Geo Crowd-Surveys

André Coelho Lopes

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. João Nuno De Oliveira e Silva
Dr. Ana Mafalda Marques Carapuço

Examination Committee
Chairperson: Prof. António Manuel Raminhos Cordeiro Grilo
Supervisor: Prof. João Nuno De Oliveira e Silva
Members of the Committee: Prof. João Pedro Faria Mendonça Barreto

November 2017
Acknowledgements

Obrigado ao Prof. João Nuno Silva por me ter suscitado interesse nesta área, com a cadeira de Arquitectura de Sistemas de Internet. Agradeço ter-me dado esta oportunidade para aprofundar o meu conhecimento e trabalhar neste projecto, e por me ter apoiado sempre, ao longo do trabalho.

Obrigado à Dra. Mafalda Carapuço pela orientação importante para conhecer as necessidades e circunstâncias reais de quem trabalha no terreno, e as ideias de *crowdsourcing* e *surveying* para solucionar problemas actuais.

Obrigado à minha família, em especial aos meus pais. Tanto nos meus piores como nos meus melhores momentos sempre me apoiaram para tentar alcançar os meus sonhos.

Os meus amigos ao longo do curso também fizeram com que este caminho deixasse muitas memórias positivas. Agradeço a todos, e quero deixar um abraço especial ao Francisco Pires e ao João Maurício.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.
Abstract

Geocrowdsourcing is a new way for institutions to obtain data from the crowd. It makes it possible to harness an effort from the crowd, in order to satisfy a need for information, in which geographic information (locations on the map) plays a key role. Surveys have been, and still are, important means for institutions to gather information from people. It is common for surveyors to not have direct personal access to their target population, and surveyors need innovative ways to reach their audiences. There might be a geographical component, and a good solution needs to make sure this isn't a significant burden on participants. If the surveyor needs media like photographs and other digital files, this should also be attainable.

Nowadays, devices like Smartphones and Tablet computers have become widespread and popular and can be applied to surveying as well. Surveying can also be made simpler, and an easy task for participants to accomplish using smartphone devices.

The implemented system demonstrates several functionalities that when combined provide a useful and cohesive tool for surveyors. It attempts to solve their whole surveying process, from survey creation, configuration and deployment on a Web platform, with usage of geographical tools, to the management of participants and submission/export of survey answers. This is shown to be possible, along with striving for friendly and easy to use user interfaces that make the participant's task straightforward and simple.

Keywords
Survey; Crowdsourcing; Mobility; Geotagging; Forms
Resumo

Geocrowdsourcing é uma nova maneira para as instituições obterem dados através da multidão. Torna possível gerir um esforço da multidão com o fim de satisfazer necessidades de informação, na qual informações geográficas (localizações no mapa) têm um papel chave. Os inquéritos têm sido meios importantens para as instituições obterem informação junto das pessoas. É comum que quem faz o inquérito não tenha acesso directo à população-alvo, e para isso são necessários métodos inovadores para alcançar esse público. Se o inquérito tiver alguma componente geográfica, uma boa solução deve garantir que esse factor não implicará esforço acrescido por parte da população que participa. Caso sejam precisas fotografias ou outros tipos de ficheiros digitais, também deve ser possível integrar esses tipos de campos num inquérito.

Hoje em dia, aparelhos como Smartphones e computadores Tablet tornaram-se generalizados e populares, podendo também ser aplicados a inquéritos. É possível criar situações mais fáceis, e ter inquéritos que apenas requeiram simples tarefas num smartphone para que a multidão participe.

O sistema implementado demonstra várias funcionalidades, que quando combinadas fornecem uma ferramenta útil e coesa para quem faz inquéritos. Tenta resolver todo o processo de um inquérito, desde a sua criação, configuração e lançamento numa plataforma Web, com utilização de ferramentas geográficas, até à gestão de participantes e submissão/exportação de respostas. É demonstrado que isto é possível, com a tentativa de alcançar interfaces de utilizador fáceis de utilizar, que tornem a tarefa de quem responde direta e simples.

Palavras Chave
Inquérito; Crowdsourcing; Mobilidade; Geo-referenciação; Formulários
# Table of Contents

1 Introduction ................................................................................................................................. 1  
1.1 Motivation ................................................................................................................................. 1  
1.2 Objectives ................................................................................................................................ 2  
1.3 Results ....................................................................................................................................... 2  
1.4 Outline ....................................................................................................................................... 4  

2 Surveying and Crowdsourcing .................................................................................................... 5  
2.1 Surveying with a geographical component ............................................................................. 5  
2.2 An Ideal Tool - the Smartphone ............................................................................................... 6  
2.3 Internet Crowdsourcing Systems .............................................................................................. 7  
2.4 Other Smartphone surveying Applications and Platforms ...................................................... 11  

3 System Goals, Functionalities and Roles .................................................................................... 15  
3.1 User roles .................................................................................................................................. 16  
3.2 Solution Overview ..................................................................................................................... 17  
3.3 Main Implemented Features ..................................................................................................... 17  

4 System Architecture/Implementation .......................................................................................... 27  
4.1 Overall Internet System Architecture ....................................................................................... 27  
4.2 Django Server ............................................................................................................................ 28  
4.2.1 Django Project/Applications, Modules .................................................................................. 30  
4.2.2 Running Django on Apache .................................................................................................... 31  
4.2.3 GeoDjango Model API and View API usage ............................................................................ 33  
4.2.4 Database Architecture ........................................................................................................... 34  
4.2.5 Integrating third party administrative tools .......................................................................... 35  
4.2.6 Google and Facebook OAuth2 over Django’s auth ................................................................. 38  
4.2.7 Execution Flows ..................................................................................................................... 41  
4.2.8 Data Export ........................................................................................................................... 46  
4.3 Android App ............................................................................................................................... 49  
4.3.1 Android Operating System .................................................................................................... 49  
4.3.2 Application Development ...................................................................................................... 49  
4.3.3 Application Architecture/Main Components ......................................................................... 51
4.3.4 Home Activity and the Menu.................................................................52
4.3.5 Background Service and Location......................................................53
4.3.6 Background Service and the remote Server........................................54
4.3.7 Manual Check for Location/surveys.....................................................55
4.3.8 Interacting with Web content..............................................................55

5 System Evaluation..................................................................................59
  5.1 Functionality validation.......................................................................59
  5.2 Comparison to other similar systems...................................................61
  5.3 Demonstration/APA Feedback.............................................................65
  5.4 Mobile Data/Energy consumption/Application Installation................66

6 Conclusions and Future Work.................................................................69
  6.1 Conclusions.......................................................................................69
  6.2 Future Work......................................................................................70
List of Figures

2.1 Smartphone usage in some advanced economies...............................................................8
2.2 Portuguese population’s device usage, by type........................................................................8
2.3 Typical crowdsourced task assignment and validation............................................................10

3.1 Key concept- surveys and users over the map........................................................................15
3.2 Basic User Roles..................................................................................................................16
3.3 Admin reserved area: Example with full privileges.................................................................18
3.4 Attributing Administrator (superuser) status, and group memberships, to a user..............19
3.5 Surveyor Role: Form building concept..................................................................................20
3.6 Surveyor Role: Area creation..................................................................................................20
3.7 Surveyor role: User restriction creation..................................................................................21
3.8 Surveyor role: User restriction example..................................................................................21
3.9 Survey Activation/Survey Submissions...................................................................................22
3.10 Example of an Active survey, without user restrictions, and restricted to two Areas............23
3.11 Home page menu and the five (hide/show scheme) sections...............................................23
3.12 Login page........................................................................................................................25

4.1 Internet System Architecture: Client-Server model...............................................................27
4.2 Django’s MVT Pattern.........................................................................................................29
4.3 Usage of Django’s ORM......................................................................................................29
4.4 Apache Server and Django...................................................................................................32
4.5 Main Django models and their Foreign Relationships..........................................................34
4.6 Leaflet usage- Polygon building/Map querying.................................................................36
4.7 Area selection example: Two different beach Areas...........................................................36
4.8 Form building, with a title, introduction, and three fields (questions/requests)....................37
4.9 Oauth2 intervenients............................................................................................................38
4.10 OAuth2- basic procedure for obtaining a token.................................................................39
4.11 Django Middleware and HTTP requests.............................................................................40
4.12 Facebook prompting the user to authorize authentication................................................40
4.13 Authorization prompt, for a Google (Gmail address-based) account................................41
4.14 Administrative view of users’ Social authentication info....................................................41
4.15 Main User Menu.................................................................................................................42
4.16 Survey Listing; Button leading to the Map........................................................................42
4.17 Choosing a position on the map, using a browser............................................................43
4.18 Server logic for responding with available surveys..........................................................43
4.19 User restriction template including individual users and groups......................................43
4.20 Geo Querying for Area-restricted surveys........................................................................43
4.21 Survey list, presenting three omnipresent surveys and one Area restricted survey
4.22 Filling a survey, using a web browser
4.23 Survey presentation generation
4.24 Server’s reaction to an answer submission
4.25 Interface for answer export, on the website’s main page
4.26 Preparing for offline application evolution
4.27 Google’s estimation of user share per system version, as of August 2016
4.28 Google’s estimation of user share per system version, as of September 2017
4.29 Application Architecture
4.30 Home Activity
4.31 Service switch ON
4.32 Top-right menu
4.33 User movement, and background survey checks
4.34 Notification while on the Settings App
4.35 Notification over the Home screen
4.36 Play Store permission acceptance prompt
4.37 Manual Location Update
4.38 Survey List
4.39 Survey Form
4.40 Filling a text field
4.41 Tapping a file upload button (on Android 7.0)
4.42 Confirming or not (virtual device mock camera)
4.43 Filled Form, ready to submit

5.1 Form Building Interface
5.2 Map/Area Creation Interface
5.3 User Restriction Interface
5.4 Survey Deployment Interface
5.5 Android Pop-up Notification
5.6 Survey List
5.7 Form Answering, Extra File Button
5.8 CSV file containing answers, opened in a spreadsheet program
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMT</td>
<td>Amazon Mechanical Turk</td>
</tr>
<tr>
<td>APA</td>
<td>Portuguese Environment Agency</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>AVD</td>
<td>Android Virtual Device</td>
</tr>
<tr>
<td>CSRF</td>
<td>Cross-Site Request Forgery</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial of Service</td>
</tr>
<tr>
<td>GCS</td>
<td>Geo Crowd Surveys</td>
</tr>
<tr>
<td>GDAL</td>
<td>Geospatial Data Abstraction Library</td>
</tr>
<tr>
<td>GIS</td>
<td>Geographic Information System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HTML</td>
<td>HyperText Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>HiperText Transfer Protocol</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Message Access Protocol</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MVT</td>
<td>Model-View-Template</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>ORM</td>
<td>Object-Relational Mapping</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PDF</td>
<td>Portable Document Form</td>
</tr>
<tr>
<td>RegEx</td>
<td>Regular Expression</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SRID</td>
<td>Spatial Reference System Identifier</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WGS</td>
<td>World Geodetic System</td>
</tr>
<tr>
<td>WSGI</td>
<td>Web Server Gateway Interface</td>
</tr>
</tbody>
</table>
1 Introduction

Sometimes, there is a need to collect information from other people. Whether a single individual, a small business, a big institution, or even a government, they might all need information to perform some function, or at least to make what they do easier or more efficient.

We live in the Information age. Information can be, and mostly is, the key to success. If institutions can find better, more viable ways to obtain information, they can have a better performance. If information comes directly from a key target population, it becomes even more valuable. If you could get everyone you want to provide answers to your questions, and for them to do it of their own will, at little cost, that would obviously be a perfect situation. That is not really possible in a lot of situations, but the closer you can get to that, the easier it is to survey, the easier it also becomes to get people involved, and (hopefully) the better everyone's life becomes.

1.1 Motivation

Surveys have been, and still are, important means for institutions to gather information from people. They might differ greatly, whether in terms of how they are performed, what their purpose is, and what their target population is (who will produce the survey's answers). It all comes down to the type of information that the surveyor wants, who should answer the survey, how the surveying is performed, how the audience is reached, what kind of effort will it take from participants, and what is the motivation for someone to participate in the survey.

One survey might have a simple goal, be simple to perform, and be targeted at a limited population the surveyor has direct access to. An example would be determining the level of satisfaction a certain school classroom has with their curriculum, via filling a simple paper survey. That information won't be so hard to obtain, given that the surveying teacher has direct access to the students (the whole target population), and all it takes is a simple page with some questions for them to fill. It may take little effort, to just fill some answers in. Getting the students to fill a paper survey has inherent motivational factors. If they don't want to disappoint their teacher, they'll do it for that. If they don't like the class (and there's anonymity) they'll do it to vent out their complaints and opinions. This is probably as easy as surveying gets.

The surveyor may not have personal access to the target population. In that scenario, reaching the audience, giving them access to the surveys and receiving submissions become issues that must be resolved. Other issues may arise from other possible requirements, like geographical components in a survey, the need for media like photographs, or a need to authenticate the users and identify their answers. These can become tough questions, and it is possible for the surveyor to get to a situation where getting the desired information becomes a seemingly difficult task to accomplish, that might need significant investment.
1.2 Objectives

As an example, a situation where local government wants to get information/feedback from the citizens in a region. In the old days, this would have several inherent limitations. A typical example is a local government institution needing a Photographic survey, with a geographical component (precise locations). That would mean that it could either count on citizens who have a camera, and mapping/navigational skills to put in the effort and deliver the information, including those photographs, or there'd be a need to assign some skilled workers to do that job. The second option sounds like the most likely one. It would be costly, not many answers would be provided, and the scope of reachable people would be very limited.

Nowadays, devices like Smartphones and Tablet computers have become widespread and popular, not only gradually replacing the old mobile phones, but also fulfilling people’s desire for information, entertainment, and a new way to make use of the Internet. Everything was made a lot easier. Whether it’s a weather forecast, news, social networking, videos, all that sort of information gets fed to people effortlessly through their pop-ups, notifications.

This can be applied to surveying. Survey information can also be presented to people through simple pop-up menus. Instead of tedious, and sometimes very costly, surveying can be made easier, more tolerable, and more affordable for everybody. Recent technology opens up an array of possibilities that are explored.

Smartphones have several interesting functionalities that can be explored. They usually have means of connecting to the Internet (Wi-Fi and mobile data network usage). Most also have integrated digital cameras, that keep improving in average quality over the years. Also, GPS technology has been integrated, allowing the users to obtain somewhat precise locations using their devices. The two major mobile operating systems (Android, IOS) simplify the publishing of new applications, in the sense that they allow developers to easily make their software reachable and available to other people. These are factors that can be taken advantage of.

The Geo Crowd Surveys project is aimed at demonstrating the possibility of conducting surveys that include a geographical component, and media files using Smartphone technology. Specifically, developing a demonstrative system that provides users with a mobile application using their Smartphones’ location capabilities, cameras, and Internet access, to acquire, fill, and submit survey answers. This is meant to show that it is possible to simplify the surveyor’s tasks making them available on a Website, and also make it simple for users to answer and submit surveys, wherever they may be, by getting survey information (and the ability to answer) from a smartphone application.

1.3 Results

The concept that was implemented is for there to be a server side application that interacts with a user side application, over the Internet. A server should have some available surveys at a given moment. These surveys can be different from each other, ask for different types of information, and might have location restrictions (only applying to certain areas of the map). A user application fetches that information from the Server, and presents a list of available surveys to the user. The user may
choose to answer a survey. When that happens, the survey is presented to the user, who can fill in the answers. The application should be able to handle submissions using the Internet. This is the all that composes the participant's task.

A very important requirement for the user is simplicity. The user should have to put in the least amount of effort possible, and only have to fill the survey answers. The user only needs to open an application on their smartphone or tablet, and check what surveys are available for the current location. When one of them is chosen, it is presented. The user fills the survey. Even in cases where photograph answers are required, the Camera application is automatically used to fill those answers, and the new photo is integrated in the survey answer.

The user side of the system (application or Website) is responsible for some simplifications to the user's task, including automatically geotagging the answers, and using the device's camera to get photographs. This means the application makes use of the user's current geo-location and automatically includes it as an answer field. It also uses the device's camera if necessary. There is a need for all this to be integrated seamlessly, with the ultimate goal of making the user's experience as easy as possible.

Regarding the surveyor, a very important requirement is flexibility, and being able to adapt according to each different situation. The surveyors can build their own custom surveys, for different situations. They might also want the deployed surveys to be restricted to certain areas. Also, there might be the need for certain surveys to be restricted to some users, or groups of users.

The surveyors might not be skilled in Computer Science, so their interface is user friendly and doesn't require any special skills to operate. Building a simple end-user interface means that the scope of people who can actually participate in surveying is not limited by the technology. The same was applied to the surveyors' interface. If it became too complicated or required some special skill, the amount of people who can use a system like this can be greatly reduced. Making the system as friendly as possible was a concern.

The implemented system should be perceived as a useful tool by the surveyors that solves their whole surveying process, from survey creation and deployment using a web page, to the exporting of answers. They have an intuitive interface that allows them to create and manage their own surveys, and apply restrictions to them. They input some work creating and managing some surveys. The outputs they get are the answers, which are easy to extract from the Server.

Some Internet System Architecture concepts are important for this project. Highlight concepts are RESTful Web services, Mobile Code, Web Frameworks, Database Management, Mobile Computing, and also some Security Concerns. The chosen architecture to fulfill the requirements is a Client-Server architecture, in which there is a single server, and multiple clients. Several web development tools were used building the server side application.

A Server application was developed and tested on a single server. The server application provides an authentication system. Users can be authenticated using dedicated accounts or using Google and Facebook. A role system is enforced. There are multiple available roles, and specific individual permissions can be assigned to users. There are geographic functionalities. Surveyors have a graphical interface that allows them to delimitate areas on the map, and the server can determine
survey availability based on location using geo querying capabilities. Surveyors can also restrict their surveys to certain users or groups of users. Form building is made available, and surveyors can create custom forms whenever they are needed, with multiple types of available fields. A mobile application was also produced and deployed for testing purposes. Submissions are handled and stored, and both the surveyors and the participants have the option of exporting the answers.

The users can participate using a web browser or using a mobile application. Both give the user information on the server’s available surveys, and the ability to answer. When using a browser, the user has the ability to indicate the position they are submitting an answer for, and use their files in answers. The mobile application makes use of the user’s current location, and presents the option to take photographs on the field, when needed. The application demonstrates automatic survey searching using location capabilities, providing users with the option of being notified when they enter an area that has them.

GCS was presented to the Portuguese Environment Agency’s (APA) Coast and Coastal Protection Department (APA is a public institute within the scope of the Portuguese Ministry of the Environment, Territory Management and Energy) as a solution for their coastal monitoring process. They endeavor to monitor the coastline and track natural phenomena [1] like beach and dune cord erosion, oceanic gulf and flooding, coastal protection infrastructure damage, and damage to public infrastructure and beach equipment. Although not a perfect solution, with possible improvements, GCS it is able to solve their surveying problem, and found positive feedback, being appreciated especially for it’s geographic capabilities and user interface clarity.

It can also solve a similar surveying problem for a project that was a key influence for this project [2], (Cascais Beach Photo Monitoring) that was developed upon agreement by the Faculty of Sciences of the University of Lisbon and the Cascais Municipal Council. It is a quest to photographically monitor water line displacement and sand gains and losses over time in beach areas, with the purpose of establishing guidelines for crowdsourcing tools to support beach monitoring programs and ways to engage the public.

1.4 Outline

The rest of this document is organized as: An analysis of crowdsourcing and how it has been used, along with the current state of the art surveying are presented in Chapter 2. Chapter 3 describes the resulting surveying system’s functionalities and how it can be used. Chapter 4 presents the resulting system’s implementation, from it’s overall architecture to some key individual components. Chapter 5 provides a thorough evaluation of the resulting system, compared to various alternatives, and feedback obtained for the demonstration version, namely from APA. Chapter 6 presents conclusions and a perspective on possible future evolutions.
2 Surveying and Crowdsourcing

This section presents an analysis of current and past surveying systems, and how crowdsourcing is a very important concept in relation to surveying. Despite recent advances, surveying a crowd with integrated geographical components, as presented in this project, is a relatively unexplored area, even in the current smartphone era.

2.1 Surveying with a geographical component

In the old days, before personal computing, a survey taking location into account would be expensive, typically requiring a dedicated and knowledgeable staff. With the digital age, things became easier, especially given the ability to get the current location using GPS receivers, and the possibility of using of text messages or email to submit answers. Nowadays, in the smartphone era, things are much simpler, but not yet easy enough. There's an array of available mobile applications for Crowd surveying, but they implement very little in terms of geographical functionalities. There's also a significant amount of geographical applications that pack diverse functionalities but that are not well suited for surveying.

The point of this project arises from this. It is developing and demonstrating a surveying system that resides in the gap between location-unaware surveying applications, and the geographical/geological applications. It should include the world map as an important part of the surveying process, both creation and participation.

In some cases, location capabilities make a big difference. This is true because some surveyors have the need to know exactly where the answers they are getting come from. Surveys that can pinpoint occurrences, something happening in real world locations. Also, they may need to target specific surveys at specific geographic areas in the real world.

This need for location information implies a burden that would typically have to be carried by the people answering the survey. They would need to know what the area of interest for some survey is. Besides that, when filing an answer, they would have to have a way to obtain their current position, and include it in their answer somehow. This burden is actually a great limitation inherent to the Geo Crowd Surveying systems and solutions of the past.

A simple example survey is asking people to give their opinion on how crowded a beach is, during summertime. All the participant really has to provide is be something like a 0 to 10 number to describe how crowded the area was at some time. If a surveyor needs to get further detail and map which areas get the biggest crowds, location information regarding the answers would be needed (geotagging, i.e. adding geographical metadata to the answers). This is not as straightforward as just delivering some number and a time. The point would be gathering data to evaluate which parts get the most crowded, and when. Naturally, getting the opinion of several people, a broad audience, a big crowd, is desirable for that.

Apparently, this beach question is a very simple survey with just a couple of fields to fill, but things really are not so simple. Especially before GPS signal receivers were available (and before the so
called Digital Age), this would mean relying on people to have good enough mapping skills to determine their position, or giving them some tools, for them to be able to assert or mark their position, etc. Not only would that task be assigned to whoever answered, but it also incurred some extra costs for the surveyor.

There was usually a big trade-off. Asking for location means asking for more effort. Giving the participants more effort also means they need more incentives, or less people will be willing to participate. The more that is asked, the more expensive it becomes, the harder it is to get people to participate.

Surveys that asked those types of efforts typically needed a dedicated staff to produce answers. The more that’s asked of people, the harder the crowds become to reach, and this type of surveying gets limited to restricted audiences. There would be an inherent cost of providing people some way of deducing their location, by providing mapping materials or tutoring. Also, a very likely cost would be that the results are dirty/inaccurate. Not only would the surveyor have to go through some trouble to get results, but also they might not be satisfactory enough.

Even after GPS receivers became available, they were never as widespread and commonly available to the average person as smartphones have been in the last few years. Despite making precise location simple to obtain, not everybody had/has dedicated GPS receivers. So, even with GPS, asking for geotagged answers still was not easy and imposed a limit on the amount of people able to participate, which was still considerably limited, despite being in the digital age, with personal computers, the Internet, etc.

An important factor, besides the surveying requiring specific skills from participants, is their motivation to do it, and the required effort. The worse these factors get, the less likely it will be for people to be willing to contribute to a surveying project. In order to potentially reach bigger participating crowds of contributors, making it as easy and simple as possible is crucial.

It is worthy to go through the status of Geo surveying in the most recent years. It is relevant to go through some examples, in order to learn: what to do and what not to do; what are the failures and disadvantages other systems have had/still have; their recent evolutions; what can be expected from modern a location-aware crowd surveying system that relies on smartphone technology; why it has potential use.

2.2 An Ideal Tool- the Smartphone

If we went back to before the digital age, and needed surveying with any sort of geographical component, it was almost certain some staff would be needed not only to participate, but also to curate the results, and archive them properly. It was a different reality from the one we live in currently.

Besides dedicated personnel to produce answers, products, there would be a need for someone to organize/curate those products. Archiving a substantial amount of photographic survey inputs would take physical room, and browsing the results would not be an easy task. If researchers wanted to get pictures in a certain area, in a certain specific date range, that could mean an exhaustive query through all the paperwork and files.
The impracticality of these efforts lead to a poor efficiency when compared to the modern methods at our disposal. The investment it took meant that not many institutions had the financial capability to order these types of surveys. Also the big crowds were not accessible and unable to participate in efforts like that. Any idea of crowdsourcing these tasks would be very complicated.

Digital cameras allow for taking a great number of photographs without having to repetitively buy photographic film, or spending money developing them. Besides the initial investment, people can take thousands of pictures at a very low cost. Digital pictures can be easily presented on a screen, and the Internet allows for sending them to other people. This can eliminate the non-digital photograph handling and storing problem, and achieve a lower cost per picture (in the long run).

The GPS (Global Positioning System) was made available to the public in the 1990’s. Back then, it had “selective availability of public signals” [3], that intentionally added pseudo random errors to the signals. This meant that the civilian GPS receivers allowed people to obtain their approximate position, with some intentional imprecision added to it. That was removed in 2000 and since then GPS has become a staple to our daily lives. This is important for geographical surveying, because it allows people to get their current coordinates with high precision, only requiring them to know how to operate their receiving device.

Still, although it has been possible to manually get results from each device, integrating them all in a single surveying system needed considerable effort from the users. This lack of integration among the different devices leads to clumsy solutions carrying multiple devices (compared to what’s possible now) and required people to have access to all of the different types of devices.

Earlier mobile phones (plain “cell phones”) allow people to exchange information, namely phone calls and text messaging, from anywhere that has a reception. They have been progressively replaced by the smartphone devices, that has several extra functionalities, integrating personal computing concepts, with mobile phone capabilities, location capabilities, commonly (almost predominantly nowadays) possessing digital cameras, and the ability to connect to the Internet, and possibly gyroscopes and accelerometers.

2.3 Internet Crowdsourcing Systems

With personal computing and the rise of Smartphones, the general public has become more tech-savvy and able to participate in other people’s projects using the Internet. In developed countries, smartphones are especially predominant among the younger generations.
According to the Pew Research Center's spring 2016 Global Attitudes survey [4] (results shown in Figure 2.1) (applied in 14 “advanced economies”), “one thing was clear: In each of the countries surveyed, nearly all people reported owning a mobile phone. But the shares who own a smartphone vary considerably”. The growth of the smartphone market is visible through “(...) Americans (77%) said they owned a smartphone, a number that more than doubled since 2011”. A gap between younger and older people, and more educated and less educated people is also very significant: “For example, 84% of Greeks ages 18 to 34 reported owning a smartphone. Among Greeks ages 50 and older, only 13% owned a smartphone. Among more educated Greeks (those with more than a secondary education), 79% owned a smartphone vs. only 35% of Greeks with a secondary education or less”.

**Figure 2.1 Smartphone usage in some advanced economies [74]**

**Figure 2.2 Portuguese population’s device usage, by type [4]**
The 2016 Connected Consumer survey [4], on behalf of Google, and searchable via the consumer barometer tool got results percentage of the Portuguese population that uses smartphones at 59%, personal computers at 65% and tablet devices at 31%, as shown in Figure 2.2.

It is important to introduce the concept of Crowdsourcing. A simple dictionary search for crowdsourcing produces “to utilize (labor, information, etc.) contributed by the general public to (a project), often via the Internet and without compensation” [5]. It differs from outsourcing, as outsourcing gets contributions from a well specified group of participants, but crowdsourcing does not specify exactly who will participate, at the start of the project.

The recent mainstream introduction of globally-reaching Internet technologies such as crowdsourcing may be a solution to the limited participant pool with which researchers must sometimes work [6]. Not only a solution for companies to do market research and interact with the crowd, but also for scientific purposes. Scientists can view these devices as embedded sensors with the potential to take measurements of the Earth’s surface and processes [7]. A crowdsourced worker pool might be more diverse than a typical university survey group, for example. S. Whitmeyer and D. Paor [8] affirm that if the participants don’t need in depth knowledge of the topic, then Crowdsourcing is practical for rapid data collection, and that emerging digital technologies are creating new opportunities for crowdsourced geological mapping.

Crowdsourcing has evolved and taken new forms, especially in the past 5 years. From a tool where paying a crowd was a necessity [9], into new possibilities where the crowd works for free, or even pays someone for them to create products (i.e. Crowd funding), as is the case for current platforms like Patreon [10], Kickstarter [11] and Indiegogo [12].

M.Yuen et al [13] refer to Jeff Howe, who in 2006 defined crowdsourcing as “an idea of outsourcing a task that is traditionally performed by an employee to a large group of people in the form of an open call”. The existence of 2 roles is mentioned: requester and worker. The workers may be rewarded by the requesters. In some cases, workers are not motivated by rewards, but they work for fun or altruism. Crowdsourcing applications are grouped into categories, and they are voting system, information sharing system, game and creative system. Geo Crowd Surveys should, in theory, be able to fit in as a crowdsourced Information System, as the crowd can contribute with survey answers, which contain information. It is stated that Crowdsourcing (in 2011) was still a relatively unexplored new subject.

This type of Crowdsourcing might need some quality control mechanisms to assure the quality of the crowd's product [14]. For G. Li et al this is due to the results collected from the crowd being inherently dirty and ambiguous, because workers may return noisy or incorrect results so effective techniques are required to achieve high quality; A typical example, using Amazon’s Mechanical Turk [15] (AMT) crowdsourcing platform where the requester publishes their tasks on a crowdsourcing platform, would have crowd workers who are willing to perform such tasks (typically for pay or some other reward) accept the tasks, answer them and submit the answers back to the platform. The platform collects the answers and reports them to the requester. It’s also relevant to note that tasks can be classified into macro-tasks (e.g., translating a paper) and micro-tasks (e.g., labeling an image).
Individual users answering surveys constitute micro-tasks. Therefore, the Geo Crowd Surveys project does not suffer from the coordination problems that are involved when facing macro-tasks that require coordination among multiple users, as their efforts are independent from each other and produce independent inputs.

![Diagram of typical crowdsourced task assignment and validation](figure)

Figure 2.3 Typical crowdsourced task assignment and validation [16]

The quality control mechanisms and decisions depend on each surveying entity. It must decide what controls to apply according to the situation [16], as shown in Figure 2.3. Controlling user inputs requires their authentication, and that was taken into account in this Geo Crowd Surveys project. Knowing who is responsible for each submission is an important requirement, in order to have the possibility of either rewarding good contributors, or cutting off ill-intentioned users’ access privileges.

The above mentioned AMT platform [9], [13], [17] (created in 2005 and still running) is an example of Crowdsourcing where users get paid a specified rate for each contribution. The survey requester has to pay for the ability to publish a survey on this platform, and chooses a rate for the participating users’ payment, per input.

A very different example has a governmental agency asking for people to collaborate in obtaining geographical information, voluntarily [18]. This is related to multiple (29) simultaneous World Bank Global Facility for Disaster Reduction and Recovery projects, related to, for example disaster prevention and relief, adapting to climate change, or wildlife preservation efforts, taking place all around the world. The authors defend that right after defining the problem, the most important factors are communication with the audience and to “know the online community and their motivations. It is important to know whether a given crowdsourcing application will appeal to participants”. In this case, there is a generally a motivation of helping out and lending a hand to communities in need, and some competitions to reward participants.

This shows potential for crowdsourcing projects to be applied in similar, not-for-profit situations, where participants have the desire to help some institution, and donate an effort to their cause. This opens up the possibility of some institutions being able to get free collaboration from people belonging
to the general public, that are interested in helping their projects. Those institutions need to find the right way to present the situation, and the right means for people to be able to help.

The surveyor needs to be able to ask different types of questions. A recent online survey by the New York Times asked readers for their opinion on the current state of their neighbourhood [19]. They were asked to state their views via submitting a text on a Web page. This generated a (relatively) low number of inputs (around 1200). The author, Sam Hodgson, stated that the presented findings were hardly scientific, and that there is a human inclination to use surveys as an opportunity to complain and so negative opinions might be accentuated, going as far as saying “complaining about the city is a recognized art form”.

Surveyors might need to ask for text responses in some situations, but also for other types of answers, that allow them to quantify results. A good surveying system needs flexibility so that surveyors can build different types of surveys and get the best input possible.

2.4 Other Smartphone surveying Applications and Platforms

Apple’s App Store and Google’s Play Store make it very easy to find and download applications. Developers have the job of designing the software, making use of each of the necessary typical smartphone or tablet device’s capabilities. They pack some functionalities in an App, give it a graphical interface, determine the exact way the application is presented to the user, and should make the application friendly to use (if possible). Then, they publish the App. All the user has to do is download it, tap the App’s icon, and their device will provide them with functionalities that the developer built.

There has been focus from the scientific community to take advantage of this, and use smartphones as tools for scientific purposes. Of interest in the scope of this project are Geological/Geographical applications, especially those having crowdsourcing functionalities.

There is a plethora of applications like this that don’t have a crowdsourcing component, for use by individuals. Typically these applications make use of smartphones’ ability to use GPS coordinates for geotagging purposes, and use smartphones’ gyroscopes to determine orientation and tilt of surfaces [20], [21].

There’s always the possibility of just using mobile phone’s basic capabilities, namely short text messaging. The Social.Water project [22] did that with the idea of having some gauges that measure the water level at certain locations. People can read them and send a text message to a number informing about location and current level. Google Voice is used to create emails from those messages, and forward the information to an IMAP email account. A server checks the inbox of the email account every 5 minutes. In case there is new information, it is parsed and stored with a timestamp on the file corresponding to the gauge.

Projects like this don’t need a user-side application to be developed, which is possible due to a single value (and identifier) being necessary for an answer, but have some weaknesses if no App/Website is used, the most notable being a lack of authentication. The user needs to know what the message must look like (the message has to have lowercase letters and the right format). Users can also insert bad info into the system. This really depends on users’ good will.
Some geo crowdsourcing/surveying applications are developed without having any sort of Server or other devices to establish direct contact with [8], [20]. In these cases, the results have to be exported by the user from their App, using some file format. This implies that a surveyor that is being supported by several users, using the App, has the job of collecting output files from each user, and spends time and effort to merge results from several different people, which is not desirable as a surveyor-friendly solution. The users have to explicitly send files to the surveyor, and the surveyor has the task of handling them one by one. Albeit having powerful geographical capabilities, recent applications like Mappt [23] have an absence of surveying functionalities.

There’s also the possibility of only implementing online geo-surveying functionalities that don’t make use of smartphones at all, solely developing web-based solutions using publicly available tools [24]. Another possible alternative is to use pure online form building solutions.

One possibility is building PDF forms that are submitted over the Internet. This allows for plenty of form building functionalities, but has no automatic way of authenticating the users and no geographical functionalities.

Arguably one of the easiest to use, most intuitive form building systems available (and free) is Google Forms [25]. It allows for quick and easy building of surveys, easy sharing with participants, and also an easy way to either visualize results or export them. It has the somewhat rare functionality of enabling conditional forms, where some questions might differ based on the answers. It does not, however, make use of mobile devices’ location capabilities, or have any type of location/coordinates related functionalities, for that matter. Users are not authenticated. Also, the answers are stored in the cloud, which might arise privacy concerns from users. Google has not (yet) made smartphone applications for Google Forms.

To fulfill the requirement for ease of use and simplicity for both the participants and the Geo-surveyors, besides a Website, a smartphone application, and a Server entity to connect with over the Internet, are desirable. This means that most available smartphone applications (which are autonomous tools and don’t establish connections with some coordinating entity on the Internet) are sometimes usable but inadequate for crowd surveying because these requirements cannot be satisfied. There are no available free solutions that accomplish this.

There are available smartphone Crowd surveying applications, that have associated platforms on the Internet, that manage and distribute the surveys, and allow surveyors to visualize results and analyze data using their back-end software. All have form building capabilities. Some have offline capabilities, which means having an application that works even with no Internet connection, and synchronizes data with a remote entity, when a connection becomes available. These solutions are paid by the surveyor, typically via subscription plans. Commonly there are free or very cheap starter plans that impose very short limits on the number of answers, and paid plans that increase those limits. Some platforms gather and pay participants (paid crowdsourcing) to answer the available surveys, like Google Opinion Rewards [26], Panel App [27], or Surveys on the Go [28]. Others have the possibility of “manual” distribution of the surveys to select participants (via Social Networks, Email, or simply URLs) by surveyors who have a well-defined target audience, and know exactly who they want to answer their surveys, like SurveyMonkey [29], QuestionPro [30], QuickTapSurvey [31], or
LimeSurvey [32]. These applications and platforms are generally directed at market research and opinion polling purposes, and have limited geographical capabilities or none at all, with just a few like ESRI’s Survey123 [33] and Open Data Kit collect [34] offering the possibility to register the device’s GPS location, and no other geographical options like Areas or user/group restrictions for the surveyor to apply. Most of these platforms require data to be stored in their cloud platforms, which might be undesirable as in the case of Google Forms, because of privacy concerns.

Summing up, there is a wide array of geographical/geological smartphone applications packing a lot of functionality, but somewhat hopeless for crowd surveying. Some of these are/were used for surveying but are ultimately not practical enough due to each participant having to manually submit their results to the surveyor via email, cloud storage, etc. and the surveyor having to handle all the different submissions manually.

There are also integrated web platform plus smartphone application sets, with considerable form building functionalities (albeit none reaching Google Forms’ level), but these are typically geared for market research, assessment/evaluation, opinion polls etc. and have limited to no geographical capabilities and most are obligatorily on their cloud.

The idea for Geo Crowd Surveys is to achieve the potential in the gaps between the existing solutions. A Server-side project that can be run on a single machine (which doesn't have the privacy concerns that the cloud has), (or potentially also run on a cloud platform, but this was not done due to the privacy issues and the costs of cloud deployment) that presents a Website as an interface, plus a smartphone application that can be distributed to participants. The Server project should have Geo capabilities, and provide surveyors flexibility, namely through form building, user restrictions/authentication, and ease of exporting the answers. The application should allow users to answer surveys on the terrain, including media files, and automatically geotag them.
3 System Goals, Functionalities and Roles

The key underlying problem/opportunity/need is that typically, institutions need surveying to gather geo-referenced data [1], [2], [18]. That implies outsourcing the survey, or having dedicated staff to perform it. The classic solutions are inadequate. Crowdsourcing these tasks using the Internet is possible, but the current approaches to do so are somewhat unsatisfactory, mainly because they lack certain specific capabilities, or demand considerable effort from either the participants or from the surveyors, to compensate for those inadequacies. Geo Crowd Surveys is a system that uses the Internet to allow one or multiple surveyors to crowdsource Geo-surveys to the general population, or to select groups of users.

Also, a surveyor might only be interested in applying a survey to some map areas. For example, a survey might be meant for some beaches, but not for other areas around them. Surveys can be limited to one area, multiple areas, or none at all. Therefore, another goal is to enable surveyors to place surveys on specific areas of the map. Additionally, a surveyor might want to limit some of their surveys to a restricted group of participants, and have other surveys be available to the entire participant population. Participants should be authenticated, and possibly distinguished from each, in terms of permissions settings and group assignments.

The Geo Crowd Surveys project has the goal of making the participants’ task as easy as possible, using not only a Website, but also focusing on the smartphone as the key tool to provide survey retrieval and presentation, answer creation and geotagging, and answer submission over the Internet.
In principle, this system can be used in a variety of situations. In the simplest situation, the system could be used by a single administrator and a close group of friends or colleagues. The actual envisioned (and possible) use is for a situation where an institution needs help from the general population, and there is an inherent motivation (e.g. altruistic motives) for people to help (like helping an environmental organization, or helping their local community, etc.) without the need for explicit financial rewards people to participate.

### 3.1 User roles

The system’s different user roles and consequent hierarchy are shown in Figure 3.2.

Regular users (or participants) can answer surveys, given that they fit the survey’s user restrictions, and are inside the survey’s geographical area. Also, a regular user has the option of downloading their own previous answers and files they have submitted. It is up to the Administrators to decide who these people will be.

The people who need data collection are called surveyors. They can build and deploy their own custom surveys (name, description, and adding form fields that can be one of several types). They can also apply restrictions in terms of who is allowed to answer their surveys, and define areas on the map for the surveys to be deployed in. The surveyors can export the data their surveys have received. This is available as a Web Service. They also have the abilities regular users (participants) have.

Administrators have the power to create other user accounts, appoint surveyor status to other users, create user groups, assign other users to groups, and attribute specific individual permissions to each user or to a whole group. They can deactivate any survey, even if not theirs. It is possible for them to verify users’ identities, by asking them to log in with Google or Facebook. They can also deactivate accounts, which can be useful to disable users with bad behavior. They also have the abilities surveyors and regular users have.
The Webmaster is the person responsible for system maintenance. The Webmaster should monitor traffic/database storage capacity to make sure that the Server runs correctly. This person has the ultimate power to manage all users, but shouldn't be involved in that aspect unless necessary (that is the Administrators' role). The Webmaster is not necessarily involved in creating content/surveying activity like the other users are.

### 3.2 Solution Overview

The system makes use of web-based Internet technologies. The surveys are created and managed using a Website, and are answered by either Website users or smartphone application users. The survey presentation to the smartphone (Android) users is performed by an application. The application is also responsible for automatically using location information and geotagging the answers, and submitting them to a remote server. On the Website, that's done by clicking on a map to indicate positions.

There’s an authentication system, with the option of using social authentication providers to identify the users. Participants don't necessarily have to create an account by filling a form or asking an administrator to do it. It is possible for a user to create an account simply by logging in with Facebook or Google. Participants’ answers are automatically identified. Surveys have an owner (a surveyor) who can manage the survey and its restrictions, or retrieve the obtained data.

Surveyors can build custom surveys. They can restrict surveys to specific Areas on the map. It is also possible to restrict surveys to specific participants or groups of participants.

There is a Server entity, enabling a website and some web services, where surveyors and Administrators can create and manage the surveys and participants. The website also provides a main page where all the users can get an overview of their involvement and access most of the available functionalities. The website also provides an alternative (to the mobile application) way of answering the surveys, using a web browser, although it requires pinpointing a location on the map, and manually uploading files from the user machine’s file system. The Server also provides Web services, namely survey answer export, downloading the submitted files, available survey information export and area description export.

### 3.3 Main Implemented Features

Part of the Website (partially displayed in Figure 3.3) is restricted, for Administrators and surveyors (the regular users/participants don't have access to it).
Management

Group management is possible, for Administrators. They can manage user groups and their respective permission settings. This means that an Administrator can assign very specific permissions to each specific user group, which makes it easier when compared to just assigning roles individually, to each user.

This flexibility means that the roles are not set in stone. It is possible to have surveyors with different sets of permissions, or participants that don’t quite have surveyor status but might be allowed to delimitate Areas on the map, etc. Administrators can create groups, add users to groups or remove them.

Administrators can also create new users and set their account information; it is possible to suspend a user, as there is an option for activating or deactivating an account. It is also possible to outright ban a user by deleting their account. Giving a user surveyor status (staff member) and giving another user Administrator (Superuser) status (full privileges), as displayed in Figure 3.4, are options.
Authentication

In terms of social authentication, it is possible for users to authenticate, on the website, using third party authentication providers. For the demonstration version of the system, Facebook or Google (OAuth2 protocol) can be used, and it should be simple to add many other provider options in the future [35].

The Server receives the user’s basic profile information from Facebook. This can be used by the Administrator to see who that person is on the Facebook site, by visiting their Facebook profile page. On the other hand, when a user logs in with Google, their Gmail address is provided to the Server. These functionalities can be used confirm a user's identity.

On first login, the Server will create a new account specifically for the user, and store the associated social information obtained from the authentication provider. The username is unique and chosen automatically by the Server, based on the information catered by the authentication providers. The Server will also suggest the user to choose a password for their account, so that they can also login using a username-password pair.

Form Building

Form Building is one of the most important features available. Surveyors can build custom forms. There aren't any pre-made forms, and it's up to surveyor to decide what to ask from users (Figure 3.5). This sort of functionality empowers surveyors and provides them some degree of independence from the system’s developer/programmer, as new surveys can be created at any time, without any developer/webmaster intervention.

There are several usable fields that surveyors can choose. The number of fields for each survey is not fixed, and is unlimited. The available field types are: single line text, multiline text, file upload, email, number, URL, check box, multiple check boxes, drop down, multi select, radio buttons, date/time, date of birth. A hidden field for a point on the map (2 coordinate values) is automatically suggested, as location plays a crucial part in the surveys (it is possible for the surveyor to remove it, though). The surveyor can make a field obligatory or not. Fields like dropdown or multiselect require
the creation of choices, which the surveyor must provide. The surveyor can also write a help text for each field, and choose the order in which the fields will appear on their survey. The Form has a title and description (or introduction) that the surveyor can write to give participants a description of the survey, and any extra information they might want to divulge.

Survey Setup

Surveyors can delimit areas on the map (as illustrated in Figure 3.6) and save them. Various types of map layer presentations are available to surveyors: regular map, satellite view, and alternative maps. The surveyor can demarcate the areas by clicking on the map and defining the area’s polygon vertices. There is no limit on the number of vertices for the area polygons. It’s all up to the surveyor to decide where to place their Areas. According to the level of detail, several vertices may be necessary. As an example, the following figure, in order to obtain as approximate of a model of IST’s Alameda campus, over 20 vertices were placed on the map. (It would have also been possible to enclose the campus by placing a rectangular shape around it, which would only have 4 points, but would include some undesired areas).
The areas might also be very small, corresponding to just a few square meters, or be very large, encompassing a whole country or continent. It is up to the surveyors to delimit the Areas they might be interested in.

Surveyors can also create and manage their own user restriction templates (Figure 3.7), to which they can add or remove users and/or user groups. These administrative templates (as in Figure 3.8) can be associated to surveys. The purpose for this is to deploy surveys that require some specific knowledge, or otherwise private surveys, to a limited set of users chosen by the surveyor. This means that none of the other users will ever see the survey, or be able to respond to it. In fact, non-authorized users will have no knowledge of a limited survey even existing.

![Figure 3.7 Surveyor role: User restriction creation](image)

![Figure 3.8 Surveyor role: User restriction example](image)

**Survey Deployment**

Surveyors can deploy their actual surveys and have them collect answers in the real world (as expected). The concept of Active survey arises from this. An Active survey is a survey with a well-
defined form, and possibly restrictions, that participants can provide answers to. Those answers are stored in the Server, and can be accessed by the surveyor and also the participants themselves. Surveyors can have as many Active surveys as they want to, as illustrated in Figure 3.9.

![Figure 3.9 Survey Activation/Survey Submissions](image)

An active survey is comprised of 3 types of elements: a Form built by the surveyor (exactly one), and may optionally have Area and/or User Restrictions, as shown in Figure 3.10. Several Areas might be assigned as restrictions, or the survey may be active everywhere on the map. If a surveyor does not assign any areas to an active survey, then it is assumed that the survey is omnipresent, available everywhere on the world map. If it assigns one or more (there’s no limit) then the survey automatically becomes Area restricted. In terms of user restrictions for a survey, only one user/group restriction template can be applied at once. If none is chosen, then it is automatically assumed that the survey is meant for everyone. Otherwise, it becomes user-limited according to the chosen restriction configuration.
Figure 3.10 Example of an Active survey, without user restrictions, and restricted to two Areas

The active surveys are flexible. For example, if the surveyor wants to add more Areas or remove some from the restrictions, that is always possible. As an example, if an active survey has Area restrictions and the surveyor unselects all of them, then it automatically becomes omnipresent. Also, it is possible for the surveyor to switch among different user restriction configurations, and modify them by adding or removing users and groups, or abandon them altogether. It is also possible for the surveyor to change the Form fields, albeit undesirable except for minor changes, as it may mix very different types of answers together (as in a surveyor that has a 5 field Form suddenly dropping all the fields and choosing new ones). In this case, with major changes, the surveyor should just create a different, new survey.

Web Portal

The demonstration Website has a home page (partially displayed in Figure 3.11). It can be visited by all users, and presents extra options to surveyors and administrators that regular participants won’t see.
One section shows the user their Account information: username, registered email address, account status, group status; In case the user is an Administrator, a link to the reserved section (the Admin interface) is presented.

Another section lists the surveys that the user has answered to, and provides the option to download those answers in a CSV format file, that can be opened by typical spreadsheet programs and even simple text processors, as it is a text file.

A third section lists all the surveys the user can answer and their corresponding descriptions (their introduction); this takes any existing restrictions into account. It contains a button that leads the user to a map, on which they can click to look for surveys for a specific position. Surveys can be answered from there.

The fourth and fifth sections only appear for administrators/surveyors. The fourth part lists the user’s own surveys and the option to export each survey’s answers, or modify the surveys and their restrictions. The export can include every answer, or only those after a specific date. This option was implemented to avoid surveyors having to download old answers they’ve already seen multiple times. There’s also a button that leads to creating a completely new survey.

The bottom section shows the history of the user’s administrative actions (such as a creating a restriction or activating a survey).

The Website’s Home Page front-end technology is easily extensible, being wide open to future additional options like service announcements, server news, user contribution acknowledgement, etc. It is possible to implement different looks and even completely different paradigms for presentation, for different institutions using the System.

A custom Login/Logout area (Figure 3.12) was implemented, with an additional area for Google/Facebook login. For the demonstration version, Google and Facebook login work for desktop web browsers. It is possible to implement similar functionality for mobile applications with additional configuration work.
Survey Submission

Survey submission functionalities were implemented. There is a Map interface to pinpoint location (this is needed on a web browser, but unnecessary on the Android App, as location is obtained via location services (GPS, Wi-Fi, cellular network). Surveys are listed and presented, and the user (participant) can click their titles to answer, for the given location. The user can choose among available surveys. The chosen survey is presented for the user to fill the fields. In situations where additional files are wanted, the user might want to include some more (for example if pictures of some terrain feature are needed, the more information the surveyor can get, the better the chances for good input appearing will be). Because of that, there is a button that allows the user to upload extra files (one or more). The submit button can be clicked/tapped on, to send an answer to the Server.

The Server makes some RESTful Web Services available to users. Exporting survey answers via a CSV file download is possible. Each survey’s answers can be exported using their unique identifier, and the Server creates different responses according to who is making a request. In case someone tries to get answers they have nothing to do with, they’ll get an empty response. The surveyor who owns a survey can get all its answers. The CSV format is highly desirable, as it is one of the current standard formats, compact, human readable and editable either “manually” or with a spreadsheet program. The URL can include a start date, if necessary.

Also, each file uploaded by participants is made available for download to the survey owner and the person who uploaded it, with a unique identifier. The download URLs for each file are generated by the Server and placed in the CSV files, in the corresponding answer lines, so that the person who

---

![Login Page](image-url)
downloaded it knows where they can find each answer’s files, like uploaded photographs. To download a file, all it takes is for the given URLs to be requested, with a typical web browser.

**Android Application**

An Android application was developed to demonstrate the smartphone survey filling and submission potential. The application allows the user to login or logout using their username and password, that they should obtain using social authentication on a desktop/laptop web browser, or directly from an administrator. The application uses current location, typically obtained via GPS. Alternatively, it makes use of the device’s other location services, that use Wi-Fi or cell tower signals. This selection is automatically decided with no input being required from the user.

The application can contact the Server, over the Internet, and present a list of the available surveys that the user can answer at their location. Tapping on a file upload button, on a survey, causes the application to automatically open the default camera App. The User takes a picture, and has the chance to approve the image for submission or take another one. The user may click a button to submit extra photographs as is done with a browser (they don’t need to be specified by the surveyor). The answer is geotagged and sent to the server when the user taps the submit button. The Google Play Store provides an easy way for people to obtain the application (For demonstration purposes, it was deployed there, with an Alpha launch).

The main difference between using the application and using a browser is not only with using the device’s location capabilities when the user wants to fill a survey, but also using them to find surveys. The option for an automatic service that takes into account user displacement/movement on the field was implemented. This is useful to find restricted surveys. If the user is moving around, and wants to be warned when Area Restricted surveys become available, then they can turn on an option that will automatically check for them periodically, and only if the user has moved significantly. The point is that when user comes upon an Area Restricted survey, the application launches an Android system notification that warns the user, even if the application is not being used at the moment or if the device is in standby mode. The application needs an active Internet connection to function.

Additional functionalities were implemented but not used for the demonstration. The Server can export JavaScript Object Notation (JSON, GeoJSON), containing information about currently active surveys, and with the description of the current area restrictions. This is not currently used, and is meant to explore possible future offline mode functionalities for the Android App.
4 System Architecture/Implementation

A client-server architecture is implemented (as shown in Figure 4.1) (client processes interact with individual servers, to access shared resources). The client processes are run on smartphones or personal computers. The Server has been tested on a machine running a Linux-based operating system and the Apache server software. Communication between the Server and the clients is performed over the Internet.

4.1 Overall Internet System Architecture

There are client devices that perform requests to the server, in a request-response message pattern, namely making HTTP requests. These requests can be HTTP GETs or POSTs. The Server receives the clients’ requests, and replies to them, either using web templating tools, or a serialization system to generate the responses. The Webmaster manages the Server remotely using a SSH (Secure Shell) channel.

Clients access the server’s resources in a REST architectural style, using a set of stateless operations [36]. All the component interactions with the server occur over the same single interface, using HTTP for all services and resources. All resources that are made available to users have a unique resource identifier (URI). For example, there is a base URL for file export. The last element added to that URL will be a unique identifier for a file (as in base_URL/export/file_identifier). This applies to everything in the server, whether requesting a Web page representing a survey, making a submission, downloading a photo, exporting results, or exporting Area and survey informations, and whether the resource is HTML, JavaScript, CSV, JSON, or a media file. Every survey, file resource, and Area has their unique identifier. Answer submissions are performed to a URL using each survey’s unique identifier. Some of the resource representations (the Web pages) contain links to other resources.
The interface is simple, components are separated from each other, and it’s easily scalable as it is straightforward to add more components, as they are independent from each other. It also introduces the possibility to have cacheability for some of the components. Also, some code on demand occurs, as the clients get some functionality from the Server, by running JavaScript code that the Server supplies to them.

4.2 Django Server

The Server was developed using the Django framework, which is a free, open-source web framework that is written in Python. Python (the object oriented programming language) is used almost all throughout it (in this case, Python 2.7), including the settings. The only exceptions to this are found in the web templates that were created (HTML, JavaScript, JQuery, CSS, Django template language). It is important to note that it follows the MVT [37] (Model-View-Template) pattern, which is the key to understanding how the server handles data, from the Database all the way to the end user, as displayed in Figure 4.2.

This means that there is data, stored in the Database, according to some fields and behaviour, described by the models (Python). There are views, which are Python functions that can query/modify data from the Database, and that make use of web templates (using front-end languages and also Django’s own template language (there is a built-in backend for this)). These templates are typically used by the Python views, and the views generate the responses that are actually sent to the users, or handle receiving data (by means of HTTP POSTs) from the users.

The HTTP GET requests, fetching web content generated (typically) from templates and serialized data, or POSTs for the submission of filled form values), being handled by the Python code present in views, are the key concept to understanding the Server’s functioning.
There is an integrated Object-relational mapper (ORM) that handles the translation between Python-defined models and data in a relational database. It was used. (There’s also the possibility of writing raw SQL, but this was not used). The ORM makes it possible (from the developer’s perspective, as shown in Figure 4.3) to permanently store Python objects despite using a relational database. In practice, it translates Python models and queries to SQL and vice-versa.

When a client makes an HTTP request to the Server, a URL dispatcher determines what to do based on URL configuration file that was created. (Besides Python, regular expressions (RegExes) were also used here). After going through the URL configuration, the corresponding Python view function is called, and it generates the actual HTTP response to the request.
This is achieved through using the ORM (when necessary) to interact with Database elements, and using the web templates that contain some static HTML code, and can also contain some dynamic portions. In these templates, tags were used to control flow, performing some logic. Also, Django template language variables were used, for example to replace variables with their actual values (simple replacement. These are determined by a view, after some querying to the Database, via the ORM).

It's also important to mention that Django includes an administration interface, inaccessible to users without administrator privileges. In this project, the Admin interface is where new surveys can be created and managed, as will be further detailed.

Django has some built-in security features, notably user authentication, protection against SQL injections, cross-site scripting, cross-site request forgery, clickjacking, and the possibility of using SSL/HTTPS instead of plain HTTP. Also, there is some mitigation against denial of service attacks (DoS), with the possibility of imposing limits on user-uploaded content.

Django's extensibility was crucial, as it allowed for adding functionalities through usage of third party applications and tools. The availability of the GeoDjango module was also very important, because it allowed adding geographical functionality to the project. Also, the Authentication system and Admin interface (which also are extensible) were an important asset.

### 4.2.1 Django Project/Applications, Modules

Before going any further, it's important to explain the difference between the most important and broad concepts that will be mentioned. A Django project is a collection of Django Apps. Each App is a web application implementing a set of functionalities. A project is a collection of configuration and Apps for a particular website. Each project might have several Apps, and each App can be used in different projects [37].

Loose coupling is a key feature of this project, as "for example, the template system knows nothing about Web requests, the database layer knows nothing about data display and the view system doesn't care which template system a programmer uses." [38]. There are some conventions for reusable applications, that if met, mean that third-party code can be plugged into any Django project. This makes it possible to use certain third-party tools and applications (as long as their license conditions are met). It is possible to improve the Project by plugging in new applications, and inserting new functionality into applications without having to revamp the entire Project. This means that inserting new functionality that wasn't necessarily planned out at the start of the project and evolving it is mostly always possible.

Here, one Django project was developed (corresponding to the Web Application run by the Server). At the moment, it makes use of 4 distinct Django applications (and several of Django's own modules), one of which was developed specifically for this project, and others which are third-party, open source applications available to the general public for usage. Two of the third party applications were set up/configured with no modifications to it's source files (Django-Leaflet, Python Social Auth- Django), and one was (slightly) modified (Django-Forms-Builder). A small number of modifications was used
with that App, in order to adapt some functionality to the necessities of this project (in agreement with its license). Also, various publicly available libraries are required for the project’s correct functioning.

The third party applications greatly empower the project. Having to also develop them from scratch would have taken a lot of time and effort, and, in a sense sometimes “reinventing the wheel”, when someone has done it already and made it available to others. For some leaps in functionality, this part was very important.

The main purposes of the application that was developed from scratch are:

- Integrating two of the auxiliary applications into the project (Forms Builder and Leaflet)
- Using GeoDjango functionalities;
- Establishing the main models like the Active survey model, and the model for answer submissions;
- Extending Django’s User Model;
- Customizing the Django Admin. Namely, defining how the fields are to be presented to the Admin users and their help texts, and defining who can have access to/see what.
- Using Django Forms and Widgets.
- A URL configuration file implementing a RESTful interface. Defining the patterns for the URLs.
- View functions that correspond to each of the URLs, and perform the handling of those requests
- Front-end Web development: creation of several Web templates and static files, used by the view functions to create the responses to requests.

The project settings are the most central, most important elements configured for the Server. They integrate the 4 different applications into a single ‘environment’ and specify each of their settings, as well as specifying the required Django modules. This unification of all the components means that from an outside perspective, the project can be packaged/perceived as a single cohesive Web Application, to be run on a Web server/cloud platform.

The Web Application is run and managed using Django’s (automatically generated) Python script named manage.py. It is worthy to note that despite there being the Django Admin interface (actual Web pages to interact with using a web browser) and its management possibilities, a lot of the System’s administrative tasks are performed by the Webmaster running the manage.py script. Management/maintenance duties are performed this way, from within the Server. Only someone with direct access to the Server’s system itself can do this, and non-Webmaster users can be oblivious to these proceedings.

**4.2.2 Running Django on Apache**

Django is compliant with the WSGI specification. “WSGI is the Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.” [39]. The main practical consequence of this is that “if an application (or framework or toolkit) is written to the WSGI spec then it will run on any server written to that spec” [40]. Therefore, Django Websites have a wide range of servers they can
run on. Put simply, this specification dictates how the HTTP requests will be translated into data usable by Applications.

The Django Web Application was developed locally and also tested on a Linux-based (Ubuntu) Server machine, running the Apache Server Software [41], at INESC-ID (Lisbon). Apache handles the requests from the outside, and communicates them to Django, using the Apache WSGI module (*mod_wsgi*) as shown in Figure 4.4.

![Figure 4.4: Apache Server and Django](image-url)

Django and the Python libraries were installed inside a Python virtual environment (*virtualenv*) [43]. The main purpose for the virtual environment is for the libraries to be installed separately from the machine’s standard location for Python libraries (*site-packages*). This avoids changes to the machine’s own Python libraries, which could break functionalities needed by other people who have access to it, and creates a self-contained environment for the Django project to run on.

This setup was configured in Server’s main Apache configuration file. Some settings were created for it:

- Indicating the directory where the Django project’s WSGI configuration file is;
- What Python environment to use (the configured *virtualenv*’s directory);
- Where the Django Project’s *manage.py* script is located;
- Mapping requests for static file assets and media files to their locations in the Server machine’s filesystem;
- Running the WSGI process in daemon mode and specifying a process group. The other mode (*embedded* mode) could have been used, but it implies that whenever changes are made to application code then the whole Apache Server needs to restart. In this case, if changes
happen to the code, then only the daemon processes have to be restarted instead of restarting Apache itself [42];

### 4.2.3 GeoDjango Model API and View API usage

An important feature that was used is GeoDjango. It is a Django module that makes it suitable for geographical Web Applications development, which is clearly a requirement here. GeoDjango key features are also key needs for this project [45] as it enables the usage of Django model fields for OGC (Open Geospatial Consortium) [46] geometries and raster data and provides extensions to Django’s ORM for querying and manipulating spatial data. The is also necessary in order to enable editing geometry fields from the admin interface.

Simple usage of formats that represent geometric data, like GeoJSON, would be enough, if not for the fact that the ability to perform geographical Querying to the database is necessary. GeoDjango covers that aspect.

GeoDjango can run over different databases, and MySQL is adequate for the purposes of this project. The map Areas are stored as GeoDjango Polygon fields [43]. In order to Geo-Query the Database, the `Point` field is used (simple point on the map, coordinates). The spatial lookup that was used in this project is `contains`, made available by the View API [44], `Contains` determines if a geometry field actually contains some lookup geometry. In this case, it corresponds to MySQL’s `MBRContains` [45], but no actual raw SQL had to be written, just Python, as GeoDjango’s ORM takes care of the “translation”.

GeoDjango allows for various different `SRIDs` (Spatial Reference System Identifiers) for geometry fields. The default, current standard `SRID` (4326) was used, which corresponds to the World Geodetic System (WGS) number 84 [46] (which is the latest revision, and was lastly revised in 2004). This is the standard used in the GPS system.
4.2.4 Database Architecture

The above diagram (Figure 4.5) represents the main Django Python models that were used (these correspond to actual Database tables). Several models (tables) have been omitted from this diagram: form builder application models, the social authentication models (with the sole exception being the AbstractUserSocialAuth model, in the bottom right of the diagram), and also models from Django’s sessions module, content type module, and sites module.

Some of the models have customized saving behavior, and the Profile model makes use of Django signals [47] to get notified of the creation of new users.

The two fundamental models (and corresponding Database tables) in Figure 4.5 are Active_survey (top left) and User (center left). The various models can be described as follows:

The Active_survey model has relationships with:

- Submitted_Answer (each survey might have multiple answers, and each answer has one related survey);
• *Area* (each survey might have any number of Area restrictions, and Areas don’t necessarily have to be related to any surveys);

• *Restricted_Form* (surveys can have one template indicating allowed users and groups or none at all);

• *User* (the surveyor responsible for the survey);

• *Form* (referencing the form object from which the HTML fields, name, and identifying string *(slug)* will be extracted and saved in the survey).

The *User* model’s foreign relationships span almost all the other main models of the project:

• Twice related to *Restricted_Form* (representing the owner of a restriction template, and also any number of users that might be included in the restriction);

• *Group* (denoting the group’s members);

• *Permissions* (besides the possibility of attributing specific individual Administrative permissions to Admin user (non Superuser surveyors) groups, it is also possible to do it for each individual person);

• *AbstractUserSocialAuth* (in practice, this extends the base Django auth with the social authentication capabilities, providing *superusers* (full Administrator permissions) the user informations obtained from social auth providers (Google, Facebook));

• *LogEntry* (Logging Administrative actions, which applies to administrators and surveyors);

• *Profile* (every user will have one. This is meant to add extra custom information to the user model without changing the original model. Currently just a counter for the number of total answers by each user).

The relationships between the models are established using the model relationship API [53].

Running a MySQL Database, the Django Server requires drivers to implement a Python Database API for MySQL. The Python *mysqlclient* [54] interface was used to solve this problem.

### 4.2.5 Integrating third party administrative tools

Open source third party tools were used to implement some of the functionalities that the project makes available in the (Django) administrative area. These were integrated in the project’s configurations, and made available for surveyors and administrators to use.

**Interactive Maps**

Map visualization is a crucial requirement. This imposes the need for mapping software. Leaflet [49] is an open-source JavaScript library for interactive maps. It was used to implement mapping functionalities for the Django Web client, as shown in Figure 4.6. Leaflet was used to provide an interactive map, to get map Area shapes, and also to provide the ability to select a certain point on the map, and use it for geo-querying purposes. Leaflet allows for some customization of the maps, including choosing the map layers to be displayed.
Django-Leaflet [50] (a Django app) provides the embedding of Leaflet in Django projects. One of the advantages that made implementing this important is the possibility of using a Leaflet widget in the Django Admin. This is useful, as it allows the Geo surveyors to easily delimit areas of the map by clicking on it, as shown in Figure 4.7, to be stored as Areas of Interest for surveying (this is done in the Django Admin section). This Leaflet widget has been configured so that various distinct map layers (from Mapbox, which is a provider of custom maps, and an alternative to OpenStreetMap [49] and Google Maps [50]) can be used interchangeably. A single layer was used in the JavaScript code provided by the Web application to a Desktop user checking for surveys on locations (a satellite layer).

Further configurations were set, in regards to the default presentation, behaviour, and also the default location of the Django Leaflet widget. Mapbox [48] is a map layer provider (and also gives the users some functionalities such as the ability to modify and create their own maps, but this was not done). It was used to provide the five map layers used by Leaflet, of which two use satellite imagery,
and the rest are maps (different colors showing streets, land, and water). It can be easily substituted by other map layer providers.

**Form Building**

Another particular requirement for the flexibility of surveys, in terms of the ability of creating new, custom ones with ease is the ability for the surveyors to choose specific fields for each specific form. For example, one survey might have just a text field. Another might need to have a dropdown with choices and a file upload button for an image. Since surveys can have multiple fields, the surveyor needs to be able to choose specific fields for a survey with ease.

To solve this, a Django App named Django-Forms-Builder [49], which provides the ability for admin users to create their own forms within the admin interface (shown in Figure 4.8), drawing from a range of field widgets such as regular text fields, drop-down lists and file upload fields, was used. Apart from basic configuration and installation of this application in the project, some further steps were taken.

![Form building, with a title, introduction, and three fields (questions/requests)](image)

One desirable, very important feature that the project must have is the ability to store the geographical coordinates where the surveys were answered by the users.

Django-forms-builder allows for the addition of custom fields and widgets. Using Django’s Forms library, a multi-value field was created, with two floats, (one representing latitude, which can range from -90 to 90 and another representing longitude, which can range from -180 to 180) and a compression function, that can create an object to represent the coordinate pair, as a GeoDjango Point. Also, a widget to be used in templating, was created that splits the Point input into two latitude/longitude fields. These are supposed to be rendered as two hidden HTML form input fields.

Initially, all features of the Django-forms-builder application were used, but this made the project too dependent of it, and therefore some flexibility could be lost. This application can not only generate forms, but handle their answer submissions and exporting the answers. Not much can be done to
adapt these application features to this Project, as it would require a significant overhaul to the App, which is undesirable.

In order to gain some flexibility, and leaving the possibility for, in the future, substituting this form building application with alternative form building applications/mechanisms, some decoupling was done. This means that instead of having this application build the forms, serve them, handle their submissions, and handle result exports, the only thing the application is currently used for is the actual form building. The other tasks are handled by the main Django App, that was developed specifically for this project.

When a surveyor is creating an Active survey, a button to create a new Form is available. This opens a window with Django-forms-builder functionality, and the surveyor builds a Form. When "save" is clicked, the main application extracts a string with the Form's HTML, from the forms-builder App, and stores it.

### 4.2.6 Google and Facebook OAuth2 over Django’s auth

Recently, integrating social functionalities into services and even games has become very common. Giving third parties outright access to everything in the user’s accounts would be highly unsafe and undesirable, so protocols like OAuth, OAuth2 and OpenID are used by providers.

OAuth2 is a protocol used for user authorization. As explained in RFC 6749 [50]: “The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf”. Further introduction to OAuth2 is given as: “In the traditional client-server authentication model, the client requests an access-restricted resource (protected resource) on the server by authenticating with the server using the resource owner’s credentials. In order to provide third-party applications access to restricted resources, the resource owner shares its credentials with the third party”.

In essence, OAuth2 (and its predecessor, OAuth) are aimed at allowing third party entities access to some specific (and limited) part of a user’s information, from an authentication provider, as illustrated in Figure 4.9 (this is not restricted to social networks, as services such as Amazon and Paypal have also adopted it, but most of the social networking platforms such as Facebook, Twitter, Instagram, and LinkedIn use it). Facebook and Google were used in this project.

![Figure 4.9 Oauth2 intervenients](image)
In the Google OAuth2 scenario [54], before an application can access private data using a Google API, it must obtain an access token that grants access to that API, as shown in Figure 4.10. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits [55].

This token can stop working for various reasons, and these vary from provider to provider. In Google’s case, one of them is the token being revoked, over 6 months of no usage, or excessive usage (hitting a limit of requests per account). Also, there are some restrictions, in cases where the user’s password was changed.

By default, Django accepts users to login using accounts that were specifically created for it, either by Administrators or via account creation (signup) Web pages. It is desirable to also have the possibility of user logins using social network accounts.

The user sessions should be maintained just as they would with a “simple” Django user account (also using the default authentication backend). Session maintenance is done using Django’s authentication middleware [51] (middleware stack shown in Figure 4.11), and makes it so the server can react to requests differently, based on the user status. The objective is to extend the authentication system to use OAuth2.
Open-source solutions that do this already exist. So, another application was used in this Django project to help with this. It is called Python Social Auth- Django [52], and it provides the ability to configure new authentication backends for a Django Website. It is an extension of the Python Social Auth (Core) "ecosystem".

Two providers were configured (Google and Facebook (both using OAuth2)). A Key-Secret pair must be obtained for each (it's necessary to do registration as developing an app, in order to obtain those from the authentication providers). Also included is Middleware for dealing which exceptions that might arise, having to do with the social authentication app, that was also added to the project.

The Credentials for Google OAuth2 are configured for a specific Google (also a Gmail account) account. The configurations can be accessed using Google's Cloud Platform Console. This also needed activating the Google+ API for this project. The Credentials for Facebook Oauth2 are configured for a Facebook account, also representing the Server, and its configurations are set in Facebook's platform for developers.

It was important to define a scope for those authentications. This means defining the scope of information that the users will be conceding the third party access to. There is a multitude of possible options to add, for each of the authentication providers. For testing purposes, the scope was defined as just the basic user profile information (name) and email, both for Google and Facebook. In light of the authentication protocol, the login is not done directly on the Django Server, as this would imply security issues, and outright giving the Website full access to their account. The user is redirected to a Google/Facebook URL to perform the login.

If the users try to login for the first time with their account, they are also redirected to a Google (Figure 4.13) or a Facebook (Figure 4.12) web page mentioning what the third party is asking access to, and asking the user if they want to accept and provide it, or not. After this, there's redirection back to a specific URL from the Django server (this URL is easily configurable). Incremental authorization was not necessary, as only basic information is ever required.
It is possible to let anyone log in using this. In that case, a first time login with a Google or Facebook account will automatically create an account in the Django Server that stores the info obtained from the authentication provider. This removes the problem of either the admin having to create accounts for new users, or making the users create accounts specific to the server.

On the Django Admin interface, the user information, (that was provided by the authentication provider) can be viewed by Superusers (administrators) as shown in Figure 4.14. An administrator then has the ability to insert the new users into certain user groups (that correspond to some set of privileges), or specify which permissions will the user have/don’t have, one by one, just as they would with a “simple” Django user, created with the default backend. Login via Google/Facebook has been tested on personal computer browsers.

### 4.2.7 Execution Flows

The server software has the task of handling user requests, and either applying changes in the database, or retrieving information from it. In some cases, web templates are used, and in others serializers are used, both making use of information queried from the database, in order to generate the responses that are sent back to the users.
Survey Presentation

The website’s main page menu (Figure 4.15) allows users to see a list of all Active surveys they have permission to participate in (not regarding location), as shown in Figure 4.16. It also allows them to check for availability for specific locations (by clicking on a map, shown in Figure 4.17), and to submit surveys for those locations, using a web browser (in a mobile setting, the location is obtained via smartphone location services, instead of indicating positions on a map).

![Main User Menu](image1.png)  ![Survey Listing; Button leading to the Map](image2.png)

Figure 4.15 Main User Menu  Figure 4.16 Survey Listing; Button leading to the Map

This check for surveys for a specific location must obey restrictions (Figure 4.18). Surveys might have no restrictions at all, and be accessible to everyone. Surveys might be available everywhere, but to a restricted audience (as in Figure 4.19), or be available to everyone but only on certain areas. The most restrict surveys have both types of restrictions.

When the Server gets a request for surveys, evaluations are necessary to come up with the list of surveys that a specific user (the client making the request) can answer at a location (geo querying, see Figure 4.20). Restrictionless surveys don’t impose a challenge, but on the other hand the Server needs to present restricted surveys only under correct circumstances.
A Web Page is generated by the Server, using front-end technologies that are illustrated in Figure 4.23, and presented to the User, as shown in Figure 4.21, listing a button for each survey that the user can answer given current conditions. By clicking on a button, the user makes a request for another Web page that presents the survey itself.
The user can fill the survey and submit it (Figure 4.22). The coordinate fields are filled automatically, using the jQuery [53] JavaScript library. The user might want to submit more than one picture, so jQuery was also used to provide a button whose function is inserting an extra upload field. Django template language tags were used to insert variables, and loading the CSS presentation.

JavaScript and jQuery constitute mobile code, provided to the user devices to perform specific tasks. In this case, DOM [54] (Document Object Model, containing the document’s structure) manipulation is performed, and HTML fields are altered. On the other hand, Django’s template language is used by the Server (interpreted using Django Python views) to build the web pages that are sent to the users, as shown in Figure 4.23.
Answer submission and storage

The Server is able to receive form submissions, where the form can take any shape. Forms are flexible, and different Forms can have different fields from each other. These may be text-based fields, and files might also be submitted. The server needs to parse submissions and store user inputs appropriately.

The submissions are performed using a base URL, with the addition of a slug (text identifier) that distinguishes each corresponding survey. This allows the Server to register what survey each answer corresponds to. The Web page presenting the survey, that the user fills, is generated using Django’s CSRF token system for security [53].

HTML forms provide various methods for encoding. The answers are posted using the `multipart/form-data` encoding, which, according to the W3C (World Wide Web Consortium) [54], “should be used for submitting forms that contain files, non-ASCII data, and binary data”.
When the Server receives an Answer, the text fields are parsed by a (Python) Django View, which is programmed to handle the answers as shown in Figure 4.24. There might be fields with a list of corresponding values, in situations where surveyors use *multi select* fields or multiple checkbox fields. In those cases the answer is inputted as a list of the selected values. Some parsing is done, in order to compress those values in a single readable string, to be stored. All the (compressed) fields are added to a Python dictionary (key-value pairs).

If the Answer contains files, these are stored in the Server, in the file system. Each file is made available for download, using a unique identifier. After the file is stored, its download URL is added to the dictionary, so that the surveyor and the participant know exactly where they can retrieve it.

The dictionary that’s built containing all the keys and values, for both text fields and files, is serialized as a JSON (JavaScript Object Notation) [55] string and stored in the database, along with the coordinates, date of submission, and the user who answered. Only the surveyor, and the person who submitted the answer, will be able to access/export that answer.

### 4.2.8 Data Export

Survey answers can be exported in CSV format text files, that store tabular data in plain text. As Answers are structured in a tabular fashion in the file, each line represents a single Answer. Python’s CSV module is used to create the file, row by row, using Answer values queried from the Database. These CSV files can be opened and displayed using spreadsheet programs. There are two distinct ways for the answers to be exported, either via setting export parameters on a Web page or via a Web Service (using URLs).
The first of those ways consists of the surveyors being presented with a simple form, shown in Figure 4.25, to choose a baseline date for their answer exports (i.e. only exporting answers submitted after the chosen date), for each of their surveys. By clicking a button, a simple form with the chosen date and survey identifier is submitted, and as a consequence, the server responds by creating and responding with a CSV file containing the corresponding answers, to the user.

The other way (Web Service) consists of a base URL for exports. By appending a survey identifier (the slug) to that base URL, a request will result in the Server responding with a CSV file with all the answers for that survey. A date can also be appended after the survey identifier, to specify the baseline date for the export. As such, the request will be made to a URL looking like \text{base\_export\_url/survey\_slug/date}, with the date formatted as \text{dd-mm-yyyy} (d-day, m-month, y-year). The date is optional, but the survey identifier is mandatory.

The export takes into account who made the request. If the request was made by the surveyor, then all the answers are exported. If it was made by other users, then only their own answers are exported.

File downloads (files that were uploaded by participants) are also made available a service. They are served on a specific URL, with a unique file identifier appended to it (as in \text{base\_download\_url/file\_identifier}). Only the surveyor, and the participant that submits a file, have access to the file identifier.

Another two services were tested: Area export and survey export. The idea, shown in Figure 4.26 is to open the possibility of exporting information about Areas and surveys, in a way that the mobile application can save them (using non-relational storage) and introduce some offline functionality.

Django’s serializers were used. Surveys can be exported as text, using the JSON format (key-value pairs). The various components of a survey are serialized into JSON. A JSON string can contain other JSON strings in it, as values. This way, information about several surveys can be exported in a single JSON string, that has other JSON strings in it, with each of those corresponding to each survey.
Area exports require support for the Polygon geometry type, which GeoJSON [56] provides. Django's standard serializers include support for GeoJSON.

Figure 4.26 Preparing for offline application evolution
4.3 Android App

4.3.1 Android Operating System

The idea behind Geo Crowd Surveys implies the possibility of getting survey submissions from anywhere. It’s highly desirable to be able to associate survey answers with actual locations on the map. If this is to be applied to the general public (and not some strict community, such as the scientific community), people need an easy-to-use, preferably cost-free, comfortable way to participate.

The recent proliferation of smartphones makes all those requirements relatively easy to achieve. Smartphone users can download an application easily. As long as it is not too complicated and follows general rules of common sense for application design, users will most likely find an application easy to use. It is very comfortable and convenient to use applications that provide some utility, from checking the weather, to getting email, to checking a bank account, to ordering some service or item, etc., using Internet-connected smartphones.

A smartphone application giving people the opportunity to report on something near them, providing some information, even some photographs and automatically giving the exact location, is an easy concept to grasp for a user. Filling a survey like that should just take a little effort, a few taps on the devices people carry with them most of the time, to do it.

There’s a trade-off between the costs of recruiting/instructing/supplying material for people to participate in the typical surveys of old, and the cost of developing adequate software for the new types of surveys. These have the advantage of not having costs rise in direct proportionality to the number of users, because as long as Server capacity is adequate, all it takes to acquire a new participant is for them to install a mobile application which is free.

Currently, the smartphone Operating System market share is split between Android (developed by Google) users, and iOS (developed by Apple) users. There’s also a very small share of Windows Phone users, and an even smaller amount people using other mobile Operating Systems.

In order to demonstrate a working solution, consisting of a Geo Crowd Surveying smartphone app, a choice needed to be made between developing an Android application or an iOS app. The inherent advantages/disadvantages of iOS and Android are off the topic of this thesis. The decision was made based on convenience/personal preferences. A preference for developing using Java, using Android Studio, was the key factor in the decision (iOS applications can also be developed using various programming languages, with Swift (Apple’s own programming language, based on Objective-C) being the recommended one).

4.3.2 Application Development

As the name suggests, Android Studio has the specific purpose of developing Android applications. It is based on the IntelliJ Java Integrated Development Environment (IDE) [57]. The Gradle [58] build automation system was used, with the ProGuard [59] Java bytecode optimizer. Furthermore, ProGuard provides protection against reverse-engineering by obfuscating Java code.
The Android population is significantly split in terms of the user share for the different Android operating system versions, as shown for 2016 (Figure 4.27) and 2017 (Figure 4.28). It’s usual for older devices to only have upgrades available up to a certain system version, and this leads to a significant percentage of the user population still using older Android versions. Newer medium and top end devices come with the newer operating system versions, and it takes a while for the newer versions to build user share.

![Figure 4.27 Google’s estimation of user share per system version, as of August 2016](image)

![Figure 4.28 Google’s estimation of user share per system version, as of September 2017](image)

The Google Android team releases Support Libraries that guarantee some level of backward compatibility, as they allow some newer functionalities to be implemented in older Systems. But version fragmentation has the consequence of there being a possibility that some API calls, or some interface elements being available on some APIs, but not on others, therefore producing ranges of versions where some calls will work, and some won’t.

Android Studio includes an Android Emulator. This was crucial to test the application’s functioning, simulating various devices that are available in Google’s repositories. Several Android Virtual Devices (AVDs) are available for download. As explained in Android’s developer support pages, “An AVD contains a hardware profile, system image, storage area, skin, and other properties.” (...) “The hardware profile defines the characteristics of a device as shipped from the factory.” [60].

This turned out to be important. More precisely, because of a callback function that defines how the application reacts, when the user taps on a “Choose File” button, on an HTML form that is shown in a Webview. It is important for the App, and it suffered variations, with changes being introduced in operating system versions 4.1, 4.3, and 5.0. Thanks to the Android Emulator, testing the behavior of the application was possible on different AVDs, running different Android versions.

Several libraries were used. Only one of them is not a standard Google/Android library, or standard Java library. It is named Okhttp [61], and it’s purpose was to provide a request/response API for HTTP
requests (the Okhttp library does not require external downloads, as it can be found in Android Studio’s own library repository).

Also, it is necessary to make decisions regarding OS version support, namely defining the lowest and highest target APIs for an app. The lowest supported API for the application to function is 15, corresponding to Android version 4.0.3 (which was released in October of 2011).

### 4.3.3 Application Architecture/Main Components

The Android Application’s architecture is composed of four Activities, and one Service. Various **Intents** are used, and data storage, that establish the interactions between those components, as shown in Figure 4.29. Before presenting the App’s architecture, the formal definitions for these concepts are transcribed, to clarify what these Android concepts (Activity, Intent, and Service) mean:

- “An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to startActivity(). The Intent describes the activity to start and carries any necessary data.” [62].
- "A Service is an application component that can perform long-running operations in the background, and it does not provide a user interface. Another application component can start a service, and it continues to run in the background even if the user switches to another application. (...) a service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background." [63].
- “An Intent is a messaging object you can use to request an action from another application component” (Goes on to explain that those cases are starting an Activity, starting a Service, or delivering a Broadcast) [64].

The activities’ theme and presentation are set in their respective layout files (written using the XML language). The activities, intents, and the background service were implemented using the Java programming language.

Data exchange between the different components is performed using Android’s **SharedPreferences** [65] key-value pair data storage class.

There is one Activity (that is started as the application is started) that serves as the welcome screen for the App, and that allows the user to turn a background service on or off. The second Activity is used to display online content (webpages) from the Server, within the App. A third activity is used by the user to force the application to update the current location with the highest precision possible, and to manually search for surveys for the user’s current location. There’s a fourth Activity with the sole purpose of defining a custom base class for the other Activities (they extend it). The idea behind this fourth activity is for the other activities to have identical look, namely having an “action bar” at the top of the screen, and presenting an icon on it, that pops up a menu.
4.3.4 Home Activity and the Menu

The home activity is where the Application starts off. It checks if the user has already given the application the necessary permissions. Specifically, to access the device's location information, and permission to use the camera and store images (taken with the camera). This is necessary for the application to work properly. If the user hasn’t done so already (this typically occurs when downloading the application from Google’s Play Store), then a pop-up prompt will appear.

The home activity is simple (Figure 4.30). It has some basic information, and one Switch that controls the Background service (turning it on or off as in Figure 4.31). When the user toggles the switch, a callback is invoked. Depending on whether the user is trying to turn the service on or off, an
Intent is launched that either starts the background service, or stops it. The home activity extends the “Menu Activity”, as do the other activities.

One of the Activities has the purpose of creating the action bar and pop-up menu (shown in Figure 4.32) that the others use. It also extends an Android Java class, namely AppCompatActivity. This is required as the base class for activities that use the support library action bar features [66].

The menu works in a simple fashion. The menu items were defined in an XML file. The menu is created with a MenuInflater [67] object (provides an actual object, generated using the XML file). When the user taps on an item, a handling function is called, which creates an Intent to start the corresponding activity. Then, that activity is launched, and it appears on screen.

4.3.5 Background Service and Location

Surveys can be everywhere on the map, or restricted to certain areas. If the participants move around, they might have interest in finding out if new surveys become available. Besides just manually searching for restricted surveys, an alternative was developed.

The idea for the background Service is to have it automatically check for restricted surveys, and notify the user when these become available. Just checking for surveys based on some time period having passed might not be the best solution for the app. If the user has not been moving around and just stays in the same place, or moves little, that would mean an unnecessary use of the user’s Internet connection, to make a request to the server.

The practical solution to this implied the application taking movement into account to decide when to make a new check for surveys. The Service should only makes requests to the (remote) Server if the user has moved over a minimum distance, away from the last position where a check to the Server for surveys happened (Figure 4.33). It is possible for displacements to happen, without provoking another request to the Server. A new request may happen every few seconds, and it will only occur if the user’s distance to the spot of the last request becomes greater than a minimum distance. For demonstration purposes, it was set at 50 meters.

![Figure 4.33 User movement, and background survey checks](image)
Google provides the basis for a solution, by way of their APIs, namely using Google Play Services and the Fused Location Provider API [70]: “Rather than directly enabling services such as the device’s GPS, your application specifies the required level of accuracy/power consumption and desired update interval, and the device automatically makes the appropriate changes to system settings”.

Google Play Services, that include the location services, are background Android services. It is important to mention that since Android 4.0 (October 2011), Android devices automatically update Google Play Services. (These updates bring some new functionalities to older devices without the necessity of updating the system version, and bridges some gaps in functionality stemming from the fragmentation of versions that was mentioned before). These functionalities were accessed using the GoogleApiClient.

This way, it was chosen how and when location updates should be made. An interface was used to decide what happens when a location update occurs. The two most important concepts to implement this type of solution are the LocationRequest [68] data object, through which parameters for location updates were specified and also the LocationListener [69] interface. When there is a location change (according to the conditions that were requested), this interface is notified. When this happens, it’s time for the application to make a request to the Server to ask for information.

4.3.6 Background Service and the remote Server

When a location update occurs, an HTTP GET request is made by the Android device, to the Server. An auxiliary Java class for short background tasks was used for this purpose.

This auxiliary class was then configured to use the OkHttp Android Client to perform the request, and storing the HTTP response. If it is determined that the Server’s response is negative, then nothing happens. If the response is positive, then the user needs to be warned. This is done by launching a

Figure 4.34 Notification while on the Settings App

Figure 4.35 Notification over the Home screen
heads-up notification that is shown even if the user is not using applications/using another application at the moment.

To achieve this, a notification is built. Its characteristics are defined (Icon, Title, Text, Priority) to choose what it looks like/it’s behavior when shown to the user.

It is desirable for the notification to be able to “work from anywhere”. This means user may get notified while using other applications (Figure 4.34), from the Home screen (Figure 4.35) or from the Notification manager. The notification will also work if the device is standing by (with the screen turned off). In that case, the notification sound will be played.

The user can see the notification, and can tap on it to discover what surveys are available. If the user taps on the Notification without the Geo Crowd Surveys application being the one that’s currently being used, then it is opened.

### 4.3.7 Manual Check for Location/surveys

It is also possible for the user to manually look for surveys (Figure 4.37), without using the background service. This activity shows the user their current coordinates, also obtained using the Google Play Services. These are updated every second, with the maximum precision possible, and the user gets to know if location is working or not (small movements will make it change, and the rightmost decimal digits typically tend to fluctuate).

The user can tap on a button to see the survey list.

#### 4.3.8 Interacting with Web content

When the user taps on a notification, the application comes up, on another Activity that was developed. The practical purpose of this Activity is to display web content that shows a list of the available surveys on their location (Figure 4.38). This is the same web content that would be shown on
a browser. The user can then pick one (Figure 4.39), and it is requested from the Server, to be shown in the same page. The user can then fill the HTML Form (survey) and submit it (Figure 4.40).

The key concept used to accomplish this is Android’s Webview Class [77]. The activity itself will request and load the content from the Server and place it inside the Webview. The URL for the request to the server is retrieved from the SharedPreferences data storage, used to interchange data between different components. It should have been placed there, with the most current location, obtained either by the background service or the manual update activity.

The Webview’s settings are important, in order to handle web content. First of all, JavaScript execution is necessary and enabled. File access was also enabled (because of the need to take pictures with the Camera and store them).

If a loaded web form has a file upload button, the application must react to it by suggesting opening the Camera App, or possibly chose a picture that the user has already taken (Figure 4.41). After the user takes a picture, and approves it for submission (Figure 4.42), the Geo Crowd Surveys application should come back up, and the picture should be also stored in the device (this is done by default by the Camera App). The way to perform this has suffered changes over time, most notoriously with Android 5.0 (released in 2014), and this has to be accounted for so earlier versions also work.

The Webview is also used for user Authentication. Users are required to be logged in so that they can answer surveys. Users should already have an account (created by an administrator, or created automatically via socialauthentication using a desktop/laptop browser).
Uploading extra files is possible, just as it is on a browser. Tapping the *Extra Photo* button (Figure 4.43) will create a new file upload field. When the user taps that field, it will behave just as a file upload field that would have been set by the surveyor would behave, and present a menu to choose between taking a photo using the camera App or uploading a file that already exists, and is stored in the device.
5 System Evaluation

To evaluate the System, it is crucial to validate its functionalities and to compare it with other relevant systems that present alternative ways to tackle surveying. It is also important to present feedback from people who are interested in using these types of systems to solve their surveying problems. An evaluation of the resulting system, based on comparisons and obtained feedback is presented in this section, along with an evaluation of the mobile application’s characteristics.

5.1 Functionality validation

In order to validate the system, it’s important to review it’s key functionalities, and to verify the whole system’s compliance regarding its objectives. A system solving the entire surveying process, from survey creation by surveyors, to exporting the received answers, was implemented and demonstrated.

Form Building and Geographic Restrictions

The two key interfaces in the whole system are for form building (Figure 5.1) and area creation (Figure 5.2). These allow surveyors to create custom forms according to their surveying needs, and to have the option of deploying those surveys either on the entire map, or just inside multiple regions on it. Flexibility is provided to the surveyors, along with ability to adapt to different situations without needing any developer help, and resources that grant GIS capabilities. These graphical interfaces are easy to use, and do not require the surveyors to have any special skills in order to use them, which is an important requirement in order to make the system as accessible as possible.
User Survey Restrictions and Deploying Surveys

Another key ability for surveyor is to restrict forms to user and groups (Figure 5.3), which can be a very useful control tool. This is due to enabling surveys be private and inaccessible to some people, but it also has the consequence of allowing different surveying crowds to share the same server without interfering with each other, and without surveyors having to worry about overwhelming diverse crowds with information that’s irrelevant to them. The survey deployment interface (Figure 5.4) allows the surveyors to customize how their surveys are deployed, which can be changed over time, if the surveyor wants to make modifications. The surveyor can easily access a list presenting their deployed surveys, as the administrative interface is simple and intuitive to use and navigate for surveyors and administrators.

Automatic Survey Search, Survey Lists, and Answer Submissions

Automatic Survey searching is crucial to demonstrate how survey participation can be made easier, as participants do not have to manually search for geographically restricted surveys. This type of functionality also has the advantage of giving users the sense that the surveys overlap the real world, and that as the user explores their surroundings, survey participation opportunities can emerge and pop-up (Figure 5.5) on their device. Along with social authentication, this functionality contributes to GCS’ suitability for crowd surveying purposes.
The interface to answer surveys (Figure 5.7) and the way they are presented (Figure 5.6) make it very easy to answer them, and provides some flexibility, as the user can effortlessly submit multiple photos at once, if needed, without having to answer the same survey more than once.

**Answer Export and Photo/File Export**

![CSV file containing answers, opened in a spreadsheet program](image)

GCS also fulfills the requisite of not only providing surveyors access to their surveys' results, but it also allows participants access to all of their contributions. Figure 5.8 shows an example of an answer file exported by a surveyor, that contains answers from two different users, shows when they were submitted, along with their geographical coordinates, and shows the survey's fields (in this case, a single dropdown field) and respective answers. In this situation, extra file upload fields are demonstrated. These show up in the columns to the right of the survey's fields, and contain the URLs where each file can be downloaded, from the server. As a user submits more files, these will be placed in the rightmost columns of the CSV file. Participants can export the answers they've submitted, and in that case the server generates CSV files that contain only their answers, not including other users' answers. Exports can have an initial date, in order to omit older answers that don't need to be exported.

### 5.2 Comparison to other similar systems

It is important to compare the resulting system (GCS) to the other surveying systems described in section 2.4. Only systems that check minimum requirements, and are currently relevant were included. For example, the Social.Water project was not included for having been rendered obsolete by current systems, and Amazon’s Mechanical Turk was not included because of its paid crowdsourcing system, where participant contribution is financially compensated.

Table 5.1 shows a comparison to the other similar systems. A free form builder/surveyor system (Google Forms), a simple system of distributing PDF forms and submitting answers over the Internet, three commercial systems (the geo-enabled Survey123 and LimeSurvey systems, and the consumer/data analysis focused SurveyMonkey platform), an open source field data collection tool set (Open Data Kit), and a GIS application (Mappt) were analysed for this purpose.

Mappt is an example of a modern GIS-enabled application that packs a lot of geographic functionalities. When used for surveying purposes, the task is not simple as participants need to export files from the application and send them to the surveyor. In a situation with a group of a few
participants it may be viable, but as the number of participants increases the surveyor’s tasks become increasingly tedious and exhausting, which makes this application unusable for crowd surveying.

The cloud solutions have to upload and store user data and survey data in cloud platforms. This might be concerning if the surveyors need guarantees of data privacy. Some recent examples of cloud platform intrusions make this a sensitive topic. Possibilities that allow for running on a single server have the advantage of the owners being in control of their data. This is the situation with Geo Crowd Surveys. Nonetheless, it is also possible to run GCS on cloud platforms. In terms of Cloud storage privacy concerns, Geo Crowd Surveys is not troubled, and neither are LimeSurvey and Open Data Kit, as they allow users to deploy server instances using their own server machines.

Cloud Platforms incur regular costs. The solution comes ready for use, but in the long run the surveying entity will accumulate subscription costs to pay. On the opposite end, using PDF forms and Google Forms is totally free. In terms of regular expenses GCS and Open Data Kit fare well, with no costs beyond preparing the infrastructure. Ready-to-use platforms (with the presented exception being Google Forms) always have these costs. ESRI’s Survey123 has costs related to the amount of data that is stored, LimeSurvey and SurveyMonkey both have costs related to the number of survey answers that are collected (and, in the case of using an owned server, LimeSurvey has costs related to update access). The Mappt application has a (monthly or yearly) subscription.

If the participants don’t get authenticated, then ill intentioned users get some openings, through which they can harass surveyors/spam bad information. Google Forms doesn’t authenticate the participants, and neither do simple PDF form-based surveys. Survey123, LimeSurvey and Open Data Kit solve that problem and can authenticate users and identify answers, but only with dedicated local accounts. GCS and SurveyMonkey allow for authentication with social providers, making it easier for people to participate. This makes all the difference in terms of crowdsourcing possibilities, in situations where people don’t need specific skills and everyone can help. Using GCS, an institution has an advantage in terms of ability to garner the public’s support and taking advantage of a crowdsourcing scenario. All it needs to accomplish is making the population who cares for their cause to notice that they can help with their smartphones. With social authentication, people can easily identify themselves and participate, knowing that their efforts (the answers) are registered as being theirs. A surveyor can easily find the person who provided an answer on Facebook and get in contact with them. This process enables participants to get the inherent rewards from helping a cause they deem worthy without major hassle or even the need for geographical proximity to the institution that is performing surveying, and its personnel. Adding Social networking functionalities is also possible, since the base mechanisms for social authentication have already been implemented and providers make somewhat developer-friendly APIs available to the public.

Offline functionalities are an advantage that lessens the participant’s burden (namely using their mobile data). GCS does not provide offline functionality at the moment. The more commercial/consumer research-minded systems (SurveyMonkey and LimeSurvey) also don’t have it, but Survey123 and Open Data Kit do.
<table>
<thead>
<tr>
<th></th>
<th>Data Storage</th>
<th>Costs</th>
<th>Participant Auth</th>
<th>Social Auth</th>
<th>Offline Funct.</th>
<th>Mobile Apps</th>
<th>Browser Answering (via Web Interface)</th>
<th>Form Building</th>
<th>Conditional Forms</th>
<th>Survey Location</th>
<th>Map Areas</th>
<th>Users/Groups Restrictions</th>
<th>Openness</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geo Crowd Surveys (GCS)</strong></td>
<td>Own Server</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Android (Web)</td>
<td>Yes</td>
<td>Yes</td>
<td>Possible Evolution</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Google Forms</strong></td>
<td>Cloud</td>
<td>No</td>
<td>Anonymous</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Possible Evolution</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>ESRI Survey123</strong></td>
<td>Cloud</td>
<td>Yes</td>
<td>Local Account</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>PDF Forms</strong></td>
<td>-</td>
<td>No</td>
<td>Anonymous</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>SurveyMonkey</strong></td>
<td>Cloud</td>
<td>Yes</td>
<td>Oauth</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>LimeSurvey</strong></td>
<td>Own Server</td>
<td>Yes</td>
<td>Local Account</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Map</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Open Data Kit collect</strong></td>
<td>Own Server</td>
<td>No</td>
<td>Local Account</td>
<td>Yes</td>
<td>Android</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Auto/Map</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Mappt</strong></td>
<td>App</td>
<td>Yes</td>
<td>No Surveying</td>
<td>Yes</td>
<td>Android</td>
<td>No</td>
<td>-</td>
<td>-</td>
<td>Auto/Map</td>
<td>Yes</td>
<td>-</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 5.1: Comparing the result with similar available systems/solutions
Mobile applications represent another advantage, as using smartphone location capabilities means users don't necessarily need to pay attention to where they are exactly, on the map, and an interesting way to reach participants. Some platforms allow users to fill surveys via web content, using a browser, which makes them accessible to a wide range of devices that can surf the web. GCS has an Android App, and iOS users can also participate, albeit using an Internet browser. SurveyMonkey and Survey123 have applications both for Android and iOS and also support browser users. Google Forms and LimeSurvey don't have mobile applications but allow users to participate via browser. Open Data Kit users need an Android device to answer surveys.

Form building is a must-have requirement for surveying, as without it surveyor would be limited to some set of pre-made forms. Conditional forms (with questions that appear based other questions' answers) may come in handy in some situations. All the presented surveying systems have form building capabilities. They have (relatively) simple interactive menus, with the exceptions being PDF files, which need to be manually constructed, and LimeSurvey, which has a complicated user interface for surveyors that's not user friendly. Conditional forms were not a main concern for GCS, but they should be possible to achieve using different form building strategies.

All Systems except GCS have to rely on the users to know where, on the map, surveys are pertinent and need answers. Survey123, LimeSurvey, and Open Data Kit rely on users reading survey descriptions for that. Having Area restrictions gives GCS a significant advantage if the system is to be used in a crowdsourcing (crowd surveying) situation. With GCS, users can be linked to a server that has several surveys spread around in the real world, without being overwhelmed by the survey list, and not needing to worry about picking the ones that are relevant to their specific location beforehand. GCS' ability to place surveys on map areas would have allowed the New York Time's the state of your block [19] surveys (refer to section 2.3) to have been placed inside specific city blocks, and the ability to get geotagged answers would have allowed the surveyor to map people's observations and comments.

The same applies to surveys being relevant for certain users and not affecting others. For example, a geologic survey may require specific geology skills to answer. It doesn't make sense to make it available to every user. Some systems have no restrictions. For example, Open Data Kit will make every survey available to the users that participate in a given server. Others systems rely on manual survey URL distribution to each participant. GCS is the only that allows surveyors to simply publish their surveys, maybe for every participant, but, when appropriate, limit them to specific users, or perhaps more importantly, groups of users. Several different user groups can coexist in the same server without being a nuisance to each other.

Most systems are closed and cannot be changed by outside parties. Besides having flexibility/customizability as advantages, having a system be open to changes can get users closer to it, as they can try to improve it and implement the changes they want. Open Data Kit and LimeSurvey consist of open source code, and that has been a strength as they garnered user support to evolve over the years, and GCS can also stand to benefit from being released as open source.
5.3 Demonstration/APA Feedback

GCS was presented and demonstrated to the Coast and Coastal Protection Department from the Portuguese Environment Agency (APA) [70], which is a public institute within the scope of the Portuguese Ministry of the Environment, Territory Management and Energy. One of their endeavours is coastal monitoring, and they track erosion, flooding, and infrastructure damage along the Portuguese coastline over time. They need surveying with a geographic component, as location data is crucial to map the occurrences. Media files (pictures) are also necessary.

Having dedicated staff performing all the surveying would be too expensive, so the Coast and Coastal Protection Department resorts to getting help from other public servants, like the police or harbour staff. The marine police is one of their collaborators. When patrolling the coastal areas, on the water, they get a privileged perspective on the coast and can get perspectives APA researchers usually can't. This also constitutes a surveying system where the crowd is limited, because not everybody can participate. Participants receive some training in geology, in order to be able to contribute.

Their current system is using PDF forms. Participants get the forms via email. When they are out on the terrain, they take pictures with their smartphones. Later they can fill the PDF form, giving further information on what they witnessed, and post the form answer, over the Internet, when they have an Internet connection. The surveyors then need to handle these answers that are submitted.

Feedback on the functioning GCS (demonstration version) system was provided. The CSV answer export feature was deemed positive, as it is a standard format that's compact, and the possibility of selecting export dates is also desirable. The fact that there's currently only an Android application and no iOS App is not ideal, but the existence of a browser interface that can be used on iOS with a browser somewhat mitigates that. The possibility for social login was also appreciated as an alternative way to identify their participants. The menus (for the Web platform) where users can manage their surveys, survey answers, and look for surveys were appreciated for their directness and user friendliness.

The Coast and Coastal Protection Department at APA were positively surprised by the ability to restrict surveys to areas (and the Leaflet/Mapbox interface for area selection) and showing different surveys according to the user location, and with the ability for user and group restrictions.

With regard to survey presentation and form answering, the button that adds extra file upload fields was welcomed, as their researchers want to get as many terrain photographs as they can, because participants are not professional photographers, and as such the more pictures the researchers have to choose from, the better the quality of results that they can hope to achieve.

In terms of other functionalities they would like to see, an interest in conditional forms was revealed. There are some questions that should only be asked in case the participant checks some answers, corresponding to the different types of geological events that might happen, for different types of coastline. Also, some interest was manifested in having offline functionality for the App, because their participants are typically in remote areas of interest with poor or nonexistent Internet connections.
The system was also demonstrated to/used by 5 people aged between 22 and 28 years old. The mobile application's search for surveys, that pop-up as a notification when the user gets inside a survey's geographic area was compared to the Pokemon Go [71] mobile game, in the sense that the surveys become superimposed on reality, like the in-game places do. Offline functionalities for the Android application were the main suggestion for changes, as the participants seemed to want to save their mobile data plan usage as much as possible.

5.4 Mobile Data/Energy consumption/Application Installation

The application is not particularly demanding, either power-wise or data-wise. It is also relatively light as installation should require under 3 Megabytes of the device's storage space, for the demonstration version. Android devices have a pre-installed Play Store application from which they can get new applications. The application was deployed on the Google Play Store, for closed Alpha testing.

If the user turns the background service on, then the location services (likely using GPS, when away from known Wi-Fi networks) will be used. The service is not very demanding on the location services. It was set up to ask for updates only every 10 seconds, and the application receives updates if the location services have determined that there's been a displacement of over 50 meters from the last location. If there are other applications using location services, these updates might be faster. Furthermore, it stands to reason that using the service only makes sense when the user is moving around and interested in participating.

The background service energy demands were tested on a Samsung Galaxy tablet device (model SM-T113). Wi-Fi was on, to connect to a smartphone that was providing an Internet connection (but no location services at all). The tablet device's location services were turned on, the application's background service (automatic survey searching) was on, and testing was conducted on the streets. The device's screen was turned off for most of the time.

In three separate tests, with the duration of exactly 1 hour, the device's battery level suffered a 3% decrease (after each test). In the exact same circumstances, four tests were conducted using Google’s Google Maps [72] Android application, using its navigation mode (which gets location updates even when the screen is turned off). Two of the tests were conducted with the walking mode, and two using the driving mode. All showed exactly the same energy consumption, as after each 1 hour test the battery level had decreased by 3%. As such, GCS’s Android application background service (which makes use Google’s Play Services location functionalities, configured for relatively slow updates) shows the same energy consumption as the Google Maps Android application in navigation mode.

Under similar conditions (Wi-Fi on, location services on but not being used), but with the GCS application's automatic search off (background service turned off), after an hour the device either showed no difference in battery level or just a 1% decrease.

The application doesn't perform any (even moderately) demanding computational tasks, nor does it require any special graphical capabilities from the device. The most energy-demanding situation arises when manually updating the location, as quick, high precision updates, are demanded from the
location services. This is not very relevant energy-wise, as typically the user won’t need this for more than a few seconds at a time.

The application needs an active Internet connection at all times to function properly, being able to contact the remote Server. It is set up to use the device’s current connection, whether it is a Wi-Fi network or a cellular network.

When the background service receives a location, it makes a request (over the Internet) to the Server to ask about the presence of Area restricted surveys. The requests are very “lightweight”. Authentication (login/logout) operations are also very inexpensive data-wise. The survey list and survey presentation web pages are also very light, and the user shouldn’t need to use more than a few hundred Kilobytes from their data plan (in case they are using cellular network data), for these requests.

The most demanding data situation is uploading submissions containing photographs to the Server. Depending on the device camera’s specifications, a photograph might account for just a few hundred Kilobytes, but that size can get up to a few Megabytes. Typically, older devices have lower resolution/overall quality cameras that produce smaller image files, and newer, higher end devices have cameras that tend to produce bigger digital image files. In a worst case scenario, the user uses a few megabytes from their mobile data plan to upload each image they submit.
6 Conclusions and Future Work

6.1 Conclusions

This thesis presents an innovative solution for surveying with a geographic component. It introduces a new way to survey a crowd, where location plays a key part, and where surveys effectively become features on the map. The need for these types of systems by surveyors stems from the opportunities that the proliferation of Internet-connected smartphone and tablet devices present for surveyors. In the smartphone era, this solution can elevate the surveys' chances of success and the ability to get crowdsourced support, making the surveying process easier for everybody.

The crowd auditing/inquiry systems that are currently available, that are geared to reach big audiences, are well adjusted to consumer/crowd research but don't present significant geographic components. On the other hand, GIS enabled platforms present better suited solutions geographically-wise, but are tailored towards closed surveying with researcher groups as participants. Scientific studies and research aimed at these problems tend to have shortcomings, generally being very incomplete as they either leave the surveyor or the participants with burdensome, tedious tasks. The resulting system (GCS) results from a study of these solutions, their strenghts, and perhaps more importantly, their flaws and what they leave unexplored.

GCS relies on open source technology to achieve it's ends. This sort of solution is possible without the need for a big development team or company, because frameworks like Django allow the developer to use several pieces, several tools with pre-packed functionality, and plug them together with their own code to achieve a relatively powerful end product. The project provides a Django server project that enables a web platform for the surveyors and participants, where surveyors can manage their surveys' deployment and answers and participants are allowed to provide answers using a browser, and also a mobile Android application for participants to answer the surveys on the terrain.

A system evaluation was performed that showed that there is currently no dominant, "killer" solution that overshadow all others, and that this area is still open for new developments. It is shown that despite GCS not being that dominant solution, it achieves an interesting alternative, because of a unique functionality set. The way it integrates geo querying tools and spatially enabled data, combined with the survey restriction customization allowing surveyors to easily control who gets access and where, coupled with the assimilation of social authentication functionality, not only explores new possibilities, but makes it one usable, viable alternative system.

In summary, GCS provides a complete solution, from survey creation to answer collection, that in some aspects is showing some valuable advantages over already existing alternative systems, right from the outset. Open source server-side applications and tools, a modular server architecture, and the current abundance of open source smartphone libraries present the opportunity to keep evolving the system beyond this thesis, as will be reviewed in the following section.
6.2 Future Work

Overall Extensibility is one of the key strengths of this project. It is not inherently limited to current capabilities, and its architecture allows for extension options. The system was planned to be extensible and upgradeable in the future, without requiring major overhauls/significant architecture changes.

In terms of the future, one of the main possibilities is developing offline smartphone application functionalities. For demonstration purposes, the App’s automatic restricted survey searching (the background service) checks for surveys when there’s a displacement of 50 meters, asking for location updates every 10 seconds. There could also be an interface for the user to configure the automatic search settings, if there’s enough interest in using this feature.

A particular concern was enabling future use of different form building systems. It’s viable to go a different route and use other (different, or better altogether, according to the practical scenario) form building systems, whether Django applications or outside tools. For example, the form.io [73] form management/building platform seems promising as a potential alternative.

Answer handling is another possible area for extensions. For example, answer export can be extended to produce other different types of files that may be needed, along with integrations with GIS. Some interesting possibilities might arise, things like sending an email to a surveyor when an answer comes in, in situations where answers don’t come in too often. These types of needs can be added via programming/web development without the need for any revamping of already existing code.

It’s also worthy to note that developing an iOS App, similar in functionality to the Android application that was created, is also possible.
References


