, an interactive map is displayed and showing the technician's current location. As mentioned before, it periodically reads the technician's location through the device's GPS sensor. Thus, with this information, the application checks which conditions belonging to a predefined set hold true - if the user is not moving, if the user is driving or if the user is walking. With these conditions, the system is able to extract which activities are being performed at a particular moment in time.

When the technician is still for a few seconds in the same location, that location is identified as a staypoint by the application. These staypoints can then be used to identify relevant parcels (for example, identifying a field where tomato was planted). However, if the technician stays even more time (around 5 minutes) in a 25 meter area, the location is instead classified as a point of interest. Additionally, when he is walking, a blue line is drawn on the map, showing the technician's path and when he is driving, the same thing is done, but using a yellow line. Finally, sequences of activities are automatically

identified based on a predefined set of common activities that the technician may execute. All these activities are recorded in a database and may be checked by the technician in the application.

With this work, agricultural technicians have access to a tool that provides a visual confirmation of their completed tasks as well as facilitating their registration, as this information may be important to the employer.

3.5 LiveTeams - Emergency Teams Management

The information described in this section was taken from [21], by Bizarra. This work presents a Web application for emergency teams management. In a case of emergency, information about the occurrence such as blocked roads and victim numbers is usually scarce. Furthermore, the communication between the command post and field operational staff is not always fast or accurate, since it is generally done through phone calls or walkie-talkies.

Thus, this application allows users (field operational staff, command post officers, etc) to register themselves. Furthermore, teams can be created, which may include multiple members. Hence, this facilitates the leader's team management, as he is able to efficiently check the team's details, such as the members and their respective last known locations. As this system features an interactive map, users are able to perform many actions effectively. Among them: checking each member's last known location; checking each team's most recent location; defining areas (may be useful for marking flooded areas); defining polygonal lines which represent paths (may be useful for representing the most efficient path to a point of interest, taking into account road blocks).

Furthermore, this application is responsive, as it adapts to different screen sizes, including mobile devices and desktops. Moreover, the interactive map is automatically updated when new information is added (such as a team's current location), without needing to refresh the page. Finally, each user has a role in a team, to which the application adapts itself for allowing or disallowing certain actions such as editing the team's details.

By using this application, emergency teams are not only able to respond to emergencies more efficiently (they know its exact location, instead of relying on verbal communication) but also to achieve a better overall emergency resource management, as a bird's eye view is available through the map, showing every team's last location.

3.6 Summary

In the beginning of this chapter, section 3.1 presents the concept of trigger/action rules and some of its practical applications while modelling context in mobile applications. These rules are very useful in context-aware applications, as they are a simple yet effective way of creating adaptations to the system independently of the structure of the

application. Although the presented systems are adaptive and feature a mechanism to do so, the implemented adaptations do not take into account, as a feature, the potential of identifying geographic context, only to the application's interface.

Section 3.2 presents two works featuring Web GISs. Although the project developed by Ferreira *et al* [15] is directed towards public transportation, it features some important functionalities which could be improved, through adaptation, given the availability of information on geographic context., such as revealing nearby points of interest according to the user's position (e.g. bus stops) and notifying the user with updates featuring useful information to him (e.g. the morning commute's bus). On the other hand, Di Martino *et al* [11] demonstrate an automatically generated Web GIS application. This system is able to produce recommendations based on the user's interaction with the map. Although it features an adaptive Web GIS, the application is only able to adapt itself according to an history of interactions and not the user's current context, which is a significant source of information, to be used in this dissertation for adapting the system.

Maia's dissertation on coastal structures [27] presents a system that explicitly takes into account the geographic context of the user, although not taking advantage of independently representing geographic context. His work captures the user's context to improve the software's usability. Furthermore, it detects the structure part where the user is located, enabling the association of the collected data with the relevant database object. This system detects the user's location, altitude and orientation, but context detection is very limited as it was developed as a Web application, which has access to a small amount of sensors. Additionally, [21] also presents a Web application featuring adaptation based on the geographic context. It is a responsive application that takes into account screen size and team location, automatically adapting to them. However, being a Web application, it has access to a small amount of sensors, like in Maia's work. Finally, [9] is a mobile application that is the most effective of the three at capturing context. By detecting the user's location, movements and points of interest, this application is able to adapt its behaviour more adequately. However, the adaptation rules present in these three solutions are explicitly present in the system's code, not as part of a context model. Furthermore, these applications also feature georeferencing of important points and areas on the map, by using simple markers and 2D areas, respectively, which is similar to what is used in this dissertation. Basically, these three works can be viewed as starting points for the dissertation, as they feature a Web GIS with georeferencing, some degree of context detection and adaptation.

A comparison as been made between works [9, 21, 27] and is shown in Figure 3.4.

All three projects access geographic context during their runtime. However, none of them make use of non-geographic context (such as the room temperature or battery level). Furthermore, the term "adaptation" has been divided into three sub-types: interface adaptation, which refers to changes made to the system's user interface; functional adaptation, which refers to changes in the system's functionality; GIS adaptation, referring to changes to the map itself.











	Geographic Context Access	Non-Geographic Context Access	Interface Adaptation	Functional Adaptation	GIS Adaptation	Extensible Adaptation
Coastal Structure Platform		X				X
LiveTeams		X		X	X	X
Automatic Activity Detection		X				X

Figure 3.4: Comparison between the mentioned projects' ([9, 21, 27]) adaptation features.

Maia's work ([27]) supports all three kinds of adaptation - it adapts itself to different screen sizes and different devices; changes its behaviour according to certain conditions (such as automatically submitting information to the breakwater the user is standing on); and performs adaptations to the map automatically (representing the user's location), while Bizarra's ([21]) only supports interface adaptations - it simply adapts its user interface according to different screen sizes and devices. Finally, Antunes' work ([9]) is only available on Android devices, thus getting a yellow mark on interface adaptation. However, it performs functional adaptations (automatically detecting when a user is defining a new area) and map adaptations (automatically creating new polygons and markers).

Finally, one of these works support extensible adaptation. In other words, if new automatic adaptations were to be added, they would need to be manually coded. This is due to the fact that their works only include hard coded adaptations.

CONTEXT ADAPTATION MODEL

As mentioned throughout this dissertation, the developed system aims to provide a generic solution to automatic adaptations. As will be discussed later in Chapter 5, this is achieved through the implementation of an online configuration tool. However, this generalisation still needs to follow a certain well-defined pattern.

In this dissertation context and automatic adaptation are modelled through *adaptation rules*. The context adaptation model (based on early work by Paternò and others [18]) is supported by *rules* containing both a *trigger* and an *action*. Furthermore, these *triggers* and *actions* comprise *entities*, *entity attributes* and *values*.

Through the definition of each of these concepts, it becomes possible to build full conditional statements to use as triggers for rules. As an example of this, if we wanted to build a conditional statement specifying “When the ambient temperature is over 15 degrees”, it could be done as follows:

- **Entity:** Ambient
- **Attribute:** Temperature
- **Comparison Operator:** Greater Than (>)
- **Value:** 15

The *entities* can be described as a general context group containing multiple *attributes*. As an example, the User himself is an *entity* with *attributes* such as its location, its movement speed, its proximity to a certain point, etc, and the Map is also an *entity* with *attributes* like the zoom level, bearing, level of detail, etc. Finally, in each conditional statement the *comparison operator* can be “equal to”, “not equal to”, “greater than”, “greater than or equal”, “less than” or “less than or equal”.

Furthermore, the system may include as many *rules* as the knowledgeable administrators see fit and each rule is responsible for adapting a specific aspect of an external application, under certain conditions as specified in the *rule's trigger*. *Triggers* are essentially what was previously described as “conditional statements” - conditions comprising an entity, an entity attribute, a comparison operator and a value. All the *rules' triggers* are periodically verified and, when a *rule's* condition is evaluated as being *true*, the respective *action* should be automatically activated. *Actions* have a structure similar to the *triggers'*. However, as *actions* are basically assignments, rather than conditional statements, they do not comprise a conditional operator. Still, *actions* do include an entity, an entity attribute and a value which will determine the entity which is targeted in the adaptation, the attribute to be changed and the value to be applied to the referred attribute. Figure 4.1 shows a visual representation of a rule example. This rule would centre the map on the user whenever (s)he is moving with a speed of 50 km/h or more. Summarising, *triggers* read attribute entity values, while *actions* update them.

		Entity	Entity Attribute	Comparison Operator	Value
Rule	Trigger	User	Speed	\geq	50 km/h
	Action	Map	Center	N/A	User

Figure 4.1: Visual representation of an adaptation rule example.

As the context model is based on the verification of administrator-defined conditions and the activation of a corresponding action, it was important to define a way through which knowledgeable administrators could, in fact, create these conditional statements and associated actions. Generally speaking, conditional statements include at least two operands (left and right), and an operator (equals, less than, etc). In this dissertation's context model, a *comparison operator* is the same as this example's operator, while a *value* corresponds to the right operand. However, the left operand actually comprises two concepts - *entity* and *attribute*. Although a single denomination would suffice, *entities* and their corresponding *attributes* were added for a better organisation and a more efficient behaviour by the external application (as it will be mentioned later in section 5.4).

The context adaptation model was conceived to enable the design of adaptation concerning conditions that involved geographic context. This involved establishing the state in relevant situational entities such as the user (addressing context evolution geographic context of the user's device) or the geographic space (represented by its representation,

the map). This led to an extension of the information contained by its conditions described by Ghiani *et al* [18] in their context model. This work presents a context model based on specific, previously defined rules. The authors were not concerned with modelling and adapting geographic context. This dissertation extends that model in the sense that it is also based in adaptation rules comprising a trigger and an action, however, it aims to be more generic and applicable to GIS applications.

The extended model (represented as an entity model diagram in Figure 4.2) was realised in this dissertation through the implementation of an online configuration platform, where the concepts of this context-based adaptation model are applied. This platform and other components are further discussed in Chapter 5.

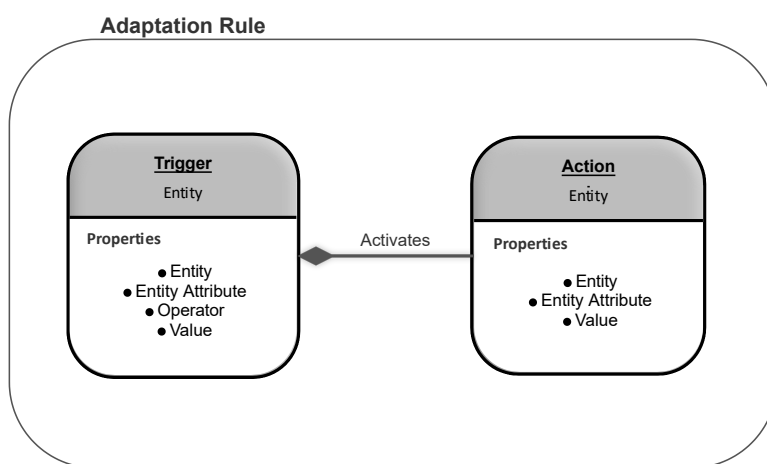


Figure 4.2: Adaptation Rule Entity Model.

IMPLEMENTATION

This chapter presents the three components which were developed in order to achieve the goals of this dissertation and also the technologies which were used during the development stage. It is subdivided into three main sections, comprising each of the developed components, starting with the simplest - Points of Interest Server (5.2), then the Configuration Platform (5.3) and finally the Android application (5.4).

5.1 System Modelling

This section starts by describing the technologies used in both the servers and Android application throughout the development of this dissertation. Then, section 5.1.2 presents the system architecture, which describes how these technologies interact with each other.

5.1.1 Technologies

This section presents both the front-end and back-end technologies which were used throughout the development of this solution. Although some of these technologies are proprietary, open source software¹ was preferred while developing this dissertation.

5.1.1.1 Android application

This subsection describes the technologies which are used by the Android application in order to behave properly.

OpenStreetMap *OpenStreetMap*² (OSM) is a crowd-sourced mapping project created in 2004, whose goal is to provide a world map, accessible by everyone. The geographical

¹<https://opensource.com/resources/what-open-source>

²<https://www.openstreetmap.org/>

data is collected from several sources, including aerial images and GPS devices. This information is available to everyone, and anyone can become a contributor for this project. Unlike most proprietary systems, OSM can be used offline.

osmdroid In this dissertation, OpenStreetMap is available through an Android library called *osmdroid*³, allowing developers to use OpenStreetMap as a built-in view, instead of Google's MapView⁴. Finally, due to recent changes in Google's APIs pricing⁵, OpenStreetMap and osmdroid are even more attractive as they provide all their services for free. *osmdroid* was used in this dissertation to present the map, points of interest and all other map-related interface elements.

Activity Recognition API The Activity Recognition API⁶ is a technology developed by Google which can be used to understand what users are doing in the physical world (i.e. walking, running, driving, etc). This API periodically reads minimal sensor data and processes it using machine learning models. Thus, this information can be used to perform different actions according to the detected activity type. Furthermore, each detected activity includes a confidence grade which may be useful to further improve the application's behaviour. This API is used in this dissertation whenever there is a rule whose trigger monitors the user's movement type (e.g. walking, in vehicle, standing still, etc).

5.1.1.2 Servers

This subsection describes the technologies used by the system's servers in order to make this solution behave properly.

Node.js *Node.js*⁷ is a free, open-source, cross-platform runtime environment for server-side programming that executes JavaScript⁸ code. This technology was initially released in 2009 and allows developers to create all kinds of server-side tools and applications. Furthermore, it contains a package manager, *NPM* (Node Package Manager), which is a rather simple way of installing libraries.

Express.js *Express.js*⁹ is one of the packages available through NPM. Initially released in 2010, this Express.js is a web application framework for Node.js, widely used for developing websites, web apps and APIs much easier. Hence, this technology, along with Node.js, are used by the solution's servers, making them available online. This technology

³<https://github.com/osmdroid/osmdroid>

⁴<https://developers.google.com/android/reference/com/google/android/gms/maps/MapView>

⁵<https://cloud.google.com/maps-platform/user-guide/pricing-changes/>

⁶<https://developers.google.com/location-context/activity-recognition/>

⁷<https://nodejs.org/en/>

⁸<https://www.javascript.com>

⁹<https://expressjs.com/>

was used along with Node.js in both the point of interest and configuration platform's servers, which made it easier to expose their APIs and interfaces to the outside world.

Angular 6 *Angular*¹⁰ is an open-source, *client-side framework* developed by Google. The latest version, *Angular 6.0.0*, was released in May 2018. *Angular* is a framework providing a simple way of associating *views* (HTML) to TypeScript¹¹ *objects*, ideal for creating single-page applications. The library provides a number of features that make it trivial to implement concepts such as data binding, routing, and animations. In this dissertation, *Angular* was used for programming the configuration platform's client-side logic and functionalities.

HTML *HyperText Markup Language* (HTML) is a markup language used for creating web pages and web applications. This language is composed by tags which tell the web browser how to format the document that is being shown. Finally, HTML was used for building the general aspect and structure of the configuration platform's web interface.

Pure *Pure*¹² is an open-source set of small and responsive CSS modules which may be used in every web project. This library is known due to its minuscule size (3.8KB). When using *Pure*, creating a web page or application suitable for both mobile and computer users is effortless. It contains responsive modules such as *Grid* and *Menus*, which automatically adapt themselves according to the screen size. *Pure* was responsible for the configuration platform's responsive design as well as the general styling of this web application.

MongoDB *MongoDB*¹³ is a free and open-source document-oriented database (NoSQL). *MongoDB* stores data in JSON-like documents and the document model maps itself to the objects in the application code. Furthermore, the database itself may be queried using JavaScript. *MongoDB* was used for storing all the rules, entities, initial settings and the point database URL, pertaining to the configuration platform.

PostgreSQL *PostgreSQL*¹⁴ is a very popular free and open-source object-relational database management system (ORDBMS). This system is fully ACID (Atomicity Consistency Isolation Durability) compliant and includes some native programming interfaces for languages like Java, Ruby, Python, Perl, among others. Additionally, *PostgreSQL* supports a wide range of data types including primitives (Integer, Numeric, String nad Boolean), document (JSON/JSONB, XML, etc), geometry and others.

¹⁰<https://angular.io/>

¹¹<https://www.typescriptlang.org/>

¹²<https://purecss.io/>

¹³<https://www.mongodb.com/>

¹⁴<https://www.postgresql.org/>

PostGIS *PostGIS*¹⁵ is an open-source software program that adds support for geographic objects to the PostgreSQL object-relational database. Furthermore, *PostGIS* adds support for geographic objects allowing location queries to be run in SQL. For example, it is possible to query the database for entries that are within a certain distance from a given point. *PostGIS* and *PostgreSQL* were used by the point of interest server, for storing and returning the POIs, when queried.

Docker *Docker*¹⁶ is an open-source program used for operating-system-level virtualisation, known as containerisation. Containers are isolated from each other (although they can still communicate with each other through well-defined channels) and they run their own applications. Even though the container concept is similar to the virtual machine's, containers run within a single Linux instance which in turn makes them more lightweight. Containers are created from "images" that specify their precise contents, including what OS to run on, what commands to execute, what network ports to expose, what programs to run, etc. *Docker* is used in this dissertation for easily launching both the servers and both the databases with a single, simple command, avoiding configuration and compatibility problems and other complications.

5.1.2 Architecture

The overall architecture of the developed solution is depicted in Figure 5.1. It details how the previously mentioned technologies interact with each other, and was divided into four layers - *presentation*, *logic*, *data* and *services*, described as follows:

The *presentation layer* is responsible for showing content and accepting interactions from the end user. As the system consists of three platforms (configuration platform, point of interest server and Android application), and only two of them have user interfaces (the POI server does not have one), the *presentation layer* is divided into two parts. The configuration platform consists of HTML5, PureCSS and CSS3. HTML5 was used to create the general template of each of the configuration platform's pages. As mentioned previously, PureCSS was used for the general style of the platform, and responsive design. However, some custom styling was also developed using CSS3. As for the Android application's *presentation layer*, osmdroid, built on top of OpenStreetMap, was used for presenting an interactive map containing all POIs. The end user is then able to interact with the internal systems through this layer. The *presentation layer* then sends the user input to the logic layer and waits for a response containing data which will dictate what should be displayed to the user as a consequence of their interaction.

The *logic layer* is divided into two main components - server and client. The server component is composed by both the POI and configuration platform servers. These servers both comprise Express.js, a web application framework for Node.js, which is built

¹⁵<https://postgis.net/>

¹⁶<https://www.docker.com/>

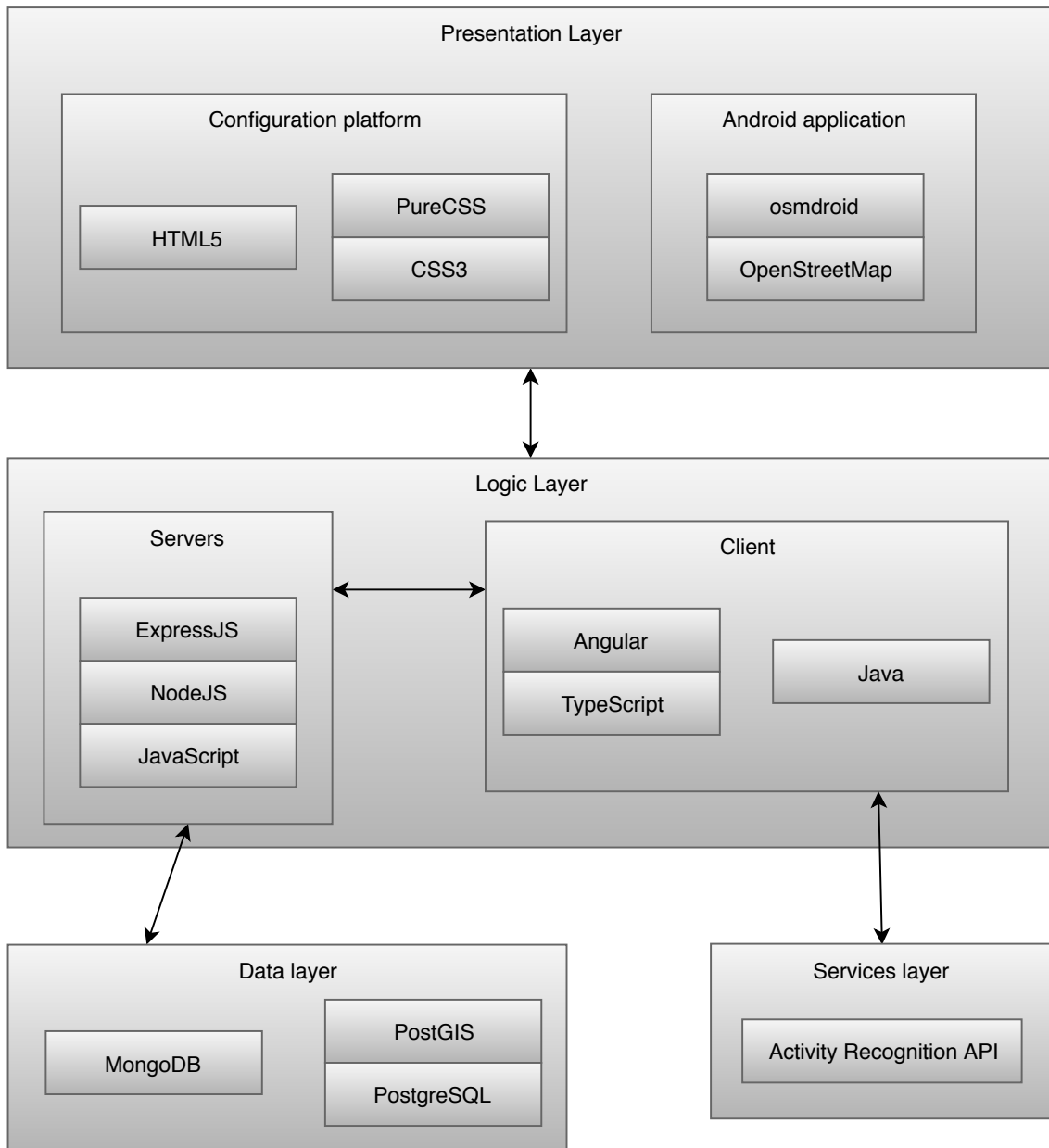


Figure 5.1: System architecture

on top of JavaScript. They are not only responsible for interacting with the data layer and retrieving all the necessary information for supporting the system (rules, settings, POIs, entities, etc), but also for exposing this data to the outside world in the form of a REST API. This *logic layer's* client component comprises Angular/TypeScript, osmdroid and Java. Angular is a single-page application framework that uses TypeScript for defining its behaviour. Angular is used in the configuration platform and is responsible for interacting with the REST APIs defined by the server component and sending all the information to the presentation layer to be rendered. Similarly, Java is used in the Android application for the same purposes: communicating with the data layer through the server component and telling the presentation layer (more specifically, osmdroid) what must be rendered.

The *data layer* comprises two databases: the MongoDB database stores all the created rules and entities and also the initial application settings and the POI database API URL, which is all managed by a responsible entity (e.g. an administrator or technician). The PostGIS database stores all the POIs that are used in the configuration platform for creating specific POI rules and the Android application for showing them on the map interface. Both database connections are established by a Node.js driver, the former being the official driver¹⁷ and the latter being a PostgreSQL driver¹⁸.

The *services layer* consists of a single external service, used by the logic layer's Android application. This service is the ActivityRecognitionAPI, developed by Google and it is used to detect with a relatively good precision the kind of movement which the user is performing. The logic layer then uses the results from this service to activate certain actions, if they are specified in the list of rules.

Finally, the technologies presented throughout this section were chosen as a result of evaluating a balance between open source and proprietary technology advantages. Open source technologies were preferred, however, sometimes there they are not (as) viable. For example, the ActivityRecognitionAPI is proprietary due to its ease of use, integration and lack of open source frameworks meeting its measurement's precision. Yet, all the other technologies are in fact open source, which is also advantageous, due to their vast community of contributors willing to help solving problems. If desired, it is possible to make small changes to these technologies, fitting the system's needs.

5.2 Points of Interest Server

Since this system is aimed toward a self-adapting tourism application, it is crucial to store points of interest. Hence, a special POI server was created, using Node.js and Express.js, as previously mentioned. Considering the fact that this server has a rather straightforward goal, its implementation is also simple. It features a PostGIS database containing a single table, called "pois". This table comprises four fields - id, name, description and geom. Their meanings and types are as follows:

¹⁷<https://www.npmjs.com/package/mongodb>

¹⁸<https://www.npmjs.com/package/pg>

- **id** - This field stores an automatically-generated, incrementing, unique, positive integer number and is used as the primary key for each of the POIs.
- **name** - As its name suggests, this field stores the POI's name and is stored as string type. It is intended to store values like "Eiffel Tower", "Torre de Belém", etc.
- **description** - The description field is used so it can hold useful information about each POI, such as the history behind it, the year it was built, etc. As it will be exposed later, this information is then available in the Android application through a InfoWindow.
- **geom** - The geom field holds a geometric value created through PostGIS¹⁹. In this dissertation, it holds the latitude and longitude values in a way that can be used to compute distances to other points, which may be used for future work.

For the purposes of this dissertation, this component is only meant to be accessed by administrators, as they are responsible for managing all the system's POIs. Due to this reason, this component does not feature a graphic interface, it can only be accessed in one of two ways: either through the PostgreSQL interactive terminal or through an HTTP RESTful API. The first method is recommended if the administrator desires to create more tables or add new fields, etc. However, when inserting, querying or deleting POIs, the latter method is recommended. Thus, this API has three endpoints, one for each of these functions, and are defined as shown in Table 5.1.

Table 5.1: RESTful API description.

Method	URI	Description	Payload Content
GET	/api/pois	Returns all the POIs in JSON format.	N/A
POST	/api/pois	Inserts a new POI.	name, description, latitude, longitude
DELETE	/api/pois/{id}	Deletes the POI with id = {id}, if it exists.	N/A

Although there are no available endpoints for updating POIs and getting the information of only a specific one, these features could be easily implemented in the server. Furthermore, even though the latitude and longitude could be included in the "pois" table as separate fields (and not take advantage of the geometric type), we decided that PostGIS would be more suitable, as it features a better extensibility potential, for distance-based queries, etc, which could be especially useful in the Configuration Platform.

5.3 Configuration Platform

The configuration platform is most likely the central component of this dissertation. It is here that entities and rules are defined, effectively enabling adaptation of the behaviour

¹⁹https://postgis.net/docs/ST_GeometryType.html

of the Android application. For better understanding this component, the following description was divided into six subsections, each of them addressing an important part of the platform. Finally, their order is important, as it introduces important concepts gradually, in order to better understand this component.

5.3.1 Entities, Attributes, Values, Comparison Operators

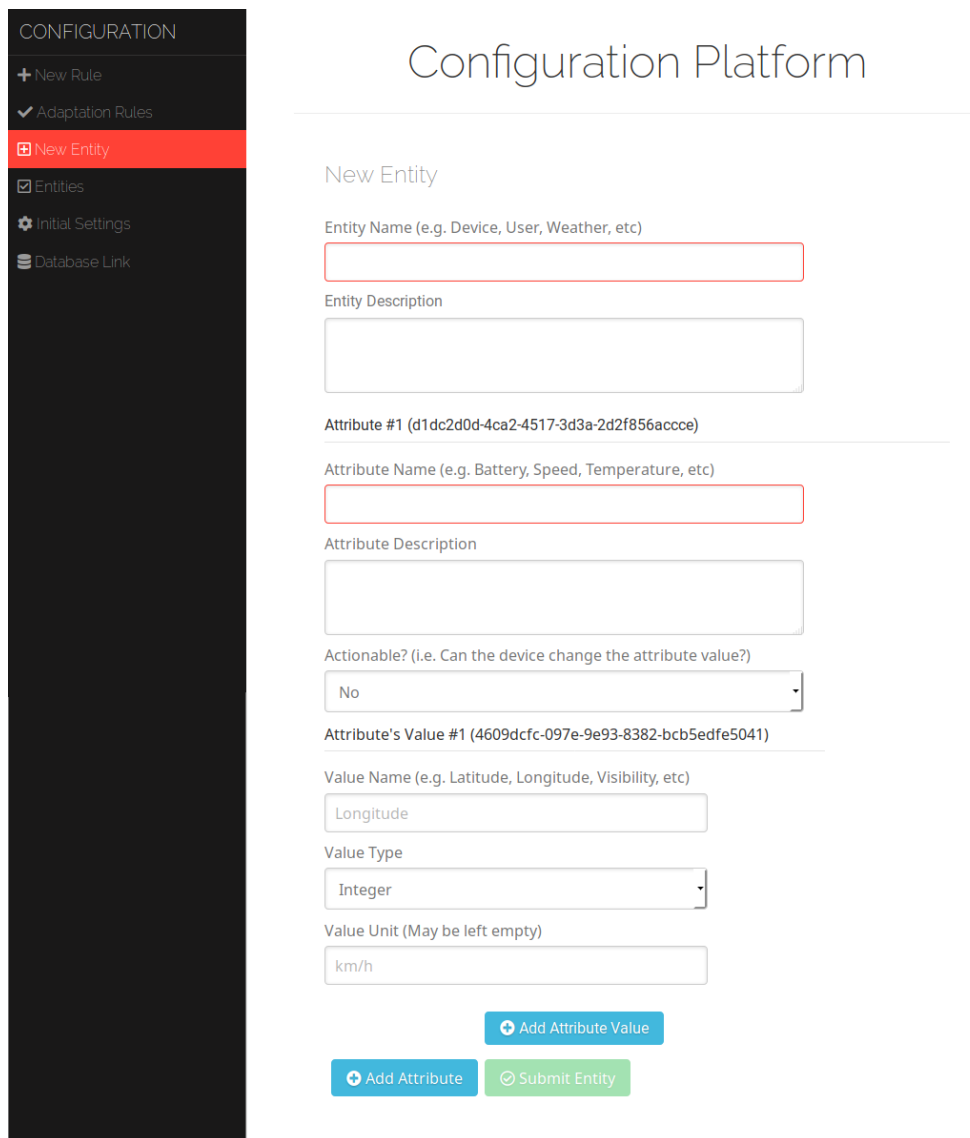
As this dissertation aims to provide a highly generic solution, knowledgeable administrators are able to create entities and respective attributes through one of the platform's pages, as seen in Figure 5.2. This page includes fields for the *entity's* name, its description (which is optional), and for each of the entity's attributes: the *attribute's* name, description (also optional) and a boolean field for defining if the attribute is "actionable" (see section 5.3.2). Furthermore, the administrator can define the type of each attribute (i.e. integer, string, boolean, etc), the units used (km/h, °C, etc) and, finally, the field name which may be useful when one attribute is described by multiple values. In this dissertation an *attribute* may actually contain multiple fields. Although it is rarely used, it may be important in cases where the *attribute* is "Location", in which case there will be a field for the latitude and another for the longitude (the "Value Name" text box shown in the Figure is used to indicate what the value refers to (e.g. latitude or longitude)). Hence, there is a button which will add another field for a *value* for the corresponding attribute.

Finally, there is a button which allows the user to add new *attributes* to the current *entity*. This page's form will only be ready for submission when all mandatory fields (*entity's* name and all *attributes' names*) are filled out, otherwise the submission button will be greyed out (as seen in Figure 5.2). After this process, the knowledgeable administrators can visit a platform's page which shows all the created *entities* and respective *attributes*, where they can also delete them, if necessary.

All the created *entities* and respective *attributes* will then be available as options for creating conditional statements, used in adaptation rules.

5.3.2 Adaptation Rule Creation

The configuration platform includes a page for creating new *adaptation rules*. Figure 5.3 presents this interface, which includes fields for choosing the *trigger* and *action* entities and entity attributes. These fields include a predefined set of options, according to what entities and respective attributes were previously created (as mentioned in subsection 5.3.1). Furthermore, when using the platform, the administrator may notice that when choosing certain entities or attributes, a question mark icon is shown. When hovered, this icon shows a little tooltip with the entity or attribute description, also previously specified during entity creation, which may be useful for remembering what each entity and attribute are used for. Additionally, Figure 5.3.1 also shows a field called "Rule Priority", which will be discussed in section 5.3.2.1.



CONFIGURATION

- + New Rule
- ✓ Adaptation Rules
- + New Entity**
- ☑ Entities
- ⚙ Initial Settings
- 🗄 Database Link

Configuration Platform

New Entity

Entity Name (e.g. Device, User, Weather, etc)

Entity Description

Attribute #1 (d1dc2d0d-4ca2-4517-3d3a-2d2f856accce)

Attribute Name (e.g. Battery, Speed, Temperature, etc)

Attribute Description

Actionable? (i.e. Can the device change the attribute value?)

No

Attribute's Value #1 (4609dcfc-097e-9e93-8382-bcb5edfe5041)

Value Name (e.g. Latitude, Longitude, Visibility, etc)

Value Type

Integer

Value Unit (May be left empty)

+ Add Attribute Value

+ Add Attribute Submit Entity

Figure 5.2: Entity and Attribute creation page.

As it may be obvious, not all entity attributes can have their values rewritten by a rule. For example, for the purpose of this dissertation the user’s location can not be decided by the application. Hence, it was required to differentiate between “normal” and “action-only” attributes. This is the reason why not as many entities may show up as *action* options - only entities containing one or more “actionable” attributes are shown.

As new administrators may find the creation of *adaptation rules* confusing, a verbal description is also automatically generated when the *triggers* and *actions* are chosen. Figure 5.3 shows a description example highlighted in red. Although the English language is not always 100% correct, this description still makes the *rule’s* concept easier to grasp.

Moreover, some entity attributes may accept a POI as a value. The POI Server (section 5.2) has an important role in this feature, as it provides all the POIs that may be chosen as a value, through its API (further discussed in subsection 5.3.3). Hence, knowledgeable

The screenshot shows the 'Configuration Platform' interface. On the left is a dark sidebar with a 'CONFIGURATION' header and a '+ New Rule' button. Below it are menu items: 'Adaptation Rules', 'New Entity', 'Entities', 'Initial Settings', and 'Database Link'. The main area is titled 'Configuration Platform' and contains a 'New Rule' form. The form has three main sections: 'Trigger', 'Action', and 'Rule Priority'. In the 'Trigger' section, 'Entity' is 'User' and 'Entity Attribute' is 'Speed'. The 'Comparison Operator' is '≥ (Greater Than or Equal)' and the 'Value (m/s)' is '10'. In the 'Action' section, 'Entity' is 'Map', 'Entity Attribute' is 'Zoom', and the 'Value' is '10'. Below these sections, a red-bordered box contains the text: 'WHEN User's Speed GTE 10 m/s THEN THE Map's Zoom WILL BE SET TO 10'. The 'Rule Priority' section has a 'Priority' field set to '0' and a green 'Submit' button.

Figure 5.3: Rule Creation on the Configuration Platform.

administrators can define rules such as: “When the User is Near Torre de Belém, Then Open the POI’s InfoWindow”. This feature unlocks many adaptation possibilities, specific to each point of interest. However, there is a special option called “All POIs” that acts as a POI generaliser. In order to avoid possibly having to configure the same behaviour for each POI (for example, generalising the earlier example for every POI available), this special option states that every *trigger* must be **individually** evaluated for **each POI available**. Again, taking the earlier example, it could instead be configured to “When the User is Near *All POIs*, Then Open the POI’s InfoWindow”. This rule would then cause the application to open the InfoWindow of any point of interest considered to be near the user.

Finally, the administrators can also check all the created rules (and delete them) through a page available in the platform, which is sorted by priority.

5.3.2.1 Rule Priority

If this platform did not feature the possibility of storing rule priorities, the administrators would probably realise that some rules may overlap with each other, without a clearly defined way of handling these conflicts. Taking into account an actual behaviour modelled in this system, let us say we wanted to create rules for the following adaptations:

1. When the User's speed is 20 km/h or more, set the Map Zoom to 12.
2. When the User's speed is 50 km/h or more, set the Map Zoom to 10.
3. When the User's speed is 100 km/h or more, set the Map Zoom to 6.

These three rules would automatically adapt the zoom according to the user's speed. However, if the user is moving at a speed of 100 km/h or more, all three triggers are evaluated as true, and all three actions will be executed. Hence, only the last action execution will produce a visible result, because all three actions are writing the same property.

Rule priorities are used to solve these conflicts. These priorities indicate what order should the rules be checked. In other words, the lowest priority rule will be checked first and the highest priority rule will be checked last. This mechanism is able to solve action conflicts as the most important action will set the final entity attribute's value.

Finally, in order to solve the earlier example's problem, the knowledgeable administrators could set the first rule's priority to 1, the second to 2 and the third to 3 (or 4, 5 and 6, or 10, 20 and 50, etc, respectively). Hence, let us say that the user is moving at 70 km/h. Rules one and two will activate their actions. However, as rule two has a higher priority, it will be executed last. Thus, the zoom will be set to 10, as specified in rule two. Basically, with this feature, administrators are able to model specific behaviours with overlapping actions.

5.3.3 Database link

The earlier subsection (5.3.2) mentioned the possibility of creating rules specific to certain points of interest. As this dissertation aims to provide a highly generic solution, not only are the POIs hosted in another server (POI server, section 5.2), but the configuration platform also retrieves these points of interest from a public API, which can be explicitly defined through one of its pages.

As shown in Figure 5.4, this page shows a field containing the current endpoint (if it exists) and another field which can be used to update it. The endpoint URI should be the same as what is used by the Android application - a resource that returns a list of POIs, in JSON format, through an HTTP GET request. Each POI must contain an **id** (unique integer), a **name**, **description**, **latitude** and **longitude** fields - exactly what is provided by the developed POI server. They may contain more information, but it will not be used by the configuration platform.

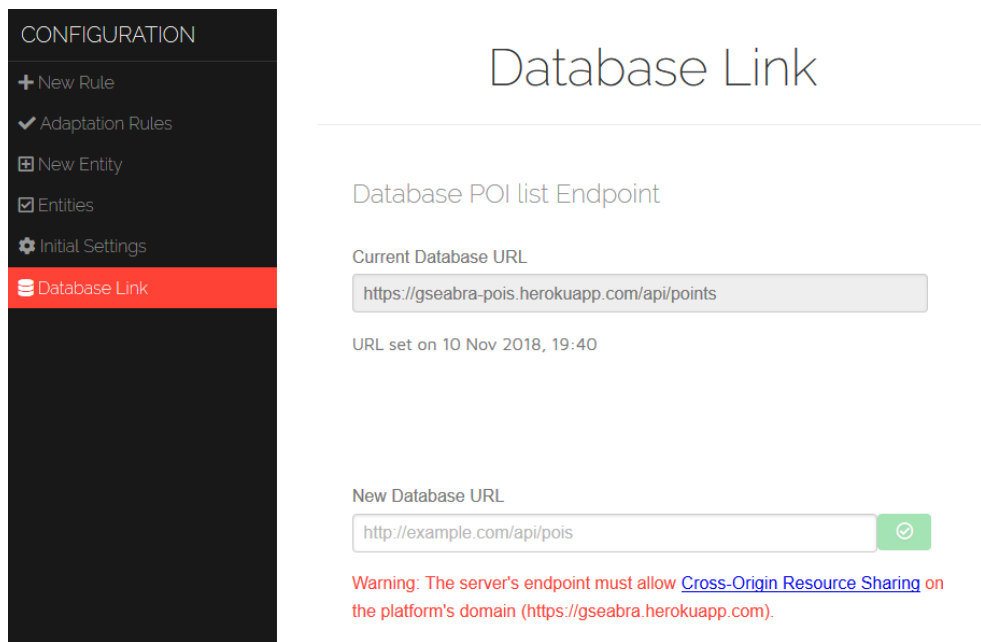


Figure 5.4: Database Link page.

Figure 5.4 also displays a warning message in red. This message’s goal is to remind the administrators that, as most browsers feature a security mechanism called “same-origin policy”, resources from another origin are not accessible by the website (by default). Hence, the endpoint URL needs to allow CORS²⁰ on the platform’s URL, thus allowing it to access the JSON data made available by a POI server.

When correctly configured, the endpoint URI is stored in the platform’s database and will be automatically queried during rule creation. Thus, if the administrators want to create POI-specific rules, only the available points of interest returned by the endpoint will be available as options.

5.3.4 Initial Settings

Although this system’s main focus is automatic adaptation, it may be useful to initialise some properties with a specific value. For example, a tourism application for Lisbon should be centred on Lisbon by default, while a tourism application for Algarve should be centred on Algarve by default.

Hence, the configuration platform includes a page for initialising the value of some properties, called “Initial Settings”, as shown in Figure 5.5.

This page contains fields for setting the minimum, maximum and initial application zoom level, initial map theme (dark or light), initial map centre, automatic adaptations and time interval between rule verifications.

These properties can still have these values later altered by adaptation rules or the user, although the default information may only be changed by the administrators.

²⁰Cross-Origin Resource Sharing

CONFIGURATION

- + New Rule
- ✓ Adaptation Rules
- + New Entity
- ☑ Entities
- ⚙ Initial Settings
- 🗄 Database Link

Initial Settings

Map Settings

Zoom

Minimum Zoom

Maximum Zoom

Initial Zoom

Look

Initial Theme

Center

Latitude

Longitude

App Settings

General

Automatic Adaptation Default

Rule Verification Interval (milliseconds)

✔ Submit Settings

Figure 5.5: Database Link page.

5.3.5 NoSQL Database

As mentioned in section 5.1, the configuration platform stores its data in a MongoDB database. This database is composed of four collections - “settings”, “database”, “entities” and “rules”.

The “settings” collection is a single-document collection, containing all the information related to the initial settings page (see subsection 5.3.4). The structure of the document is shown in Listing 5.1 containing dummy field data (the field names are correctly shown). The *_id* field represents the automatically-generated document ID, and the *setDate* field represents the date when the settings were last updated.

Listing 5.1: “settings” document structure.

```

1 {
2   "_id": "5c0803fb3872641a6c864d18",
3   "min_zoom": 3,
4   "max_zoom": 20,
5   "init_zoom": 12,
6   "init_theme": "Light",
7   "init_center": {
8     "latitude": 38.6968,
9     "longitude": -9.4204
10  },
11  "update_interval": 5000,
12  "start_adapting": true,
13  "setDate": "12/02/2019, 3:47:10 PM"
14 }

```

The “database” collection is also a single-document collection and contains just three fields (see Listing 5.2), which makes it the smallest collection. The *_id* and *setDate* fields are similar to what was just described, while the *link* field holds the endpoint containing the POI data, as described in subsection 5.3.3.

Listing 5.2: “database” document structure.

```

1 {
2   "_id": "5be731da2e9f16da19c3be0f",
3   "link": "http://gseabra-pois.herokuapp.com/api/points",
4   "setDate": "11/10/2018, 7:40:53 PM"
5 }

```

The *entities* collection holds multiple entities. Whenever the knowledgeable administrators create new entities and respective attributes, that information is stored in this collection. Listing 5.3 shows the entity “User” and its attributes (fields with “...” as values were shortened for better fitting this dissertation’s document. Some attributes were also omitted for this reason). This entity is a good example because it comprises all the entity, entity attributes and value possibilities:

Listing 5.3: “entities” document structure.

```

1 {
2   "_id": "5bec617e63455429786de0f7",
3   "name": "User",
4   "description": "May also be interpreted as being the device
5   .",
6   "attributes": [
7     {

```

```
7     "name": "Location",
8     "asAction": false,
9     "fields": [
10        {
11            "name": "Latitude",
12            "type": "number"
13        },
14        {
15            "name": "Longitude",
16            "type": "number"
17        }
18    ]
19 },
20 {
21     "name": "Speed",
22     "asAction": false,
23     "fields": [
24        {
25            "type": "number",
26            "unit": "m/s"
27        }
28    ]
29 },
30 {
31     "name": "Near",
32     "asAction": false,
33     "fields": [
34        {
35            "name": "Location",
36            "type": "$database"
37        }
38    ],
39     "description": "..."
40 },
41 {
42     "name": "Movement",
43     "asAction": false,
44     "fields": [
45        {
46            "name": "Type",
```

```
47         "available": [  
48             "IN_VEHICLE",  
49             "ON_BICYCLE",  
50             "ON_FOOT",  
51             "RUNNING",  
52             "STILL",  
53             "WALKING",  
54             "UNKNOWN"  
55         ],  
56         "type": "custom"  
57     }  
58 ],  
59     "description": "Movement type, detected by Google's  
60     ActivityRecognitionAPI."  
61 },  
62     "setDate": "11/14/2018, 5:55:10 PM"  
63 }
```

Listing 5.3 starts with an *_id* field, similar to the other collections, a *name* and a *description*. This *name* is the entity's name, and the *description* is the entity's description (which will show up as a tooltip during rule creation. The *attributes* field contains all the User entity's attributes.

Similarly, each attribute contains a field with its name. An attribute object also contains an *asAction* field, which indicates if that attribute may be used when building actions. Furthermore, this User entity does not contain any *asAction* property with the value "true", because the application can not change the value of his/her Location, Speed, Movement type, etc. Attribute objects may also contain a *description* field (e.g. Listing 5.3's line 59), containing the respective attribute's description, which will also show up as a tooltip on rule creation.

Furthermore, attribute objects contain an array of objects called *fields*. This array of objects could also be simply called "values" (as it was presented throughout this chapter), but was set as "fields" in order to avoid confusion. Each object represents a field which will be shown during rule creation, if the administrator picks the respective entity and entity attribute. These objects are only required to contain one property - the *type*. The *type* property indicates what type of data may be inserted. Note that on Listing 5.3's line 36, there is a "\$database" type. This is a special type, which tells the application that that field accepts one of the available POIs as a value. Additionally, when it makes sense, these objects may also contain a *name* property (e.g. Listing 5.3's line 11). If there is no such property, the configuration platform will simply display "Value" above the text area. An *unit* property may be included (e.g. Listing 5.3's line 26), specifying the value's unit, in

order to avoid confusion. Lastly, note that there is a *type* whose value is “custom” Listing 5.3’s line 56. When the *type* property is set as “custom”, it is accompanied by an *available* array. The configuration platform is then able to acknowledge that this field is meant to be filled with one of the elements included in the *available* array. Hence, only allowing the administrators to choose one of the options through a selection box.

Finally, the “rules” collection contains one document per adaptation rule. Listing 5.4 contains a rule document example.

Listing 5.4: “rules” document structure.

```
1 {
2   "_id": "5c645f08b583020015bfd5ef",
3   "trigger": {
4     "entity": "User",
5     "attribute": {
6       "name": "Speed"
7       "fields": [
8         {
9           "type": "number",
10          "unit": "m/s",
11          "value": 14,
12          "operator": "GTE"
13        }
14      ]
15    }
16  },
17  "action": {
18    "entity": "Map",
19    "attribute": {
20      "name": "Zoom",
21      "fields": [
22        {
23          "type": "number",
24          "value": 10
25        }
26      ]
27    }
28  },
29  "priority": 9,
30  "setDate": "2/13/2019, 6:16:40 PM"
31 }
```

Similar to the rules described on subsection 5.3.2.1, Listing 5.4's rule changes the map's zoom to 10 whenever the user is moving with a speed of 50 km/h or more (14 m/s is approximately 50 km/h). This document includes two main objects - the "trigger" and the "action" - and one main property - the "priority". As explained earlier in subsection 5.3.2.1, the latter refers to the rule's priority and contains an integer. The main objects have a similar structure - a *entity* property, representing the entity's name, an *attribute* object which contains a *name* property and a *fields* array (also similar to the "entity" document structure). The *type* and *unit* properties represent the value's type and unit, respectively. Although in this example the *action's field's* object does not contain a *unit* property (the map's zoom does not have units), it may appear in other cases. In addition, the *value* property has different meanings in the *trigger* and *action's fields* objects. While the first refers to a value that a certain application property must be compared to, the latter holds the new value for the respective entity attribute.

The *trigger's fields* objects also contain a *operator* property. This property may hold one of the following values, with the respective meanings:

- EQ: Equal to;
- NEQ: Not Equal to;
- GT: Greater than;
- GTE: Greater than or Equal to;
- LT: Less than;
- LTE: Less than or Equal to.

Finally, all this information is available through an API exposed by the configuration platform, which is discussed in the next subsection.

5.3.6 RESTful API

The configuration platform allows administrators to create adaptation rules. As these rules are meant to be used by an external application, there was a need to expose the stored information. As it has been mentioned earlier this chapter, this platform also comprises a RESTful API. The implemented endpoints are represented in Table 5.2. All these endpoints accept or return data in JSON format.

The endpoints whose method is an HTTP GET are easy to understand - numbers 1 and 4 are used to retrieve all the existing rules and entities, respectively, while 9 retrieves all the initial settings defined by the administrators and 7 returns the current external point of interest API URI, if it exists.

Furthermore, the POST-based endpoints are used for creating new information. Number 2 is used for creating a new rule and the request's payload must include a JSON object

Table 5.2: Configuration platform's RESTful API description.

#	Method	URI
1	GET	/api/rules
2	POST	/api/rules
3	DELETE	/api/rules/{id}
4	GET	/api/entities
5	POST	/api/entities
6	DELETE	/api/entities/{id}
7	GET	/api/database
8	PUT	/api/database
9	GET	/api/settings
10	PUT	/api/settings

similar to what was shown in the previous subsection. Likewise, number 5 is used for creating a new entity and its payload must also contain a JSON object similar to what was already shown.

Although the endpoints 8 and 10 are mainly used to **update** information (point of interest API URI and initial settings, respectively), they will create the single-document with the given information, if there is none. Endpoint number 8's payload must simply contain a JSON object with a single property - "link", while endpoint number 9's must contain the new initial settings defined by the administrator.

Finally, the DELETE endpoints - numbers 3 and 6 - are used to delete rules and entities, respectively, and they do not need to include any payload. Instead, when the URI contains a valid rule or entity's ID, the server will delete the document containing that information.

While all the shown endpoints are used by the configuration platform, the Android application simply uses one of them - number 1 - so it can retrieve all the rules which have been created by the administrators, hence being able to effectively perform automatic adaptations according to their specification.

5.4 Android application

The Android application is the realisation of this dissertation. It is a practical example of one of many kinds of applications which the configuration platform may support. This section presents the developed mobile GIS application and was divided into two sections - Geographic Interface and Automatic Adaptation.

5.4.1 Geographic Interface

This application presents a simple interface at first glance - it shows a large interactive map and a title bar (see Figure 5.6a). The title bar contains a button which may be used

to turn automatic adaptations On or Off, while the map contains two buttons placed near the bottom used for zooming in and out.

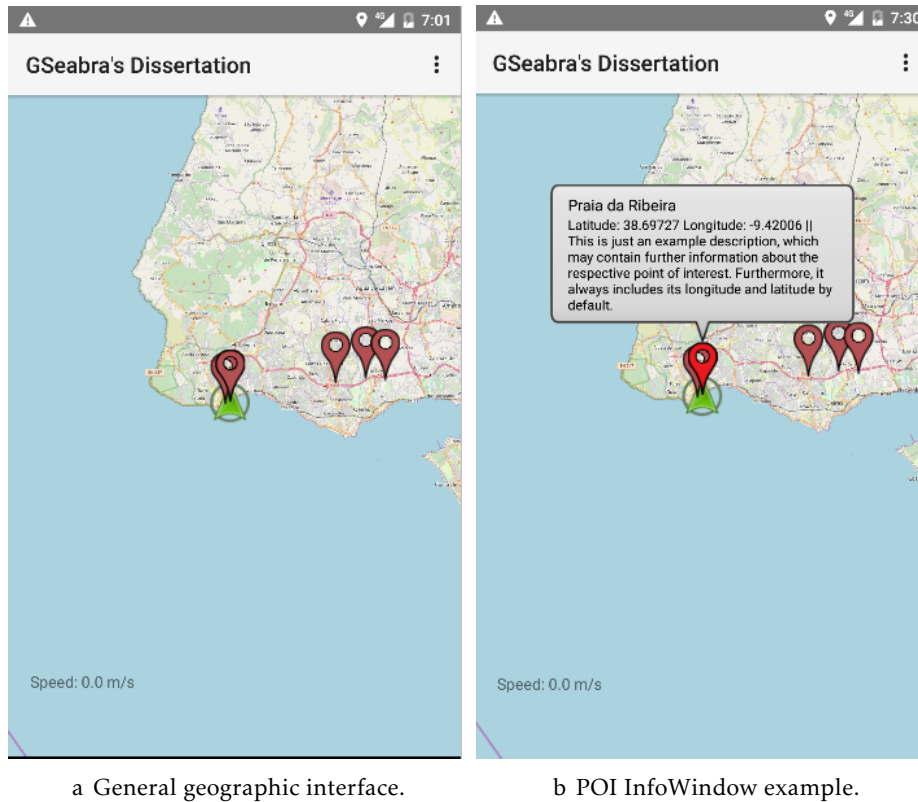


Figure 5.6: Android application interface.

The geographic interface however, contains further details. The application users will quickly notice multiple red coloured markers on the display. Each of these markers represents a different point of interest, registered in the POI server. Furthermore, when a marker is clicked on, a small InfoWindow will pop up above the marker, holding some information. First, it shows the point of interest's name written on top, while the description is presented just below (Figure 5.6b). As specified earlier, the POI server's table holds 4 fields, which are crucial in this application - the "id" is stored internally, for differentiating points of interesting, the "geom" is used in order to decide where each marker will be placed in the map, the "name" and "description" fields are used on the InfoWindows.

As shown in Figure 5.7a, if standing still, a little yellow man represents the user's location, while a green arrow (visible in the previous Figures) indicates the user is moving (the arrow is automatically oriented according to the user's movement direction).

Anyhow, some interface changes only occur when certain rule types are present in the database or when they are activated. The next paragraphs describe some map features which are only visible when under these conditions.

If the administrators have created a rule whose action changes the marker's relevancy

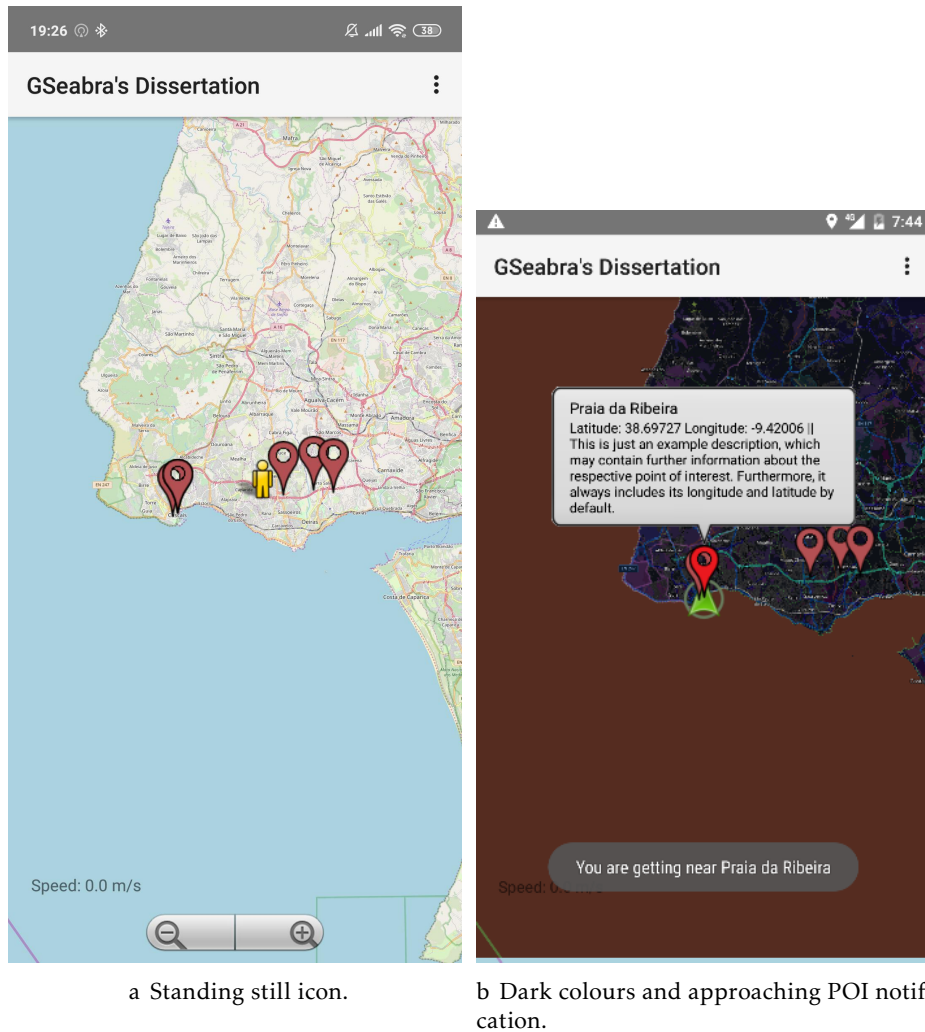


Figure 5.7: Android application details.

to “Relevant” under a certain condition, the marker’s colour will be changed to a brighter red (see Figure 5.6b). Likewise, if a rule’s action acts on the map’s theme, the geographic interface may be changed to dark colours, better avoiding eye strain (Figure 5.7b).

On the other hand, if there is a rule containing a trigger based on what is “Near” the user, a small notification will be shown on the bottom of the screen telling the user when a new point of interest is considered to be near the user (see Figure 5.7b). Finally, when there is at least one rule based on the user’s speed, the current speed value will be shown in the screen, and the same principle applies to the ambient lighting and user movement type (walking, driving, etc).

5.4.2 Automatic Adaptation

As the work developed in this dissertation aims to provide a very generalised system, able to support many kinds of applications, a generic system was developed to achieve this. Hence, the general operational steps are as follows - first, the application initialises

the map and all the necessary libraries. Then, it fetches all the initial settings from the configuration platform and initialises the variables accordingly. After this, all the POI data is queried and inserted into the map and finally the adaptation rules are fetched and passed into *RuleController* - a crucial component - which will periodically be told to check the rules, to see which ones should be activated.

The component “SettingsManager” is responsible for the initial configuration and is only used during the application’s start up. This component simply reads all the JSON data returned by the configuration platform’s settings endpoint and initialises all the application’s variables mentioned in the document accordingly.

However, as mentioned before, the “RuleController” is a crucial component - it controls all the automatic adaptations. After storing all the rules locally, the main activity queries this component from time to time (the interval is defined by the administrators) so it is able to know which actions to activate. This query is done through a package method which will analyse all the stored rules or, more specifically, their triggers. Listing 5.5 shows a simplified version of this method.

Listing 5.5: “RuleController” trigger checking.

```
1 void checkRules(MapView map) {
2     for (Rule rule : rules) {
3         Trigger trigger = rule.getTrigger();
4         Attribute triggerAttribute = trigger.getAttribute();
5         boolean activateRule = false;
6
7         switch (trigger.getEntity()) {
8             case RuleNames.USER: {
9                 switch (triggerAttribute.getName()) {
10                    case RuleNames.LOCATION: {
11                        activateRule =
12                            markerManager.isUserAt(getMyLocation());
13                        break;
14                    }
15                    case RuleNames.SPEED: {
16                        double currentSpeed = getSpeed();
17                        activateRule =
18                            triggerAttribute.matchedValues(currentSpeed);
19                        break;
20                    }
21                    case RuleNames.MOVEMENT: {
22                        activateRule =
23                            triggerAttribute.matchedValues(activityState);
24                        break;
25                    }
26                }
27                break;
28            }
29
30            case RuleNames.BATTERY: {
```

```
31         switch (triggerAttribute.getName()) {
32             case RuleNames.LEVEL: {
33                 int battery = getBatteryLevel();
34                 activateRule =
35                     triggerAttribute.matchedValues(battery);
36                 break;
37             }
38         }
39         break;
40     }
41
42     case RuleNames.MAP: {
43         switch (triggerAttribute.getName()) {
44             case RuleNames.ZOOM: {
45                 double zoom = map.getZoomLevelDouble();
46                 activateRule =
47                     triggerAttribute.matchedValues(zoom);
48                 break;
49             }
50             case RuleNames.THEME: {
51                 activateRule =
52                     triggerAttribute.matchedValues(mapTheme);
53                 break;
54             }
55         }
56         break;
57     }
58
59     case RuleNames.PROXIMITY: {
60         switch (triggerAttribute.getName()) {
61             case RuleNames.DISTANCE: {
62                 int prox = getProximityDistance();
63                 activateRule =
64                     triggerAttribute.matchedValues(prox);
65                 break;
66             }
67         }
68         break;
69     }
70
71     case RuleNames.APPLICATION: {
72         switch (triggerAttribute.getName()) {
73             case RuleNames.UPDATE_INTERVAL: {
74                 activateRule =
75                     triggerAttribute.matchedValues(interval);
76                 break;
77             }
78         }
79         break;
80     }
```

```
81         case RuleNames.AMBIENT: {
82             switch (triggerAttribute.getName()) {
83                 case RuleNames.LIGHT: {
84                     activateRule =
85                         triggerAttribute.matchedValues(getLight());
86                     break;
87                 }
88             }
89             break;
90         }
91     }
92
93     if (activateRule) {
94         activateRule(rule.getAction());
95     }
96 }
97 }
```

Although this method includes many code lines, it is not hard to understand. It simply iterates all the stored rules and checks their triggers, one by one. There is a variable called “activateRule”, which dictates if the rule’s action is activated. Furthermore, each “Attribute” object contains a method called “matchedValues” which returns a boolean - it checks the trigger’s comparison operator and value and compares it to the value given by the argument. Finally, it returns the result of the trigger evaluation and that is what effectively tells the “activateRule” its new value. Additionally, the method is long because of the nested “switch” statements. Although their contents are quite similar, they differ depending on the trigger’s entity and entity attribute as that is what dictates what value shall be passed to the function “matchedValues”.

Furthermore, as seen on line 94 of the same Listing, there is a method called “activateRule”, which takes an action as an argument. This function is responsible for activating rules and, as all the trigger evaluations have already been performed by its caller, it just needs to analyse (in a similar nested-switch fashion) what the action’s entity and entity attributes are and set their new value. In other words, every single trigger condition will be evaluated during the “checkRules” function’s iterations. However, the function “activateRule” will only be entered as many times as there are trigger conditions evaluated to *true*.

Listing 5.5’s line 12 shows a different way to handle the “activateRule”’s new value - it queries an object called “markerManager”. In order to separate concepts, a component called “MarkerManager” was created for handling map/marker/location-related rules. This component not only places all markers on the map, but also executes operations such as opening InfoWindows, keeping track of the user’s distance to each point of interest, changing the marker’s colour, etc. As previously mentioned, some rules may include triggers specific to certain POIs. Although they are not included in this Listing (for space related purposes), they look similar to the location verification, with a simple twist - they

inquire the “MarkerManager” component if the user is near a POI with a certain *id* and, if that is the case, change the “activateRule”’s value. In case administrators have specified the value “All POIs”, the “MarkerManager” component will instead evaluate all the POIs and return a list of *ids*. Lastly the action will be applied to every single POI whose *id* is in the list.

The nested switch implementation is also advantageous due to the fact that adding new entities and entity attributes can be done effortlessly - one just needs to add an existing “case” and define what is to be checked and what is to be executed, in case it gets activated. This characteristic was implemented keeping in mind that genericity and extensibility of the solution should be maximised.

However, this system does not deactivate rules by itself. In other words, if a trigger condition goes from “true” to “false”, the action performed is not reversed, unless the administrators specify a rule which reverses another (it is not always possible). As it will be mentioned later, this will be a feature suggested as future work.

This application avoids keeping unnecessary information consuming resources, by doing an evaluation of what will be needed during its execution. Right after receiving the adaptation rules from the configuration platform, the component “RuleController” will iterate the rules and instantiate resources based on what is present in the list. For example, if it detects that no triggers need data from the device’s sensors or the user’s movement type, the application will not instantiate the “SensorManager” nor Google’s ActivityRecognitionAPI, respectively.

Finally, as this application was designed as a tourism application, some adaptation rules were created in order to improve the user’s experience while visiting a city.

5.4.2.1 Created rules

Although the Android application contains the basic requirements of a tourism application, even if no rules were added (it shows POIs, their information, etc), this dissertation aims to improve this basic functionality.

For clarification purposes, the entity “Proximity” and its attribute “Distance” refer to the distance of what is considered to be in close proximity of the user. For example, if a rule changed this distance to 100 meters, it would mean that every POI 100 or less meters away from the user would be considered near him/her. On the other hand, the entity “User” and its attribute “Near” are dependent on the previous entity/attribute. By default the proximity distance is 50 meters, but it may not be suitable in all cases (e.g. when driving a car 50 meters is considered too close). Thus, if the distance is unchanged by an action and there is a rule whose trigger is “User - Near - *Pelourinho*”, the action will only activate if the POI *Pelourinho* is within 50 meters of the user. The entity/attribute “Proximity - Distance” is able to dynamically change this default distance.

Hence, some adaptation rules were added in order to support an extended functionality. This information will be organised in two sections - “Recommended rules” and

“Non-essential rules”. They will be shown as:

- Trigger entity - Trigger entity attribute - (Trigger comparison operator) - Trigger value
- Action entity - Action entity attribute - Action value
- Priority

Recommended rules First of all, it is important to automatically open the InfoWindow of the POIs near the user. As *osmdroid* only allows one InfoWindow to be open at any time, the nearest POI is the most desirable, in the majority of the cases. This rule would be structured as follows:

- User - Near - All POIs
- Nearest POI (Marker) - InfoWindow - Open
- 0 (no other action will change the InfoWindow’s state)

As the rule contains a “Near” entity attribute, the application will also show a notification every time the user is near a new POI.

Furthermore, although this system supports rules based on movement types, during testing we discovered that speed-based rules usually work better, as Google’s API may return wrong values sometimes. For example, when riding a car, this API would sometimes indicate that the user was walking or even in an unknown state. Hence, three rules were created for changing the *Proximity’s Distance* based on the user’s speed. When walking, 50 meters were considered to be a good distance, while 200 meters and 500 meters were chosen when the user is moving at a speed of up to 70 km/h or anything greater than that, respectively. Thus, the created rules are as follows:

1.
 - User - Speed - \leq - 10 (km/h)
 - Proximity - Distance - 50 (m)
 - 10
2.
 - User - Speed - \leq - 70 (km/h)
 - Proximity - Distance - 200 (m)
 - 0 (the previous rule takes priority for a correct behaviour)
3.
 - User - Speed - $>$ - 70 (km/h)
 - Proximity - Distance - 500 (m)
 - 0 (no other action will change the proximity distance based on the user’s speed being **greater than** a value)

As these rules include triggers based on the user's speed, the current speed will be displayed on the screen as well.

Likewise, when the user is not walking (i.e. driving a vehicle), automatically centring the map on his/her location greatly improves the user experience. Hence, a simple rule was created:

- User - Speed - > - 10 (km/h)
- Map - Center - User
- 0

Furthermore, it is also important to change the map's zoom automatically when the user is driving at different speeds (so they do not need to do it and potentially putting themselves in danger). Hence, three rules were created:

1.
 - User - Speed - > - 30 (km/h)
 - Map - Zoom - 11
 - 0
2.
 - User - Speed - > 60 (km/h)
 - Map - Zoom - 10
 - 1 (takes priority over the previous rule)
3.
 - User - Speed - > - 100 (km/h)
 - Map - Zoom - 9
 - 2 (taking priority over the two previous rules)

Lastly, some simple improvements may be modelled with rules, although they are not as important for a good user experience.

Non-essential rules In order to reduce the battery drain, some rules may be created for increasing the rule-checking interval in cases where it is not so important to constantly update information. For example, when walking, the interval may be increased, as the user is not moving too fast.

1.
 - User - Movement - WALKING
 - Application - Update Interval - 7000 (ms)
 - 0
2.
 - User - Movement - IN_VEHICLE
 - Application - Update Interval - 4000 (ms)

- 0

Furthermore, in order to avoid eye straining, it may be desirable to set the map's theme to dark when it is dark, but it is also important to revert this rule as soon as its bright again (for example, when entering and exiting a tunnel, the map's theme should be changed). Hence, two rules may be added:

1.
 - Ambient - Light - \geq - 40 (lux)
 - Map - Theme - Dark
 - 0
2.
 - Ambient - Light - $<$ - 40 (lux)
 - Map - Theme - Light
 - 0

Finally, in cases where it is desirable to keep track of what POIs the user has already been near to, the marker's colour may be changed permanently to "Relevant":

- User - Near - All POIs
- All POIs (Markers) - Colour - Relevant

5.5 Conclusion

Throughout this chapter, the set of functionalities provided by the developed solution was presented. This solution was developed with the mindset that it should highly generalisable and extendable, hence potentially be fitting for a wide variety of concrete mobile applications.

It starts by describing the points of interest server and what is stored in its database. This server is used by both the Android application and configuration platform, for placing markers on the map and creating rules specific to each POI, respectively.

Additionally, the configuration platform was introduced by describing the core concepts and explaining how they integrate this system. This section starts by describing what entities, entity attributes, comparison operators and values mean, so it is easier to understand the rest of the section, thus better understanding how the whole system effectively models rule-based automatic adaptations. This platform is a tool for knowledgeable administrators and allows them to create adaptation rules which are meant to be passed onto the Android application and telling it what to adapt and when to do it.

The Android application is a tourism application containing multiple points of interest which was designed as a proof of concept of the designed context-modelling system. This application starts by applying all the initial settings defined by the administrators

on the configuration platform, then it places all the markers on the map and finally it retrieves the JSON data containing all the adaptation rules which have been created. Lastly, during its operation, the application periodically iterates all rules and compares them to the application's state, thus dictating which actions shall be activated.

Finally, after concluding the development of the configuration platform, points of interest server and Android application, an evaluation process took place in order to analyse the potential success of the model. This process is further described in the next chapter.

EVALUATION

This chapter starts by describing the methodology adopted in order to evaluate the developed system (excluding the POI server, for a lack of interface). Then, the results of the evaluation process are presented and, finally, some conclusions are drawn out from this information.

6.1 Methodology

Once this dissertation's system was implemented, there was a need to determine if the configuration platform and, most importantly, the Android application, achieved their initial purposes and if these software components matched the expected usability results. Namely, we wanted to verify: if the context-based adaptation model implementation successfully allows the creation of adaptation rules used in the Android application; the chosen set of automatic adaptations based on certain context characteristics were well chosen for the Android application. Hence, Figures C.1, C.2 and C.3 portray the audience of the evaluation of this system. It is worth mentioning that users having computer-related studies intentionally comprise the majority of the participants. As the configuration platform is supposed to be handled solely by knowledgeable administrators, only the first group was asked to evaluate the platform, but both groups were capable of evaluating the Android application.

The evaluation process was meant to evaluate the usability and usefulness of the configuration platform and/or Android application. The participants were asked to perform several tasks in the developed system. There were two main parts in this component: one meant to perform configuration platform-related tasks; and another for Android application-related tasks. While the configuration platform's tasks were to create adaptation rules specifying some adaptations meant to be used later by the application, the

Android application's tasks consisted in not only using it while driving a car on a highway, but also while walking in *Cascais* and seeing some predetermined points of interest. Moreover, the Android application users were also asked to turn the automatic adaptation off during a part of the tour, so they could compare both versions. The task was designed so the participants would first test the application with automatic adaptations and later turn them off, so the lack of adaptation would be more obvious. Lastly, this component had a total of 16 participants (see Appendix C for more details) with different ages and backgrounds. A total of twelve participants evaluated the configuration platform. Three of these did not test the Android application, as they were not available during the testing days. Furthermore, four other subjects simply evaluated the Android application because they lacked sufficient computer knowledge for evaluating the platform. Finally, as it may be deduced later, the Android application tasks while driving had less participants than the walking tasks. This is because some participants were driving the vehicles, thus being unable to look at their smartphones.

After performing the tasks described in Appendix A, each participant answered a questionnaire. This questionnaire, which is shown in Appendix B, was divided into two parts - configuration platform evaluation and Android application evaluation. Hence, each participant was asked to fill out only the part(s) respective to the tasks they had performed. Moreover, each of these two parts was further divided into two sections; the first was constituted by the *System Usability Scale* (SUS) [6, 7], whereas the second section comprised more specific questions related to specific features of the tested platforms. As the points of interest server was not evaluated by any of the participants, all the planned route's POIs were inserted before the task executions. Furthermore, as the participants were given access to the configuration platform, a hidden button was created in order to reset all the information back to a previous, known and correct restore point. This way each of the participants faced the exact same scenario and conditions before performing the tasks.

As mentioned in the evaluation's description, the System Usability Scale (SUS) was present in the participants' questionnaires. This system is a robust, reliable and low-cost tool for measuring a system's usability. Furthermore, it may be used for evaluating a wide variety of products and services, including applications, software, hardware and websites, providing reliable results, even if used on small sample sizes. Since its creation, this tool has become a standard in the industry, and it is able to effectively distinguish between usable and unusable systems. This tool comprises ten affirmations, each one containing a 5-item linear scale (similar to a Likert scale), varying between "Strongly disagree" - value 1 - to "Strongly agree" - value 5. The SUS suggested questions are:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.

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

In order to obtain the SUS score of a participant, one must first sum the values selected in all 10 questions according to very specific rule: for each question numbered with an odd number (questions 1, 3, 5, 7 and 9), the value to be summed is obtained by subtracting 1 to the value chosen by the participant. For example, if the participant chose “Totally Agree” in question 1, the value which would be summed to the total SUS score is 4 (because “Totally Agree” is the same as 5). However, for even-numbered items (questions 2, 4, 6, 8 and 10 - negatively worded items), the value to be summed to the score is obtained by subtracting the chosen number to 5. For example, if the participant chose “Agree” in question 2, the value which would be summed to the total SUS score is 1 (because “Agree” is the same as 4). After this transformation is applied, each item contribution varies between 0 and 4, hence making the SUS score comprise a score from 0 to 40. In order to obtain the score in a 0 to 100 scale, it is necessary to transform each contribution by 2.5.

According to Lewis and Sauro in [24], a global SUS value above 68 is considered above average. On the other hand, a value inferior to 68 is below average, and the system is considered to have usability issues. However, Bangor *et al* [5] have stated that the value 70 has been considered as the mean value of SUS when several types of interface are being evaluated. However, it is also specified that the “cell phone” interface type has a total mean score of 66 while the web interface type has a total mean score of 68 based on the same studies. Hence, these two values will be considered in the evaluation of the Android application and configuration platform, respectively.

Although the SUS tool was initially designed to measure just the usability dimension, the research carried out by Lewis and Sauro [24] suggests that two dimensions may instead be derived from SUS - usability and learnability. Both these dimensions have been considered in the evaluation in order to complement the global SUS value and thus providing a better perception of this dissertation’s developed systems total usability. The usability dimension may be calculated by summing the values associated with the answers of SUS’s questions 1, 2, 3, 5, 6, 7, 8 and 9 and then multiplying the result by 3.125. The learnability dimension is obtained through a similar process - first, by summing the

values of the remaining questions (4 and 10), and then multiplying the resulting number by 12.5.

Taking into account the previously mentioned studies, some modifications have been performed to the original SUS questionnaire proposed by Brooke [6, 7]:

- The question “Overall, I would rate the user-friendliness of this application as:” was added and it provides 7 adjective-based answer possibilities - 1: *Worst Imaginable*, 2: *Awful*, 3: *Bad*, 4: *OK*, 5: *Good*, 6: *Excellent*, 7: *Best Imaginable*. According to Bangor *et al* [5], the answers to this question present a high correlation to the value obtained from the SUS score;
- The second set of SUS questions (related to the Android application) had the term “system” replaced by “application” in each statement;
- the first set of SUS questions (related to the configuration platform) had the term “system” replaced by “platform” in each statement.

6.2 Summative Assessment

The evaluation results obtained from the questionnaire answers of the configuration platform are shown in Table 6.1 and Figure 6.1. Although only one evaluation (8.3%) has given an adjective with a negative connotation - *OK* - four participants (33.3%) had a SUS score below the acceptable level of 68. The most common adjective was *Good* which represents 66.7% of the total answers. Overall the learnability values were spread around the acceptable level, having scored lower than both the SUS and usability. However, the mean learnability score is still above the acceptable level, with a value of 69.8. On the other hand, the mean SUS score is equal to 72.5, whereas the mean usability score is 73.2. Hence, it can be concluded that the configuration platform is considered to be acceptable concerning both usability and learnability. As the mean learnability score was barely above the acceptable level, it can also be concluded that the idea of the platform only being used by knowledgeable administrators is correct - it may be hard to comprehend all the concepts at first, but after understanding them, the platform is quite usable, as the SUS and usability scores suggest.

On the other hand, the Android application had very different results. The results of this evaluation are depicted in Table 6.2 and Figure 6.2. As mentioned earlier, this application has a “cell phone” type interface, hence the acceptable level is 66. Regarding the overall SUS score, no participants had a score below this acceptable level. The most rated adjective was *Excellent* with 8 (61.5%) of responses, while the lowest was *Good* with a total of 4 (30.8%) evaluations. The mean SUS score is equal to 81.5, which is well above the value of 66. Furthermore, the learnability scores were generally higher than the usability’s, the first having a mean score of 85.4 whereas the second has a value equal to 80.5. It can then be stated that the Android application had much better scores than

Table 6.1: Results of the SUS Questionnaire (Configuration Platform).

Participant	Adjective rating	SUS score	Usability score	Learnability score
1	Good	65	65.625	62.5
2	Good	75	78.125	62.5
3	Good	62.5	59.375	75
4	Good	67.5	68.75	62.5
5	Excellent	87.5	84.375	100
6	OK	60	62.5	50
7	Good	72.5	78.125	50
8	Good	75	75	75
9	Excellent	80	81.25	75
10	Good	70	68.75	75
11	Good	75	78.125	62.5
12	Excellent	80	78.125	87.5
Mean		72.5	73.2	69.8
Min.		60	59.4	50
Max.		87.5	84.4	100
Std. Dev.		7.6	7.6	13.9

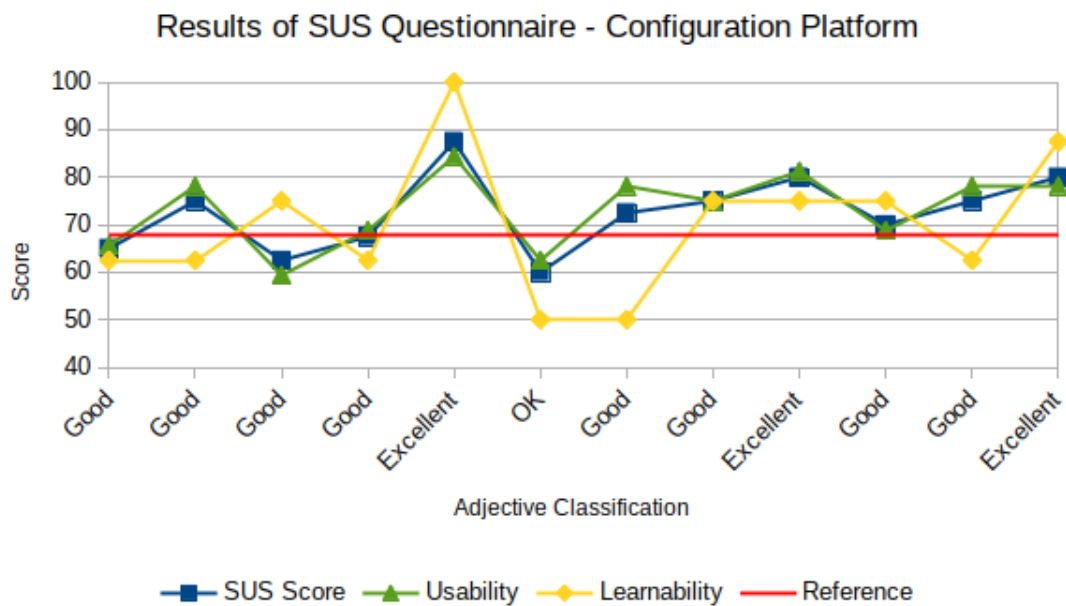


Figure 6.1: Results from the SUS Questionnaire (Configuration Platform).

the configuration platform and can be considered acceptable concerning both usability and learnability.

Table 6.2: Results of SUS Questionnaire (Android application).

Participant	Adjective rating	SUS score	Usability score	Learnability score
1	Excellent	85	81.25	100
2	Excellent	85	84.375	87.5
3	Excellent	90	87.5	100
4	Excellent	72.5	65.625	100
5	Good	80	78.125	87.5
6	Good	80	81.25	75
7	Excellent	82.5	81.25	87.5
8	Best Imaginable	92.5	90.625	100
9	Excellent	72.5	71.875	75
10	Excellent	85	84.375	87.5
11	Good	72.5	75	62.5
12	Excellent	87.5	90.625	75
13	Good	77.5	75	87.5
Mean		81.5	80.5	85.4
Min.		72.5	65.6	62.5
Max.		92.5	90.6	100
Std. Dev.		6.6	7.3	11.2

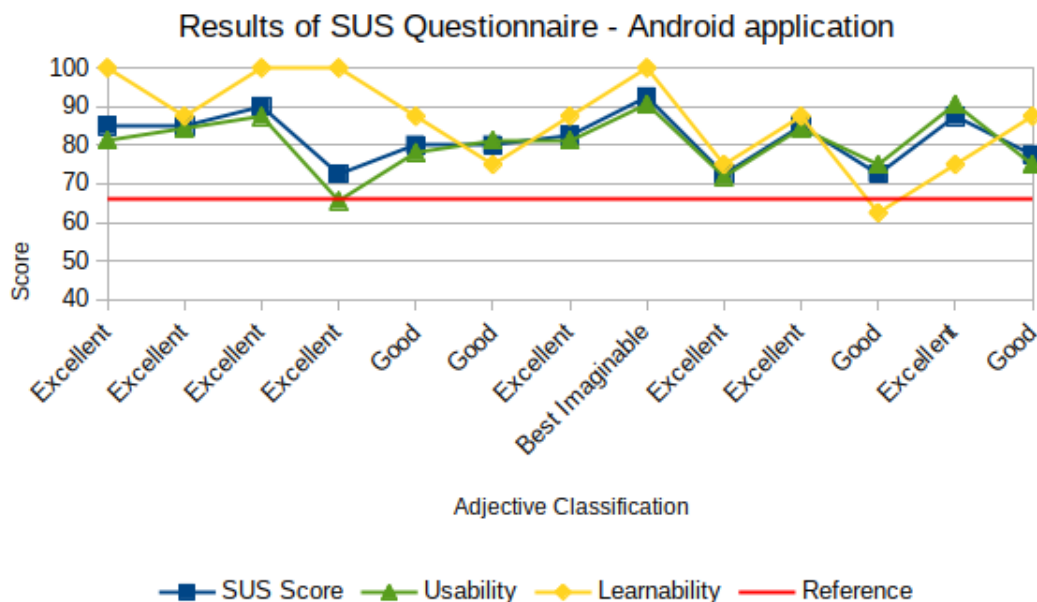


Figure 6.2: Results from the SUS Questionnaire (Android application).

When comparing the adjective ratings present in Tables 6.1 and 6.2, one can understand how the participants evaluated the general user-friendliness of both the configuration platform and Android application. The Android application had overall better

adjective ratings, as expected when taking into consideration its SUS results. The application contained no *OK* ratings (unlike the platform), and was evaluated once with *Best Imaginable* and had a general superior amount of positive connotation adjective ratings, compared to the configuration platform. These results are depicted in Figure 6.3, along with the results of the research carried out by Bangor *et al.*. The authors obtained these values as a result of 959 surveys, however, this dissertation had a total of 16 participants (4 of them did not perform the platform's tasks and 3 of them did not perform the application's tasks). Yet, as the Figure shows, the results obtained in this dissertation are similar to the study's.

Comparison between the average SUS score per adjective in our study and Bangor et al's.

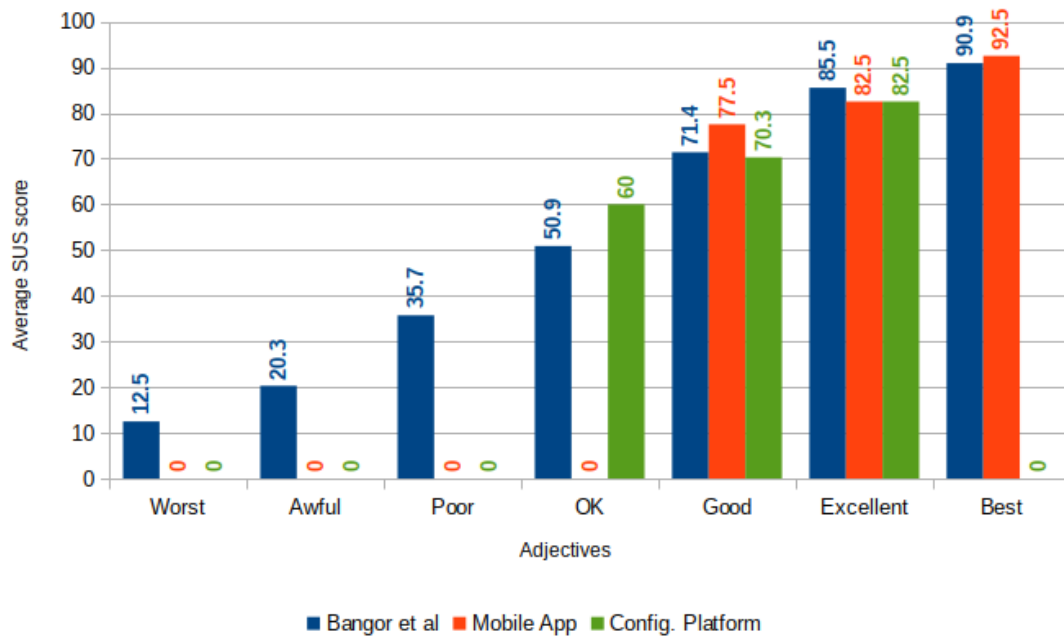


Figure 6.3: Comparison between adjective ratings and SUS scores.

Finally, the fact that the Android application's interface is quite simple may have contributed for a good evaluation. Participants did not need to interact much with the application (besides looking at the displayed information), hence, even users with little technology knowledge will find it easy to use it. On the other hand, the configuration platform was briefly explained to the participants before their usage. Although they had computer knowledge, most of them did not have knowledge specific to GIS, context and adaptation, which may help explain the low SUS score and the difference in learnability results.

6.3 Formative Assessment

In this section, the results from the questions pertaining specific characteristics from both the configuration platform and Android application's interfaces will be presented. This

part of the questionnaire comprised more statements to which the participants had to reply on a Likert-like scale, similar to what they had done on the SUS component.

6.3.1 Configuration Platform

This subsection presents the formative assessment done to the configuration platform. It was evaluated by a total of twelve computer-knowledgeable participants.

Figure 6.4 suggests that the majority of the participants had a positive experience while using the platform, while Figure 6.5 shows that most of them also has no trouble understanding the necessary concepts for rule creation. This can be explained by the fact that only computer science students (or ex-students) performed this evaluation. Yet, one third of the participants had some trouble while using the platform, as shown in the second Figure. This information further consolidates the idea that the platform should only be used by knowledgeable administrators. Nevertheless, these results are still positive and, therefore, it can be stated that the platform is easy to use.

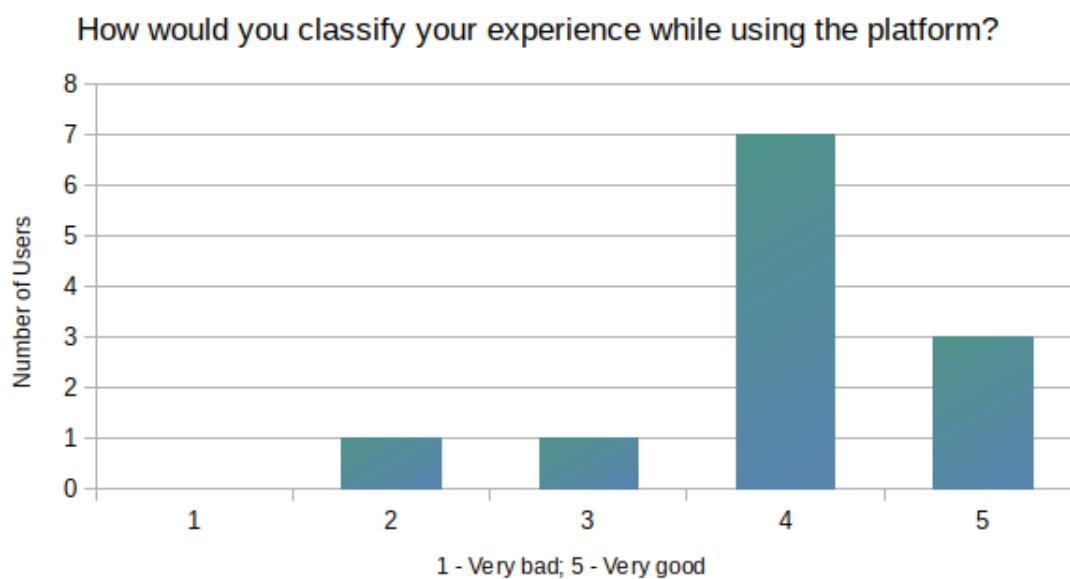


Figure 6.4: Question 1: How would you classify your experience while using the platform?

Furthermore, the participants found it easy to create the adaptation rules in the task list - only two users (16.7%) remained neutral, while four (33.3%) answered with “Agree”, and a total of six participants (50%) answered “Strongly Agree”, as represented in figure 6.6. Moreover, Figure 6.7 shows that every participant agreed to a certain extent - 3 of them (25%) agreed and 9 strongly agreed (75%) - that they could effortlessly verify which rules had been created.

Figure 6.8 shows that knowledgeable administrators would not have problems understanding what each rule present in the list would perform - 6 participants (50%) agreed with the statement, while 4 (33.3%) strongly agreed. However, two users (16.7%) did not

I think it was easy to understand the relation between rules, triggers, actions, entities, attributes, values and comparison operators.

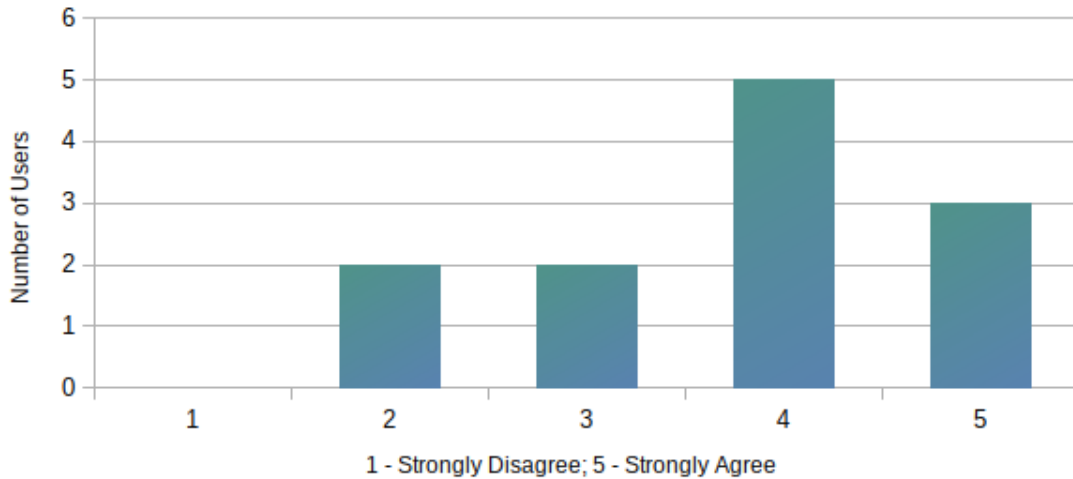


Figure 6.5: Statement 1: I think it was easy to understand the relation between rules, triggers, actions, entities, attributes, values and comparison operators.

agree with this statement (one stayed neutral and another disagreed), suggesting that it would be beneficial to take more time introducing this platform to administrators.

I think it was easy to create adaptation rules.

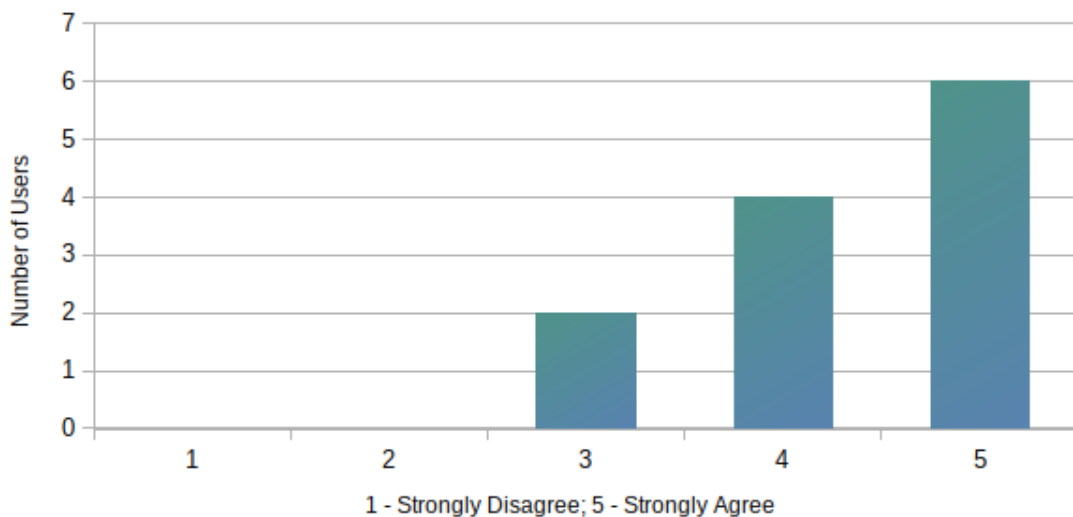


Figure 6.6: Statement 2: I think it was easy to create adaptation rules.

Figure 6.9 shows that every participant agreed that the provided descriptions about entities and attributes are useful, having had 7 (58.3%) participants strongly agreeing with the statement and the remaining 5 (41.7%) also agreed. This feature was added

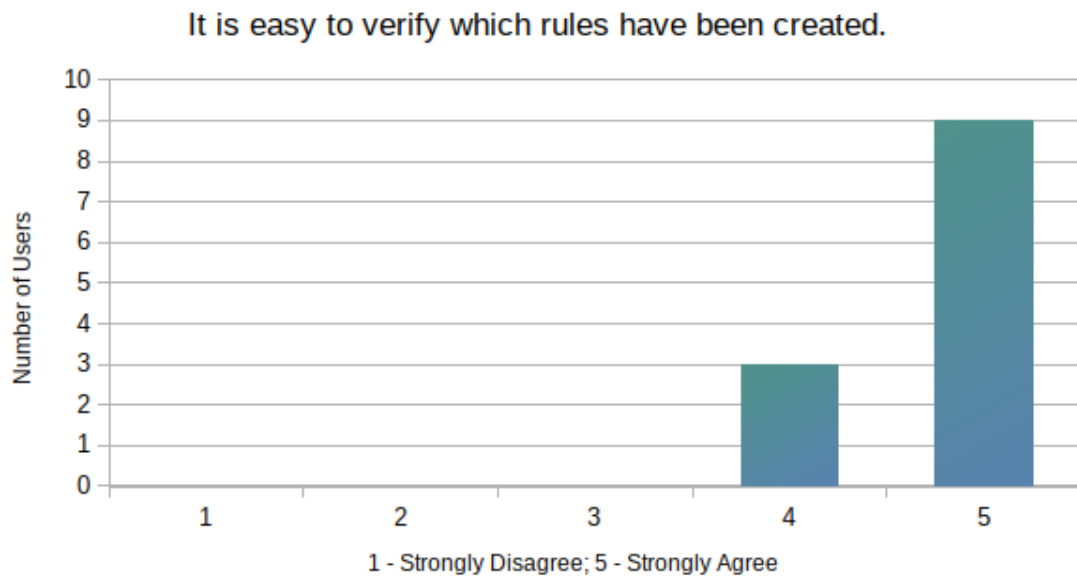


Figure 6.7: Statement 3: It is easy to verify which rules have been created.

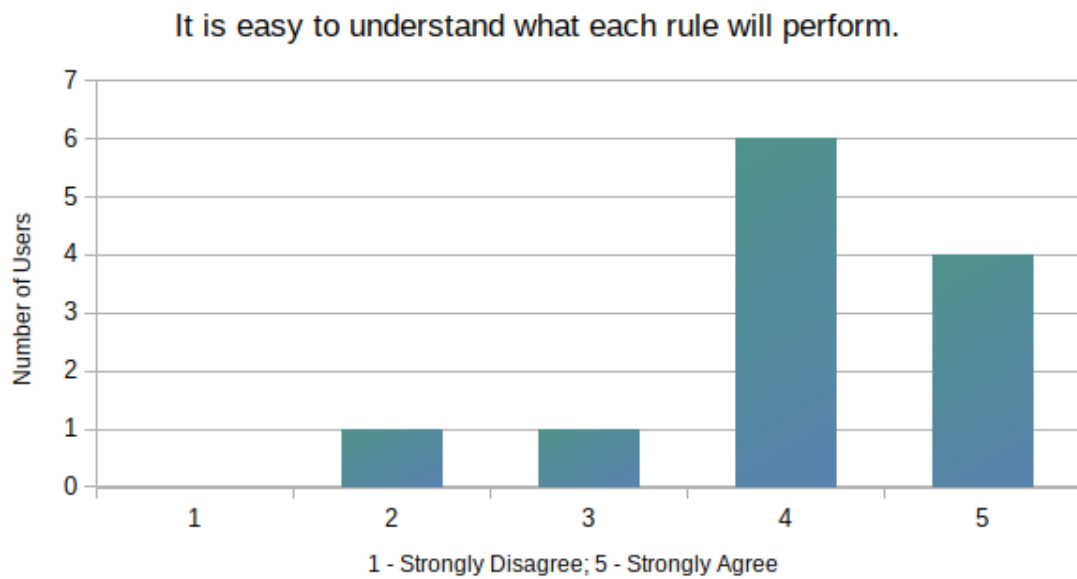


Figure 6.8: Statement 4: It is easy to understand what each rule will perform.

during the final stages of development because a previous assessment revealed that some entities and attributes were confusing (such as Proximity - Distance, User - Near).

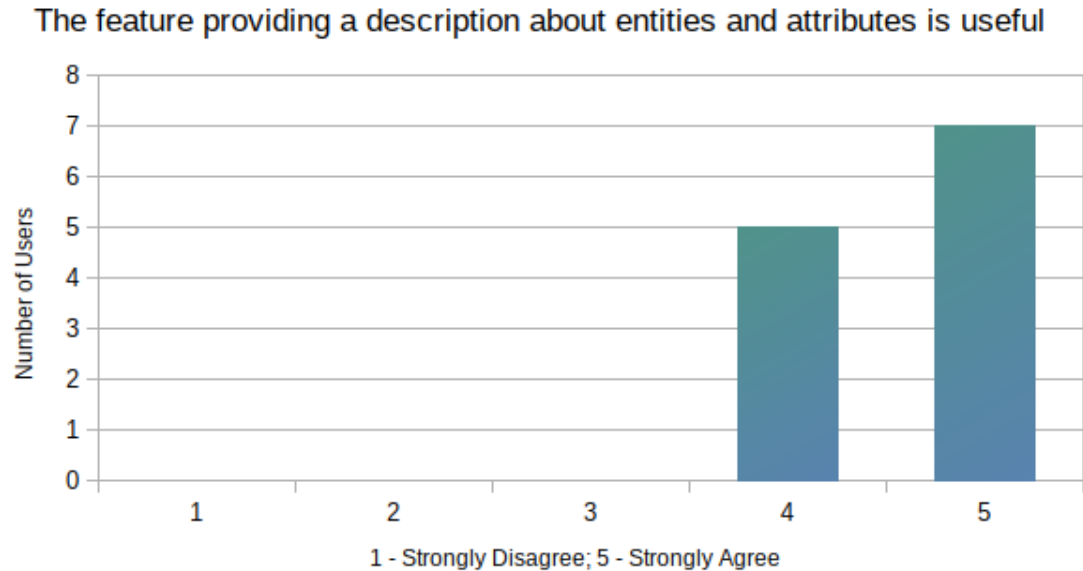


Figure 6.9: Statement 5: The feature providing a description about entities and attributes is useful.

Although the results shown in Figure 6.10 are also very positive, one of the participants (8.3%) gave a neutral answer about the rule priority feature. This participant later wrote some feedback, mentioning “Although the rule priority feature works fine, I think it would be more intuitive to create rules with intervals”. The rule priority mechanism is more generalised (some cases may not work with inequality intervals), nevertheless, this is an interesting approach that may be considered as future work on the cases where it makes sense.

Figure 6.11 shows that two users did not think the platform’s style adaptation to mobile devices was useful. When asked about this, both participants mentioned that they did not find that feature too useful because they thought the platform was most likely to be used in a desktop anyway. Nevertheless, the rest of the users (83.3%) agreed with the statement, to a certain degree - 5 agreed while another 5 strongly agreed.

While the participants felt that the configuration platform’s goals are useful, 4 of them (33.3%) remained neutral when asked if it made sense to define the application’s adaptation through the platform, as shown in Figures 6.12 and 6.13. This can be explained by a lack of understanding of the advantages of defining the adaptation on the application itself or on the platform - the latter provides a tool which may be used on a wide variety of applications, while the first is restricted to its own application. Nevertheless, the general results of both these questions are quite positive.

Figure 6.14 shows that 10 participants (83.3%) ran into problems during their tasks.

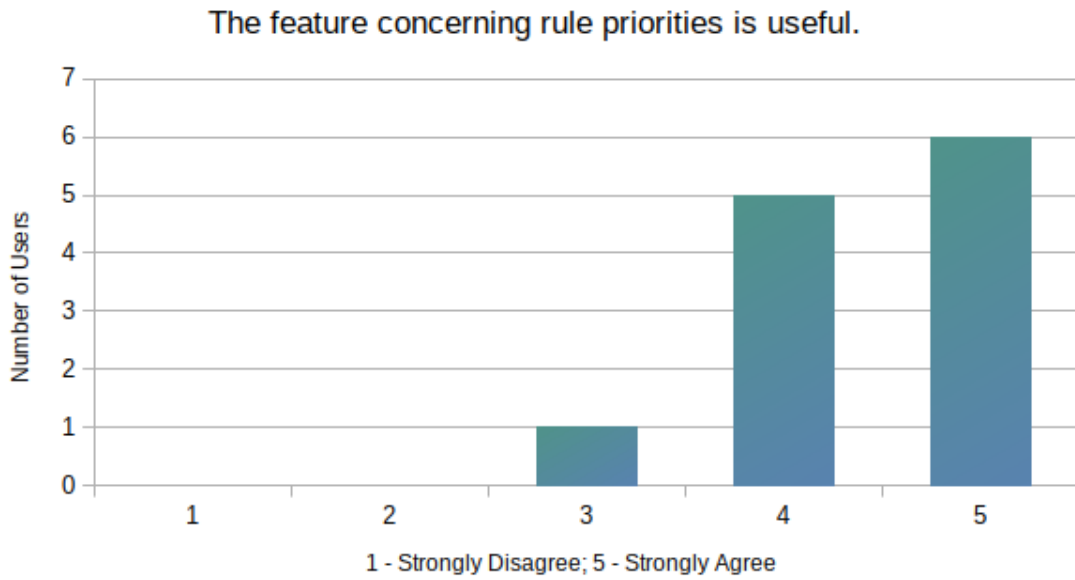


Figure 6.10: Statement 6: The feature concerning rule priorities is useful.

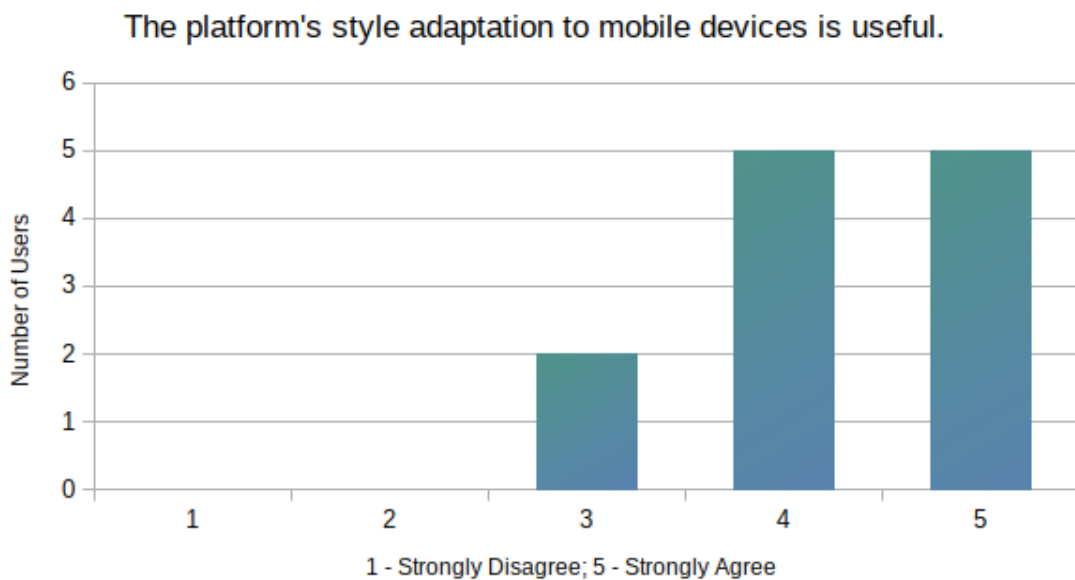


Figure 6.11: Statement 7: The platform's style adaptation to mobile devices is useful.

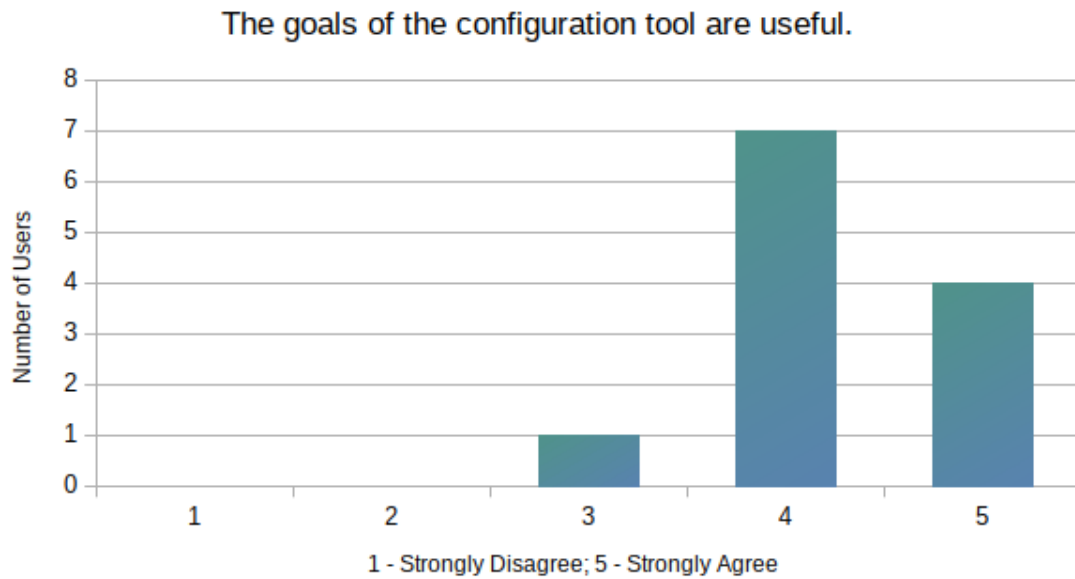


Figure 6.12: Statement 8: The goals of the configuration tool are useful.

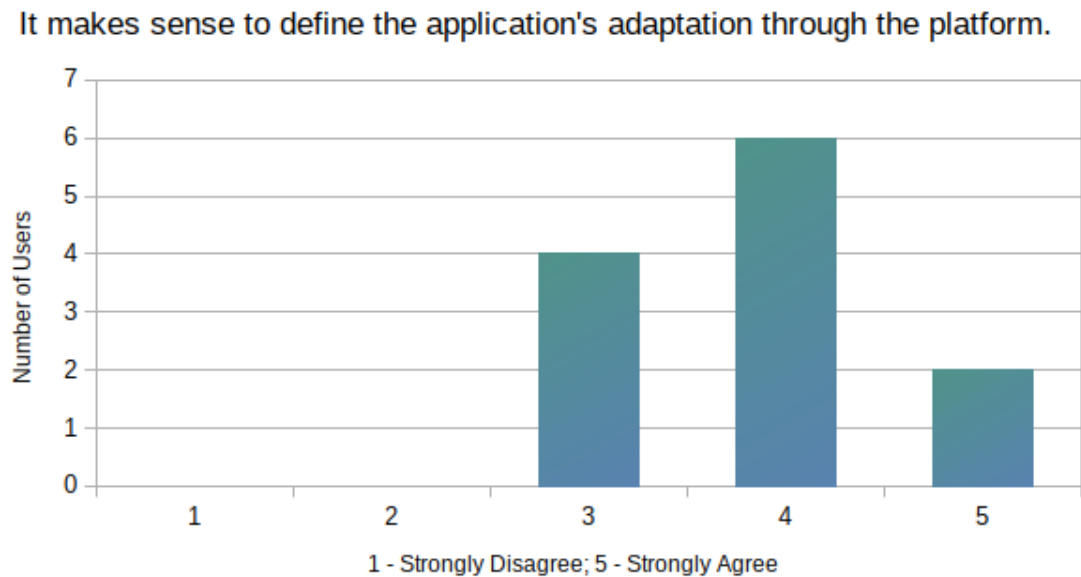


Figure 6.13: Statement 9: It makes sense to define the application's adaptation through the platform.

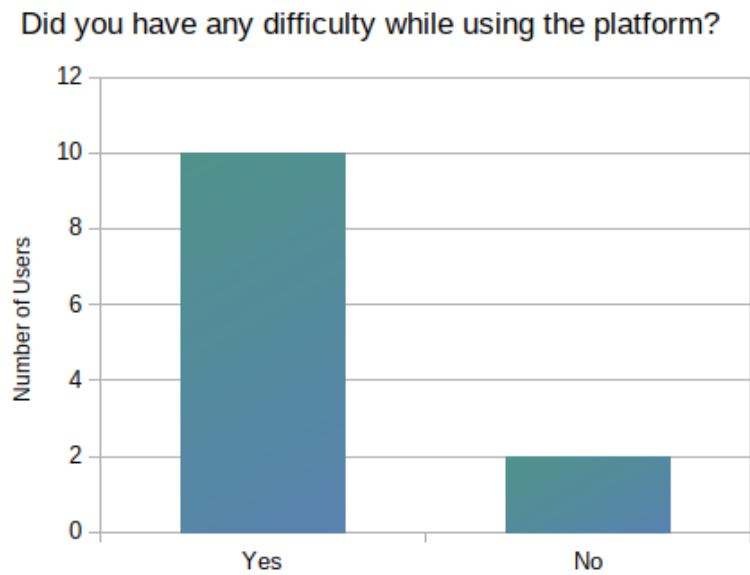


Figure 6.14: Question 2: Did you have any difficulty while using the platform?

Even if they ran into a problem with minor relevance, the participants were asked to answer positively to this question. Overall, the users considered that their difficulties were mainly related with understanding the difference between entities and entity attributes, hence having had some problems while configuring the first adaptation rules. Nevertheless, the participants seemed to perform the latter tasks effortlessly, which indicated that they had mostly understood the general process of rule creation.

6.3.2 Android Application

This subsection presents the formative assessment done to the Android application. It was evaluated by a total of thirteen participants, with different backgrounds - computer science, economy, biology and tourism. Out of these thirteen participants, nine also evaluated the configuration platform.

Figure 6.15 shows the results of the first question. This question asked about each participant's experience while using the application's interface, and only one participant (7.7%) remained neutral. The rest of the 12 participants were split into "Good" and "Very Good" - the first having 4 votes (30.8%) and the latter having 8 votes (61.5%). Compared to the configuration platform's assessment, this is a great improvement as, in this case, no participant provided a negative rating and the majority actually graded the application with the best rating. However, the Android application has a rather simple interface (even when not compared to the configuration platform) that automatically adapts itself, which does not require much input from the user. Hence, it is normal that these results are high - the users are not likely to face obstacles while using the application.

Figure 6.16 depicts the results regarding the statement "It is useful that the application adapts itself automatically according to what was specified in the configuration

How would you classify your experience while using the application's interface?

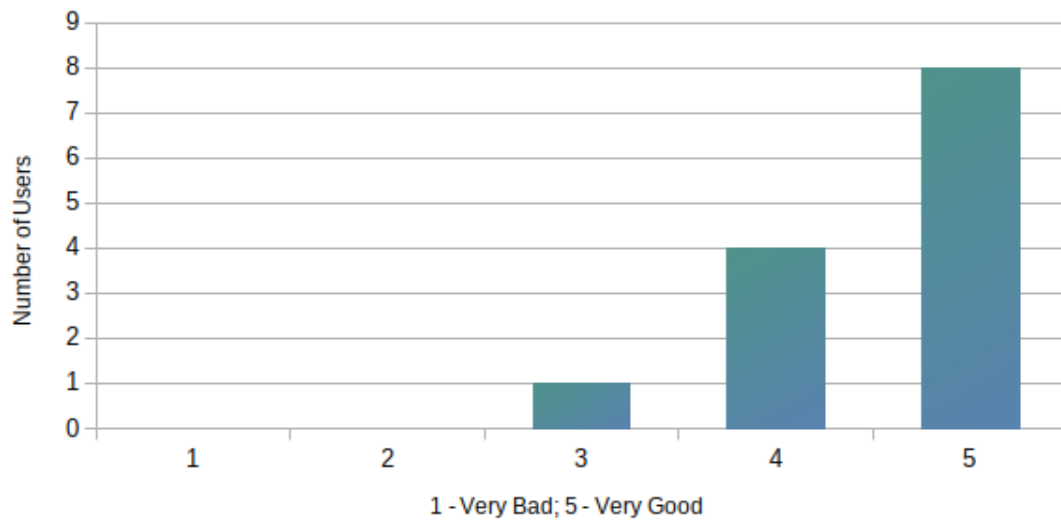


Figure 6.15: Question 1: How would you classify your experience while using the application's interface?

platform". As not all the participants had performed the configuration platform tasks, a brief explanation was provided to these users, where they were told that all the automatic adaptations they see in their screens were actually "programmed" in an external configuration tool. The results show that every participant found this feature useful - 7 of the users (53.8%) agreed with the statement, while the remaining 6 (46.2%) strongly agreed.

It is useful that the application adapts itself automatically according to what was specified in the configuration platform.

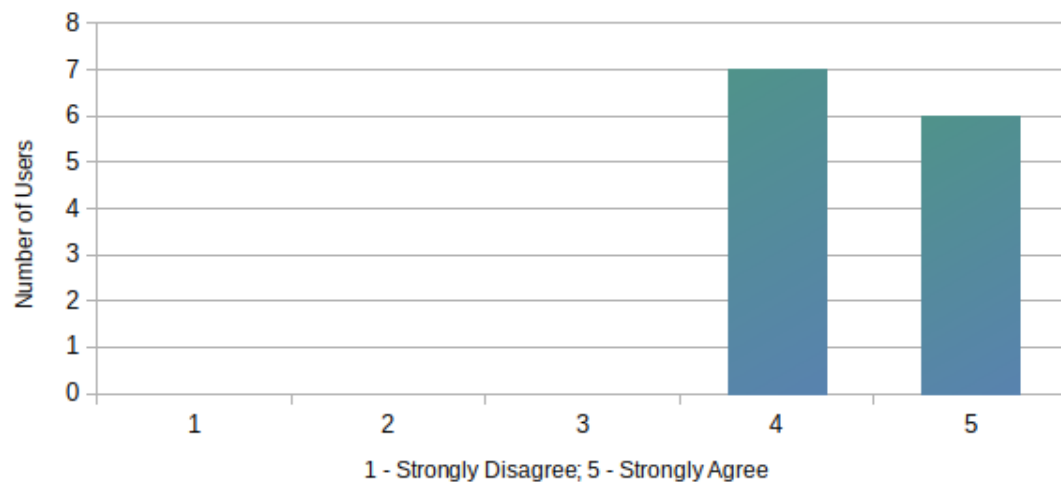


Figure 6.16: Statement 1: It is useful that the application adapts itself automatically according to what was specified in the configuration platform.

Regarding the option to turn off the automatic adaptation, only one participant (7.7%) had a hard time finding it, as depicted in Figure 6.17. Although this participant eventually found the option, it visibly took them a while to locate it. This can be explained by the fact that this participant (the second oldest from the participant pool) was not very familiar with smartphone applications, hence not finding it too obvious. Yet, 6 participants (46.2%) agreed that it was easy to find and 5 (38.5%) of them strongly agreed. Moreover, one other participant neutrally rated this statement, probably because they also had some difficulty finding the option. Nevertheless, it can be concluded that this option should be more visible, in order to help users unfamiliar with these applications.

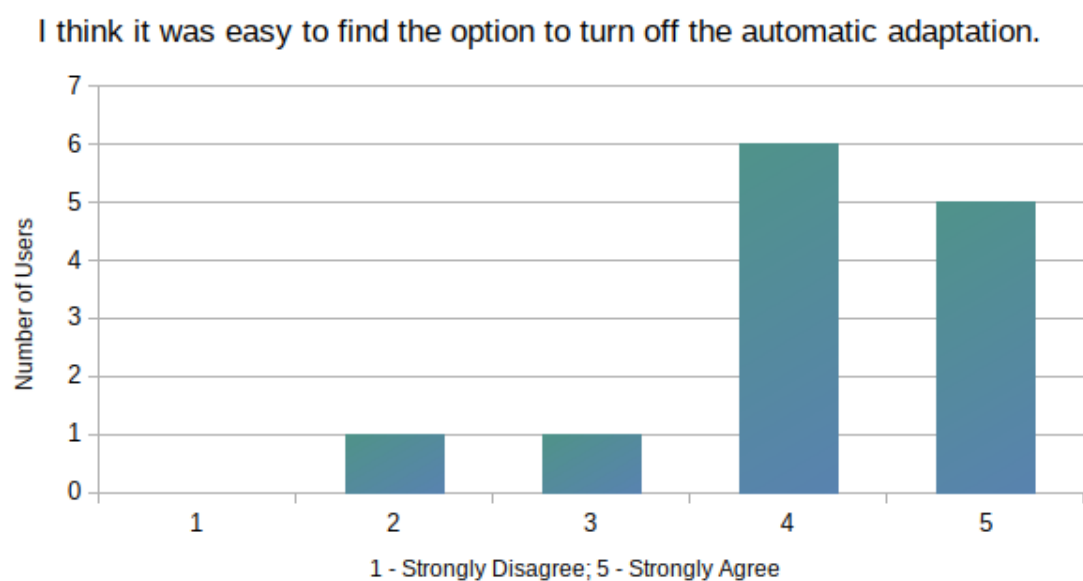


Figure 6.17: Statement 2: I think it was easy to find the option to turn off the automatic adaptation.

Figures 6.18, 6.19 and 6.20 depict the results of the participant answers regarding the proximity of the participants to the points of interest. The first Figure shows that the vast majority of the participants - 11 (84.6%) - noticed that notifications were showing up whenever they got close to a point of interest. However, two users disagreed with this statement. Later in the suggestion box, one of the participants mentioned they did not notice the notifications when riding a car due to external distractions. This suggests that auditory notifications would have been beneficial when getting close to a point of interest, as will be mentioned later in the future work section. Nevertheless, the text notifications sufficed for most users. The second Figure suggests that opening the closest point of interest's InfoWindow is useful, as only one participant (7.7%) did not positively grade the statement. The remaining 12 (92.3%) participants were evenly split between agreeing and strongly agreeing with the statement. Finally, the third Figure shows that there is a consensus on the statement regarding the usefulness of telling the user they are

getting near a point of interest, even if it is not the closest - 10 participants strongly agreed with the statement (76.9%), while the remaining three agreed. Overall, the notification system seems to be consistent among the users, who generally enjoy knowing the extra information.

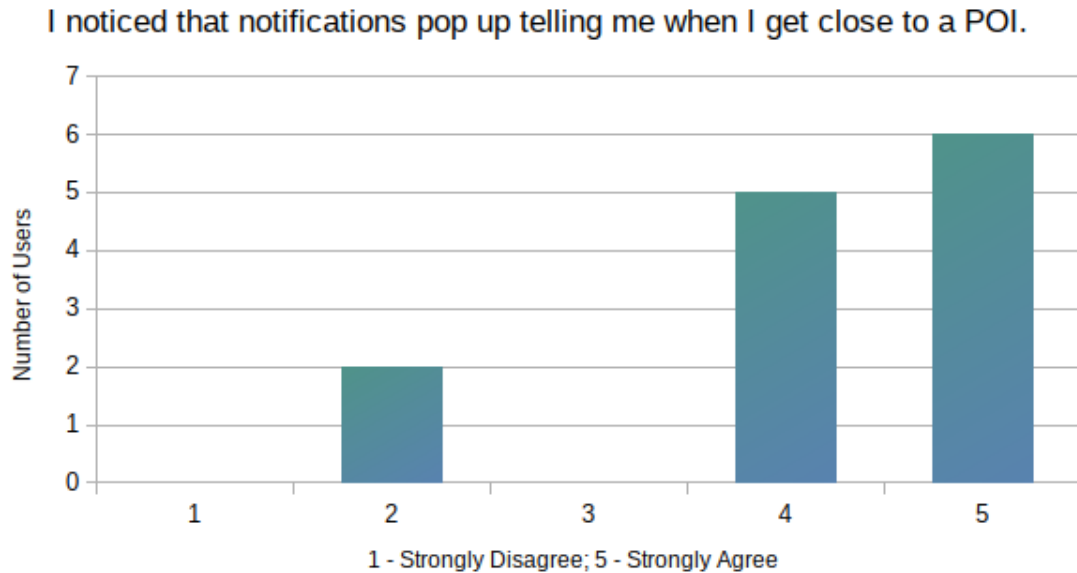


Figure 6.18: Statement 3: I noticed that notifications pop up telling me when I get close to a POI.

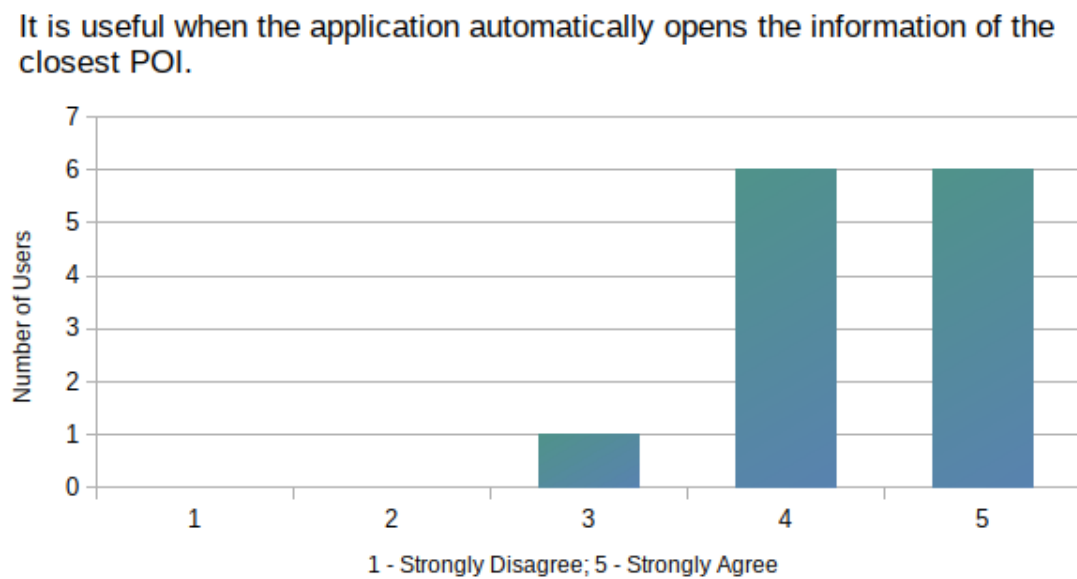


Figure 6.19: Statement 4: It is useful when the application automatically opens the information of the closest POI.

It is useful when the application tells me I am getting near a new POI, even if it is not the closest.

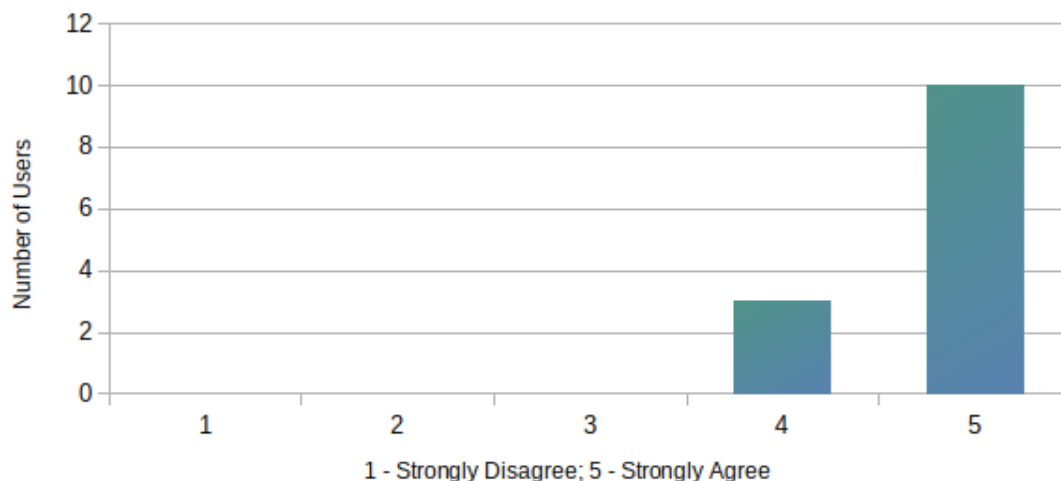


Figure 6.20: Statement 5: It is useful when the application tells me I am getting near a new POI, even if it is not the closest.

Figures 6.21 and 6.22 had two less participants. As previously mentioned, two of the Android application users were driving while performing the car tasks. Hence, they were not able to assess this part of the tasks. Nevertheless, out of the 11 users grading those two statements, six (54.5%) agreed that the application’s ability to adapt the proximity distance according to the movement speed was useful, while 4 (36.4%) strongly agreed. However, the answers regarding the second statement were not as consensual. When asked if the proximity distance while riding a car was adequate, 3 users (27.3%) disagreed, while 4 (36.4%) agreed. Similarly, two users stayed neutral and two others strongly agreed. The results came out as unexpected due to the fact that the speeds defined in the adaptation rules were previously tested and chosen as “ideal”. However, as figure 6.22 shows, further refinements are required.

Furthermore, the walking proximity distance is generally easier to model. While driving, users may have speeds ranging from 0 km/h to over 200 km/h. However, when walking, the movement speed is much lower and limited. Most likely due to this fact, the results shown in Figure 6.23 are quite positive - 4 users (30.8%) agreed with the proximity distance while walking being adequate, while the rest of the users strongly agreed.

Moreover, the statement regarding the usefulness of centring the map on the user solely when they are driving a car was controversial. While most participants agreed with the statement (5 (38.5%) agreed and 3 (23.1%) strongly agreed), the remaining 5 users did not provide positive answers. While one of these five (7.7%) stayed neutral, the other four participants provided negative answers (1 strongly disagreed and the remaining 3 disagreed). Nevertheless, it can be concluded that most people prefer having the ability to

It is useful that the application is able to change the proximity distance according to the movement speed.

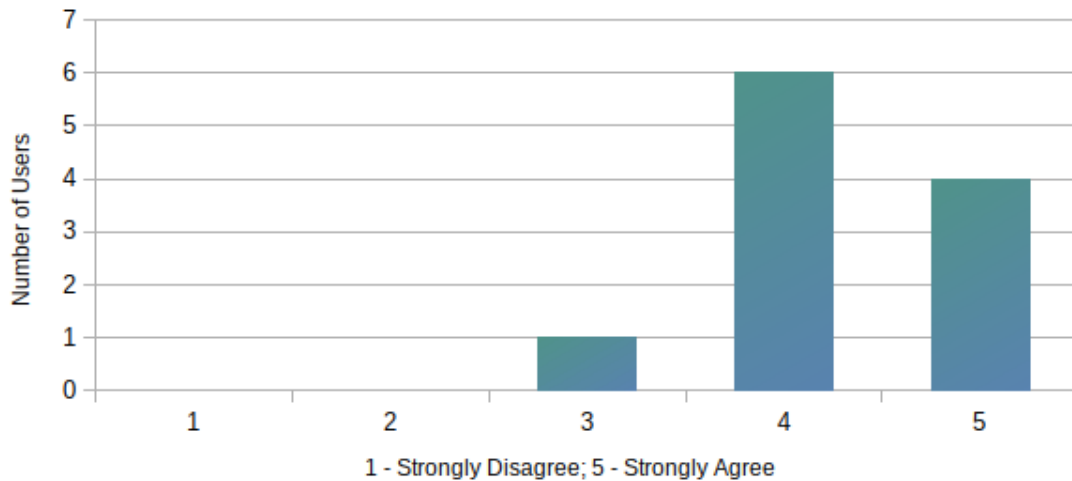


Figure 6.21: Statement 6: It is useful that the application is able to change the proximity distance according to the movement speed.

I felt that the proximity distance while riding a car was adequate.

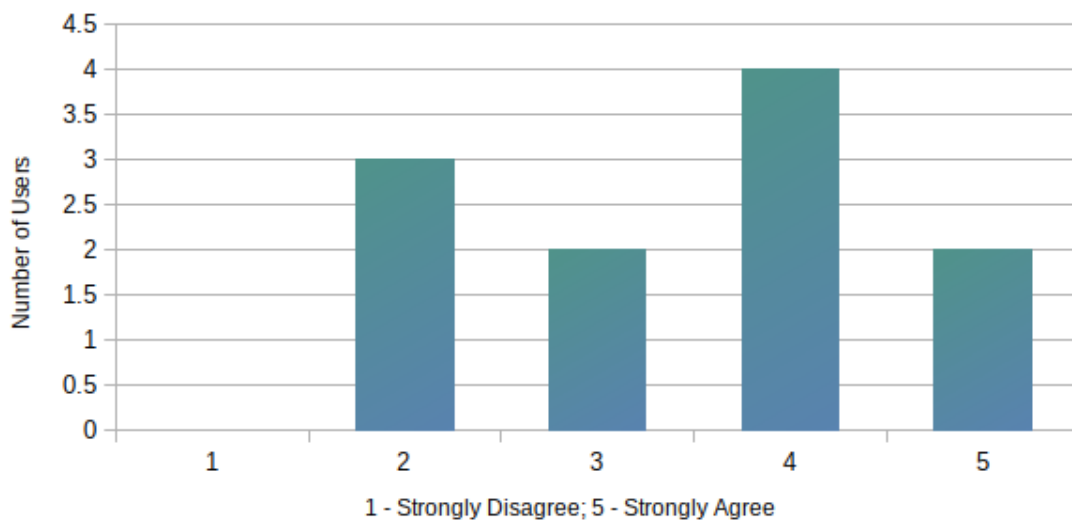


Figure 6.22: Statement 7: I felt that the proximity distance when riding a car was adequate.

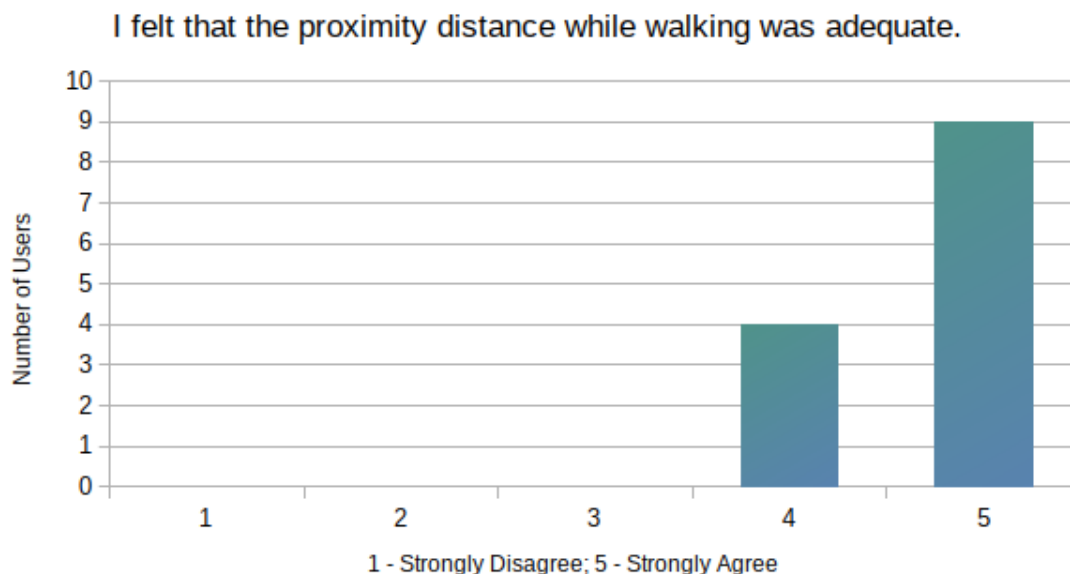


Figure 6.23: Statement 8: I felt that the proximity distance while walking was adequate.

navigate the map instead of having the application automatically and constantly centring it on their position.

The fact that the map solely centred itself on the user when driving a car is useful.

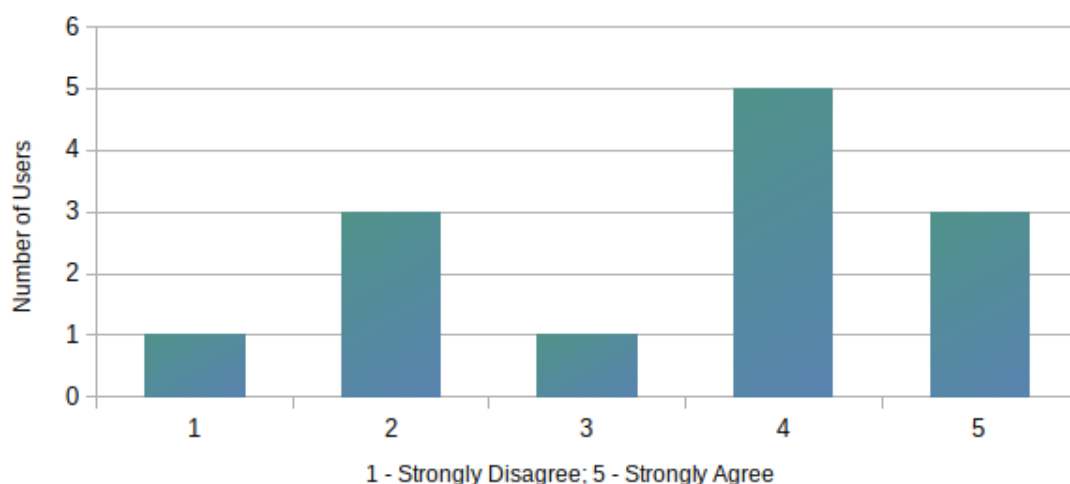


Figure 6.24: Statement 9: The fact that the map solely centred itself on the user when driving a car is useful.

Figure 6.25 shows that two users (15.4%) preferred using the application when the automatic adaptation was turned off (as all participants were asked to do during part of the tour). Nevertheless, all the other participants disagreed with the statement to some

extent - 6 of them (46.2%) strongly disagreed and 5 disagreed (38.5%) - which in this case indicates a positive outcome.

I enjoyed the application better when the automatic adaptation was turned off.

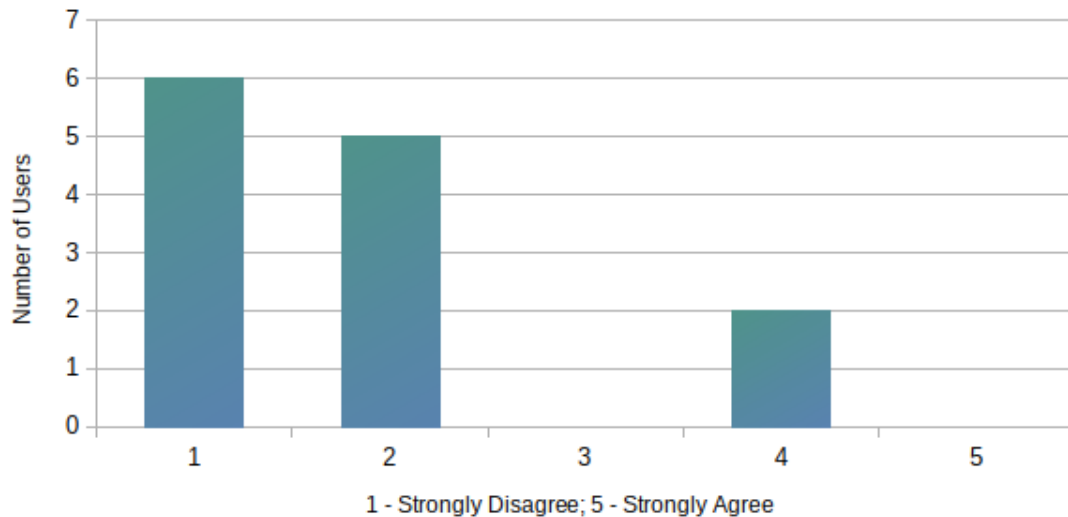


Figure 6.25: Statement 10: I enjoyed the application better when the automatic adaptation was turned off.

Finally, Figure 6.26 depicts the results of the last questionnaire question - “Did you have any difficulty while using the application?”. Most users answered with “No” (9 participants (69.2%)), while the rest of the participants answered with “Yes”. When asked what difficulties these participants faced, the answers varied - one of them mentioned the map taking a long time to load (this seemed to be a problem with his phone and/or internet connectivity), while two others described having problems closing the points of interest information (because the application keeps opening it). However, this problem could have been solved if they had turned off the automatic adaptations, which they were not supposed to do while performing part of the tasks. Overall, the Android application evaluation returned positive results. Nevertheless, some bad results must be taken into account in order to improve this application.

6.4 Conclusion

The main goals of the evaluation process - retrieving qualitative feedback from users in order to improve the system and its validation - have been achieved.

Although the total number of participants - 16 - who tested the system (or part of it) is not very high, the results show that, in general, the users are satisfied with both parts of this dissertation’s system. The configuration platform performed worse than the Android application in most of the tests, including SUS. However, its mean SUS score is still above the acceptable level of 68, when referring to web interfaces. Furthermore, the usability

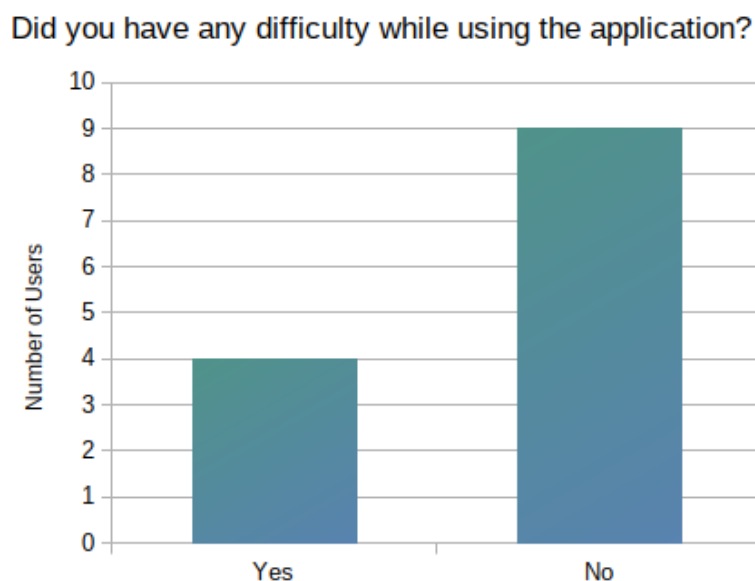


Figure 6.26: Question 2: Did you have any difficulty while using the application?

and learnability components are also above this acceptable level (although the latter is above by just 1.5 points). The Android application excelled in the SUS test, having had a score above the acceptable level of 66. Its learnability and usability scores were also way higher than 66.

Finally, the feedback returned by the participants was very important, not only for detecting problems in the system and how to solve them, but also for pointing out ways to generally improve it. Early informal assessments were also provided by users who expressed their difficulties while using the configuration platform and Android application. Both the optional description and POI closeness notifications derived from the participant's feedback, revealing these suggestions were crucial for improving the system. Furthermore, the fact that only participants having a computer science background performed the configuration platform tasks helped us understand how a knowledgeable administrator would interact with the configuration tool, during their first interaction. It further revealed that, at first, administrators will need some kind of introduction to the tool so they can learn the core concepts, but will rapidly be able to create meaningful adaptation rules once they start experimenting with the tool. Likewise, the application feedback revealed important information. It mainly concerns the proximity distances refinement and map centring, however, no setting will please all the user tastes. Hence, it most likely makes sense to have individual rules for each user (as will be suggested in the future work chapter) or, instead, allowing users to personalise the received rules according to their taste.

CONCLUSIONS AND FUTURE WORK

This chapter summarises what was accomplished throughout the dissertation, as well as what was achieved by its outcome. Finally, some suggestions about what could be added to the system in order to improve the developed solution are presented.

7.1 Conclusions

Context-based adaptations are a great way of improving the user's experience when using a GIS application. Hence, this dissertation focused on developing a tool which is able to model a user's context in a short and precise way, using it to adapt the application.

Thus, this work presents a web-based solution where knowledgeable administrators are able to interactively define how a certain, external application will adapt itself when under certain conditions (specified by these administrators). This solution aimed to provide a generic method for defining adaptation and it can be potentially applied to several types of WebGIS application, based on the developed context-based adaptation model presented in Chapter 4. Hence, the configuration platform stores the adaptation rules in a database that should be queried by the external application.

A mobile tourism application was developed as proof of concept to be supported by the configuration platform. Likewise, the application contains a highly generic way of operation - by itself, the application supports no adaptation, however, if the knowledgeable administrators create adaptation rules, the mobile application will adapt itself according to what was specified. This generic model periodically evaluates its state and, if a certain context aspect matches what the administrators specified, an automatic adaptation (also administrator-defined) will be activated. Furthermore, the tourism application retrieves its points of interest from a database located in another external server.

Moreover, in order to improve the user experience as a tourist, the application periodically checks the device's sensors for information, namely the user's location, speed and others. Having received the adaptations designed by the knowledgeable administrators (in the configuration platform), the application evaluates which actions should be automatically activated in every given moment in order to improve the tourist's user experience.

The configuration platform revealed to require some leaning at first by new administrators, but they learned to correctly use it rapidly. On the other hand, the mobile application performed very well during the evaluation stage due to the automatic adaptations modelled in the platform - users tended to not need to perform many actions on their phones, as the application automatically tried to suit their needs.

The evaluation process featured 16 total participants, where 12 of them performed the configuration platform tasks and 13 of them performed the Android application activities. These 16 participants allowed us to understand how administrators and users would perceive the system at first glance and make small adjustments in order to improve their receptivity. Most automatic adaptations please the majority of the users, however, proximity-based adaptations proved to be a little more sensitive, as their tastes were very different.

Summarising, the developed system provides a way to model automatic adaptations outside of the main application. This way, new adaptations can be easily added (or old ones removed), without having to change the application's code, through a simple web interface. These modifications are made without needing to update the external application - it will instead just retrieve the new rules and automatically apply them. The configuration platform was able to successfully model an Android application's adaptations according to the user's context, which consequentially confirmed the system's viability.

7.2 Future Work

As previously mentioned, during the evaluation stage, we found that some rules did not fully please most users. Thus, it would make sense to allow the users to manually tune their preferences, instead of having one set of adaptation rules applied to the application. However, this feature would require one of two approaches: either these preferences are stored locally, in each device; or users would have to register an account, and each account would include their own set of rules. The first approach seems more desirable as it takes less space on the platform's database. However, users would need to understand this dissertation's concepts.

Likewise, each user has their own concept of point of interest. Thus, it makes sense to allow users to define and manage their own points of interest. The application should locally save their POIs in order to avoid overloading the points of interest server. However,

rules specific to certain points of interest would not apply to the user’s selection, unless the rule is directed toward “All POIs”.

The priority system is used to solve rule conflicts, when they contain triggers with overlapping conditions (see section 5.3.2.1). This problem could be avoided if the administrators were able to, instead of defining numeric values, define mathematical expressions. For example, they could define that the map’s zoom is $Zoom = 15 - \frac{Speed}{30}$. This way, the map’s zoom would be automatically calculated for each speed measurement, without needing to define priorities. An alternative way to avoid setting rule priorities would be to define trigger intervals (e.g. $0 \leq Speed \leq 30 \Rightarrow Zoom = 10$), as suggested by an assessment participant.

In order to make this application usable while driving a vehicle, textual notifications must be accompanied by (or replaced) auditory notifications. Hence, the Android’s TextToSpeech library¹ is a great candidate for this purpose. This feature should also be modelled in the configuration platform (by creating a new entity attribute called “TTS” on the “Application” entity), which would allow the auditory behaviour to be modelled through automatic adaptations. This feature is expected to highly increase the user’s experience.

Furthermore, it is important to provide a mechanism for reversing actions taken. In other words, when a trigger condition is no longer evaluated as “true”, it may make sense to undo the previous action. In order to implement this feature, the Android application would need to keep in memory the list of rules that have been activated, but not deactivated. During the application runtime this list would need to be periodically checked and, when a rule was deactivated, the opposite action would be taken and the rule would be removed from this list. This feature would also need to keep track of the previous state of each attribute before the action was activated, making it possible to reset the attribute to a previous value.

As noticed during the evaluation stage (Chapter 6), users did not generally agree that only centring the map on the user’s location when walking was positive. Thus, it makes sense to add a configuration option in the Android application which allows users to disable certain aspects of automatic adaptation, instead of disabling the adaptations altogether.

Finally, an authentication page should also be provided in the configuration platform. Although the platform’s security was not a concern during the system development, users should not be able to access it. In order to achieve this, the configuration should only be accessible through HTTPS. Then, a new collection should be created, for storing a SHA-512 hash of the chosen password (this way, even if the document were to be stolen, the password would remain unknown). Then, the platform should include a new endpoint for authentication. If a correct password was provided, the server would return an authorisation token to be included in the following server requests. The endpoints

¹<https://developer.android.com/reference/android/speech/tts/TextToSpeech>

pertaining information updates or insertions on rules, entities, initial settings and POI server URI would then only be accessible if the request contained the authorisation token previously provided, or, in other words, personnel who know what the password is.

BIBLIOGRAPHY

- [1] R. Alnanih, O. Ormandjieva, and T. Radhakrishnan. “Context-based and Rule-based Adaptation of Mobile User Interfaces in mHealth.” In: *Procedia Computer Science* 21 (2013). The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2013) and the 3rd International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH), pp. 390–397. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2013.09.051>. URL: <http://www.sciencedirect.com/science/article/pii/S1877050913008442>.
- [2] ArcGIS. *Datums - Help*. 2016. URL: <https://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/datums.htm> (visited on 05/15/2018).
- [3] ArcGIS. *Geocentric Coordinate System*. 2016. URL: <https://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/geocentric-coordinate-system.htm> (visited on 05/15/2018).
- [4] ArcGIS. *Coordinate systems, projections, and transformations - Properties of Maps*. 2017. URL: <https://pro.arcgis.com/en/pro-app/help/mapping/properties/coordinate-systems-and-projections.htm> (visited on 05/15/2018).
- [5] A. Bangor, P. Kortum, and J. Miller. “Determining what individual SUS scores mean: Adding an adjective rating scale.” In: *Journal of usability studies* 4.3 (2009), pp. 114–123.
- [6] J. Brooke. “SUS: a retrospective.” In: *Journal of usability studies* 8.2 (2013), pp. 29–40.
- [7] J. Brooke et al. “SUS-A quick and dirty usability scale.” In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7.
- [8] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. en. URL: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html> (visited on 06/29/2018).
- [9] L. M. Cândido Antunes. “Detecção Automática de Atividades (Aplicação à Agronomia).” pt. Master’s thesis. Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa, Sept. 2015.

- [10] A. K. Dey. "Understanding and Using Context." In: *Personal Ubiquitous Comput.* 5.1 (Jan. 2001), pp. 4–7. ISSN: 1617-4909. DOI: [10.1007/s007790170019](https://doi.org/10.1007/s007790170019).
- [11] S. Di Martino, F. Ferrucci, G. McArdle, and G. Petillo. "Automatic Generation of an Adaptive WebGIS." In: *Web and Wireless Geographical Information Systems*. Ed. by J. D. Carswell, A. S. Fotheringham, and G. McArdle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 171–186. ISBN: 978-3-642-10601-9.
- [12] C. of Earth and T. P.S. U. Mineral Sciences. *The Nature of Geographic Information - Coordinate Systems*. 2018. URL: https://www.e-education.psu.edu/natureofgeoinfo/c2_p10.html (visited on 05/15/2018).
- [13] ESRI. *What is a Coordinate System?* 2005. URL: http://edndoc.esri.com/arcscde/9.1/general_topics/what_coord_sys.htm (visited on 05/15/2018).
- [14] ESRI. *Coordinate systems, map projections, and geographic (datum) transformations*. 2010. URL: <http://resources.esri.com/help/9.3/arcgisengine/dotnet/89b720a5-7339-44b0-8b58-0f5bf2843393.htm> (visited on 05/15/2018).
- [15] J. C. Ferreira, H. Silva, J. A. Afonso, and J. L. Afonso. *An android-based personalized public transportation advisor*. 2017.
- [16] P. Fu and J. Sun. *Web GIS: Principles and Applications*. Esri Press, 2010. ISBN: 158948245X, 9781589482456.
- [17] G. Ghiani, M. Manca, F. Paternò, and C. Porta. "Beyond Responsive Design: Context-Dependent Multimodal Augmentation of Web Applications." In: *Mobile Web Information Systems*. Ed. by I. Awan, M. Younas, X. Franch, and C. Quer. Cham: Springer International Publishing, 2014, pp. 71–85. ISBN: 978-3-319-10359-4.
- [18] G. Ghiani, M. Manca, and F. Paternò. "Authoring Context-dependent Cross-device User Interfaces Based on Trigger/Action Rules." In: *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. MUM '15. ACM, 2015, pp. 313–322. ISBN: 978-1-4503-3605-5. DOI: [10.1145/2836041.2836073](https://doi.org/10.1145/2836041.2836073). URL: <http://doi.acm.org/10.1145/2836041.2836073>.
- [19] G. Ghiani, M. Manca, and F. Paternò. "Dynamic user interface adaptation driven by physiological parameters to support learning." en. In: ACM Press, 2015, p. 6. ISBN: 978-1-4503-3646-8. DOI: [10.1145/2774225.2775081](https://doi.org/10.1145/2774225.2775081). URL: <http://dl.acm.org/citation.cfm?doid=2774225.2775081> (visited on 06/04/2018).
- [20] G. Ghiani, M. Manca, F. Paternò, and C. Santoro. "Personalization of Context-Dependent Applications Through Trigger-Action Rules." en. In: *ACM Transactions on Computer-Human Interaction* 24.2 (Apr. 2017), p. 33. ISSN: 10730516. DOI: [10.1145/3057861](https://doi.org/10.1145/3057861). URL: <http://dl.acm.org/citation.cfm?doid=3077620.3057861> (visited on 06/04/2018).

- [21] R. Henrique Bizarra. “LiveTeams: Gestão de Emergência online adaptativa pelo contexto.” pt. Master’s thesis. Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa, Mar. 2016.
- [22] W. Höpken, M. Fuchs, M. Zanker, and T. Beer. “Context-Based Adaptation of Mobile Applications in Tourism.” In: *Information Technology & Tourism* 12.2 (2010), pp. 175–195. ISSN: 1098-3058. DOI: doi : 10.3727/109830510X12887971002783. URL: <https://www.ingentaconnect.com/content/cog/itt/2010/00000012/00000002/art00006>.
- [23] R. Knippers. *Coordinate systems*. 2016. URL: <http://kartoweb.itc.nl/geometrics/Coordinate%20systems/coordsys.html> (visited on 05/15/2018).
- [24] J. R. Lewis and J. Sauro. “The factor structure of the system usability scale.” In: *International conference on human centered design*. Springer. 2009, pp. 94–103.
- [25] P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind. *Geographic Information Science and Systems*. 4th. Wiley Publishing, 2015. ISBN: 1118676955, 9781118676950.
- [26] G. Lucci and F. Paternò. “Understanding End-User Development of Context-Dependent Applications in Smartphones.” en. In: *Human-Centered Software Engineering*. Ed. by S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, and M. Winckler. Vol. 8742. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, p. 17. ISBN: 978-3-662-44810-6 978-3-662-44811-3. DOI: 10.1007/978-3-662-44811-3_11. URL: http://link.springer.com/10.1007/978-3-662-44811-3_11 (visited on 06/04/2018).
- [27] A. F. Maia. “Plataforma para análise de Estruturas Costeiras.” pt. Master’s thesis. Faculdade de Ciências e Tecnologia / Universidade Nova de Lisboa, Dec. 2016.
- [28] D. Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2013. ISBN: 9780465072996. URL: <https://books.google.pt/books?id=nVQPAAAAQBAJ>.
- [29] T. Reichenbacher. “Mobile Usage and Adaptive Visualization.” In: *Encyclopedia of GIS*. Ed. by S. Shekhar, H. Xiong, and X. Zhou. Cham: Springer International Publishing, 2015, pp. 1–7. ISBN: 978-3-319-23519-6. DOI: 10.1007/978-3-319-23519-6_799-2. URL: https://doi.org/10.1007/978-3-319-23519-6_799-2.
- [30] L. T. Sarjakoski and T. Sarjakoski. “User Interfaces and Adaptive Maps.” In: *Encyclopedia of GIS*. Boston, MA: Springer US, 2008, pp. 1205–1212. ISBN: 978-0-387-35973-1. DOI: 10.1007/978-0-387-35973-1_1431. URL: https://doi.org/10.1007/978-0-387-35973-1_1431.
- [31] Statista. *Number of smartphone users worldwide from 2014 to 2020 (in billions)*. 2016. URL: <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>.

BIBLIOGRAPHY

- [32] Wikimedia. *Gallery: Maps of the world, of seas and about history*. 2018. URL: https://commons.wikimedia.org/wiki/User:Sting/Gallery:_Maps_of_the_world,_of_seas_and_about_history (visited on 05/07/2018).
- [33] M. Worboys and M. Duckham. *GIS: A Computing Perspective*. 2nd. Boca Raton, FL, USA: CRC Press, Inc., 2004. ISBN: 0415283752.

APPENDIX



GIVEN TASKS

Teste de Usabilidade

Este teste tem como finalidade avaliar uma aplicação móvel que permite a visualização e exploração de informação geográfica relacionada com turismo, que se adapta automaticamente conforme diversas propriedades do contexto do utilizador – velocidade, localidade, tipo de movimento, etc. Contudo, o comportamento da aplicação face àquilo que é detectado pode ser configurado por um administrador, através de uma plataforma de configuração, sem ser necessário alterar qualquer código na aplicação Android. Assim, a aplicação é dirigida a todo o tipo de público, nomeadamente turistas, enquanto que a plataforma de configuração deverá ser manuseada por alguém com um bom conhecimento na área da informática e do sistema em geral.

Este teste consiste na realização de algumas tarefas num dispositivo móvel Android e num navegador *web* à escolha do participante. O teste iniciar-se-á no navegador *web*, realizando por fim algumas tarefas no dispositivo móvel. Antes da realização das tarefas, leia atentamente as Notas fornecidas na secção abaixo. É também importante que o seu raciocínio e acções sejam expostas em voz alta, de forma a obter um feedback maior acerca do teste e sistema. Caso lhe surja alguma dúvida, não hesite em contactar o supervisor.

Por fim, após a realização das tarefas, ser-lhe-á fornecido um questionário de satisfação associado ao sistema.

Notas

- A adaptação baseada no contexto é modelada com base em **regras**.
- Cada **regra** é composta por um **trigger** e uma **action**.
- Cada **trigger** é composto por uma **entity**, um **attribute**, um ou mais **values** e pode ainda conter um **comparison operator**. A estrutura das **actions** é semelhante, no entanto não contêm **comparison operators**.
- As **entities** são entidades que agrupam propriedades pertencentes ao contexto do utilizador e que podem ser alvo de configuração com o objetivo de adaptação, sempre que o contexto de utilização da aplicação se altera. O ambiente, o próprio utilizador, o dispositivo móvel o mapa da aplicação são exemplos de **entities**.
- Cada **entity** contém um ou mais **attributes** (propriedades mencionadas acima), que são elementos específicos de cada **entity**. Pegando no exemplo do ponto anterior, os **attributes** do ambiente podem ser a luminosidade, ruído, pressão atmosférica, etc, enquanto que os do dispositivo móvel podem incluir a bateria, o tipo de conexão à Internet, a qualidade da rede móvel, etc.
- Os **values** correspondem aos valores correspondentes a cada **attribute**.
- Um **comparison operator** é um operador de comparação (igual, menor ou igual, maior, etc) e serve para comparar o valor que é lido pela aplicação com o **value** incluído no **trigger**.

- Quando a condição expressa num **trigger** é verificada, é despoletado o comportamento descrito pela **action** pertencente à mesma regra.
- Um **POI** ou **Point of Interest** é um ponto de interesse na aplicação, com localização geográfica – pode ser, por exemplo, Torre de Belém, Castelo de São Jorge, etc.

Tarefas

Tarefa 1 – Configuração do comportamento da aplicação (~10 min)

Esta tarefa destina-se apenas a **utilizadores conhecedores da área de informática!** Se não for esse o seu caso, passe por favor à Tarefa 2.

1. Entre na plataforma de configuração. (<http://gseabra.herokuapp.com>)
2. Dirija-se à secção de listagem de regras e apague todas as regras existentes. (<http://gseabra.herokuapp.com/rules>)
3. Dirija-se à secção de criação de novas regras, disponível na barra lateral. (<http://gseabra.herokuapp.com/rules/new>)
4. Crie uma regra que verifique quando o **utilizador** se desloca com **velocidade menor ou igual** a **4 m/s** (~15 km/h) e altere a **distância de proximidade** para **50** metros, caso se verifique a condição.
5. Crie outra regra, que verifique quando o **utilizador** se desloca com **velocidade maior ou igual** a **22 m/s** (~80 km/h) e altere a **distância de proximidade** para **500** metros, caso se verifique a condição.
6. Crie outra regra, que verifique quando o **utilizador** se desloca com **velocidade maior ou igual** a **22 m/s** (~80 km/h) e altere o **centro do mapa** no **utilizador**, caso se verifique a condição.
7. Crie uma regra que verifique quando o **utilizador** se encontra **perto** de **todos os POIs** e **abra** a **InfoWindow** do **ponto mais próximo**, caso se verifique a condição.
8. Verifique que todas as regras foram inseridas correctamente, através do separador de listagem de regras. (<http://gseabra.herokuapp.com/rules>)

Tarefa 2 – Utilização da aplicação em auto-estrada (~15 min)

Esta tarefa deverá ser realizada enquanto **passageiro** de um veículo automóvel, na auto-estrada A5, sentido Lisboa Cascais.

1. Ao aproximar-se da zona de Oeiras, esteja atento ao mapa.

2. Esteja atento ao comportamento da aplicação ao mover-se a velocidades superiores a 80 km/h. Deverá ser possível verificar que o mapa é automaticamente centrado na sua posição actual.
3. Esteja atento ao comportamento da aplicação ao aproximar-se dos três pontos diferentes – SIC, Lagoas Park e Seminário Torre d'Águilha. Deverá ser possível observar que a aplicação abre, sem qualquer intervenção da sua parte, a informação disponível acerca de cada um destes pontos à medida que se desloca pela estrada.

Tarefa 3 – Utilização da aplicação a pé (~25 min)

Esta tarefa deverá ser realizada na vila de Cascais, percorrendo o percurso sugerido pelo supervisor.

1. Dirija-se ao ponto **Casa das Histórias Paula Rego**, mantendo-se atento à interface da aplicação.
2. Verifique qual o comportamento da aplicação, comparativamente àquele já observado na auto-estrada, durante a tarefa anterior. Deverá ser possível verificar que a aplicação já não centra o mapa automaticamente na sua posição actual.
3. Dirija-se aos pontos **Hotel Baía** e **Praia dos Pescadores** (próximos um do outro).
4. Verifique qual o comportamento da aplicação, comparativamente àquele já observado na auto-estrada, durante a tarefa anterior. Deverá ser possível observar que surgiram notificações de proximidade dos dois pontos.
5. Aproxime-se de um dos pontos.
6. Verifique que a informação acerca desse ponto foi automaticamente mostrada.
7. Aproxime-se do outro ponto.
8. Verifique que foi fechada a informação acerca do ponto anterior e aberta a do ponto do qual se encontra mais próximo.
9. Complete o percurso sugerido pelo supervisor, verificando as alterações que são efectuadas automaticamente (e verifique se existem mais alterações que acha que poderiam ser benéficas).

Tarefa 4.1 – Utilização da aplicação a pé, sem adaptação automática (~25 min)

Esta tarefa deverá ser realizada na vila de Cascais, percorrendo o percurso sugerido pelo supervisor.

1. Carregue nos três pontos no canto superior direito da aplicação.
2. Desligue as adaptações automáticas.
3. Efectue a **Tarefa 3** novamente.

Tarefa 4.2 – Utilização da aplicação em auto-estrada (~15 min)

Esta tarefa deverá ser realizada enquanto **passageiro** de um veículo automóvel, na auto-estrada A5, sentido Lisboa Cascais.

1. Carregue nos três pontos no canto superior direito da aplicação.
2. Verifique que as adaptações automáticas se encontram desligadas.
3. Efectue a **Tarefa 2** novamente.

Fim das Tarefas

Preencha agora o questionário referente à aplicação e às tarefas que acabou de realizar.

Obrigado pelo tempo disponibilizado!

APPENDIX



EVALUATION QUESTIONNAIRE

Improving Mobile GIS applications through the identification of Geographic Context – Questionnaire

O seguinte questionário é anónimo, pelo que os seus dados pessoais serão meramente utilizados para fins estatísticos.

Este questionário começa com questões de carácter pessoal e de seguida passará a perguntar acerca da satisfação sobre a plataforma de configuração e aplicação móvel, relacionadas com as tarefas que acabou de realizar.

Por favor seja o mais sincero possível de modo a se fazer uma melhor avaliação do trabalho desenvolvido.

The following questionnaire is anonymous and your personal data will only be used for statistical purposes. This questionnaire starts with a few personal questions and some questions about your satisfaction with the system will also be presented.

Please be as sincere as possible in order to better assess the developed work.

***Required**

1. Idade (Age): *

2. Sexo (Gender): *

Mark only one oval.

- Masculino (Male)
 Feminino (Female)

3. Área de formação (Area of expertise): *

Mark only one oval.

- Engenharia Informática (Computer Science)
 Other: _____

4. Realizou a tarefa relacionada com a plataforma de configuração? (Have you performed the tasks related with the configuration platform?) *

Mark only one oval.

- Sim (Yes)
 Não (No) *Skip to question 30.*

Plataforma de Configuração (Configuration Platform)

Esta secção do questionário destina-se à avaliação da Plataforma de Configuração. Se não tiver realizado a tarefa correspondente a esta plataforma, por favor passe para a avaliação da aplicação móvel.

This questionnaire section refers to the Configuration Platform. If you have not performed the respective tasks, please skip this section.

25. **Faz sentido permitir a configuração da adaptação da aplicação através da plataforma. (It makes sense to define the application's adaptation through the platform.) ***

Mark only one oval.

	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

26. **Experienciou alguma dificuldade no uso da plataforma de configuração? (Did you have any difficulty while using the platform?) ***

Mark only one oval.

- Sim (Yes)
 Não (No)

27. **Se respondeu que sim à pergunta anterior, por favor detalhe-a aqui (If you answered the previous question with "Yes", please detail it here):**

28. **Sugestões para melhorar a plataforma de configuração (Suggestions for improving the configuration platform):**

29. **Efetuiu tarefas relacionadas com a utilização da aplicação Android (Have you performed the tasks related with the Android application)? ***

Mark only one oval.

- Sim (Yes)
 Não (No) *Stop filling out this form.*

Aplicação móvel (Mobile application)

Esta secção do questionário destina-se à avaliação da aplicação móvel. Esta secção tem uma estrutura semelhante à anterior (caso tenha realizado a tarefa da plataforma de configuração), mas a importância das suas respostas é a mesma.

This questionnaire section refers to the evaluation of the mobile application. This section contains a structure similar to the previous one, but the importance of your answers is the same.

System Usability Scale

40. **No geral, classifico a facilidade de utilização da aplicação como (Overall, I would rate the user-friendliness of this application as): ***

Mark only one oval.

- Pior Imaginável (Worst Imaginable)
- Terrível (Awful)
- Fraca (Poor)
- OK
- Boa (Good)
- Excelente (Excellent)
- Melhor Imaginável (Best Imaginable)

Avaliação da usabilidade da interface da Aplicação Móvel (Evaluation of the Mobile Application's interface usability)

Esta secção contém algumas perguntas acerca da interface da aplicação móvel, que devem ser respondidas de forma semelhante às anteriores – escolher um valor numa escala entre 1 e 5 onde 1 é “Muito mau/Discordo totalmente” e 5 é “Muito bom/Concordo totalmente”.

This section contains some questions about the mobile application's interface, which should be answered in a similar way to the previous questions - picking a value on a scale between 1 and 5, where 1 is "Very bad/Totally disagree" and 5 is "Very good/Totally agree".

41. **Como classifica a experiência de utilização da interface da aplicação? (How would you classify your experience while using the application's interface?) ***

Mark only one oval.

1	2	3	4	5	
Muito má	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> Muito boa

42. **É útil a aplicação adaptar-se automaticamente conforme o que foi especificado na plataforma. (It is useful that the application adapts itself according to what was specified in the configuration platform.) ***

Mark only one oval.

1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> Concordo totalmente

43. **Achei fácil de encontrar a opção para desligar a adaptação automática. (I think it was easy to find the option to turn off the automatic adaptation.) ***

Mark only one oval.

1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/> Concordo totalmente

50. **Achei útil o mapa ser centrado no utilizador apenas quando se conduzia. (The fact that the map solely centred itself on the user when driving a car is useful.)**

Mark only one oval.

	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

51. **Gostei mais da experiência de utilização da aplicação com a adaptação automática desligada. (I enjoyed the application better when the automatic adaptation was turned off.)**

*

Mark only one oval.

	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

52. **Gostei mais da experiência de utilização da aplicação com a adaptação automática desligada. (I enjoyed the application better when the automatic adaptation was turned off.)**

*

Mark only one oval.

	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Concordo totalmente

53. **Experienciou alguma dificuldade no uso da aplicação? (Did you have any difficulty while using the application?)** *

Mark only one oval.

- Sim (Yes)
 Não (No)

54. **Se respondeu que sim à pergunta anterior, por favor detalhe-a aqui (If your previous answer was "Yes", please detail it here):**

55. **Sugestões para melhorar a aplicação (Suggestions for improving the application):**



USER PROFILES AND INFORMATION

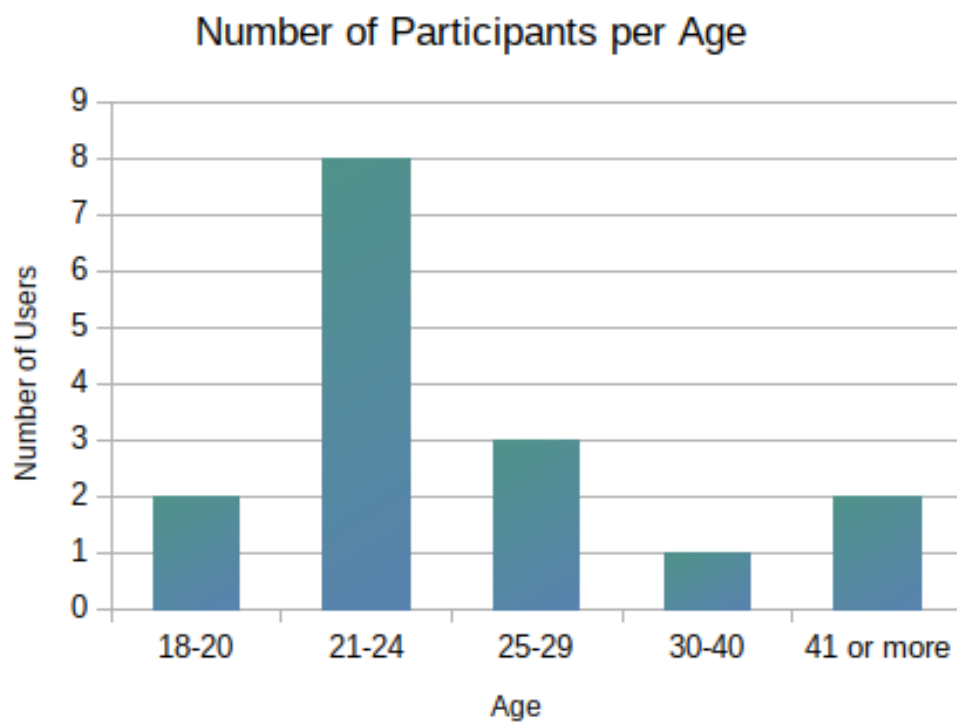


Figure C.1: Participants age.

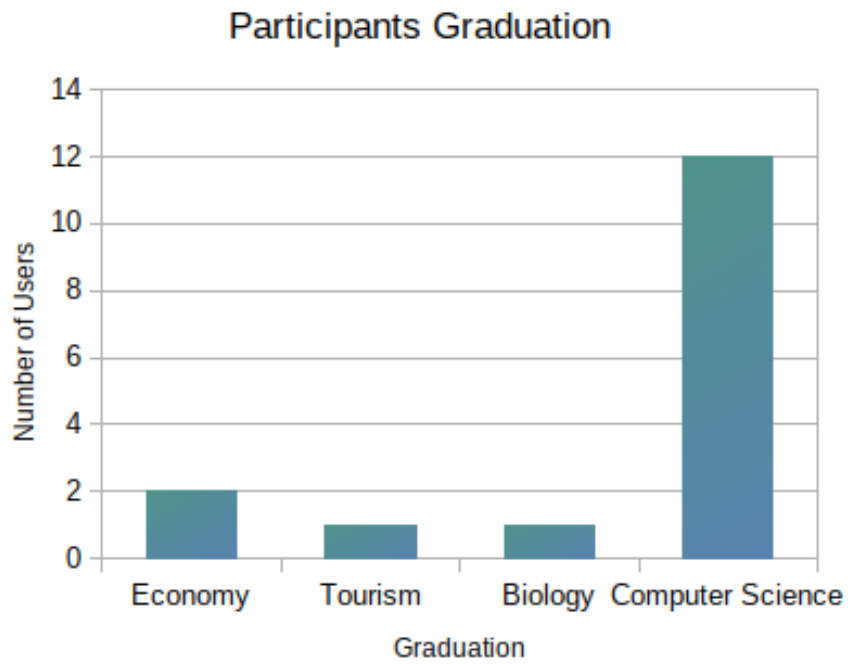


Figure C.2: Participants graduation.

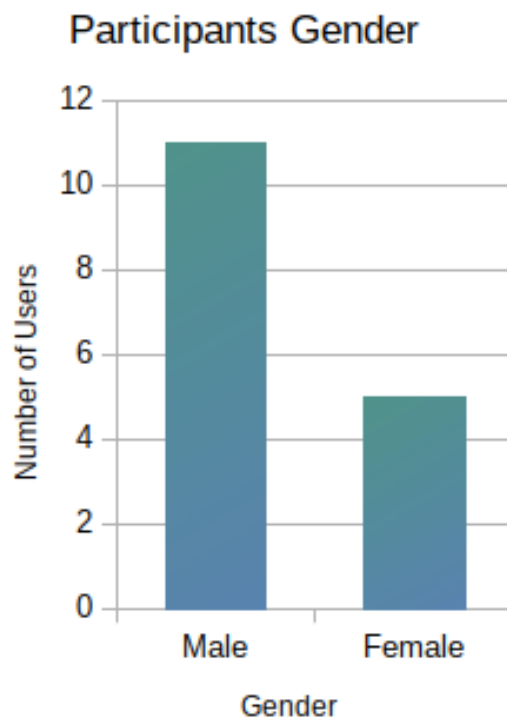


Figure C.3: Participants gender.