

# iscte

INSTITUTO  
UNIVERSITÁRIO  
DE LISBOA

---

*Radars Exchange - Open market for goods and services at no cost*

Sandro Filipe Nascimento Isqueiro

Mestrado em Engenharia de Telecomunicações e Informática,

Orientador:

Doutor Luís Henrique Ramilo Mota, Professor Auxiliar,  
Iscte – Instituto Universitário de Lisboa

Novembro, 2021



Departamento de Ciências e Tecnologias da Informação

*Radar Exchange - Open market for goods and services at no cost*

Sandro Filipe Nascimento Isqueiro

Mestrado em Engenharia de Telecomunicações e Informática,

Orientador:

Doutor Luís Henrique Ramilo Mota, Professor Auxiliar,  
Iscte – Instituto Universitário de Lisboa

Novembro, 2021





*I dedicate this dissertation to all those who supported and helped me get here. Thank you  
very much!*



## **Acknowledgment**

First and foremost, I want to thank my parents for always providing all of the tools and resources available to me, as well as all of the patience, support, and incentives that made this academic course possible.

Thank you to all of my family and friends who have always been there for me in difficult times, and a special thank you to all of my university friends with whom I have shared my knowledge, memories, and support in all situations.

I would also want to thank the Iscte - Instituto Universitário de Lisboa for all of the classes that provided me with the knowledge that enabled the completion of this project.

Finally, a big thank you to the project's director, Prof. Doutor Lus Mota, for all of the lessons, availability, and understanding that enabled the completion and completion of this dissertation.

I'll be eternally grateful to everyone!



## Resumo

A sociedade atual é caracterizada pelo seu consumismo e rápido desenvolvimento, o que por sua vez leva ao surgimento de inúmeros bens não utilizados. Contudo, pessoas preocupadas com este desperdício têm aderido a uma economia colaborativa, onde partilham bens e serviços entre si e assim reduzem o desperdício.

Com a generalização do uso das redes sociais, o aparecimento de comunidades e serviços geridos de forma partilhada tem aumentado. Como tal, grupos como o Free Cycle e Community Exchange têm emergido para ajudar a procura e a troca dos produtos.

Seguindo o exemplo destes dois grupos e aproveitando a tecnologia disponível, esta dissertação documenta a criação de um Sistema de Informação Geográfica com o objetivo de disponibilizar uma plataforma web para a troca dos produtos e de serviços. Esta nova abordagem, onde a informação geográfica é o grande foco, permite aos utilizadores, durante a sua mobilidade quotidiana, a visualização dos produtos disponibilizados consoante a sua localização, aumentando assim a probabilidade de descobrir produtos perto de si. Para além disso, é implementada uma classificação sobre o utilizador com base nos produtos e serviços recebidos e doados, de forma a aumentar a sua credibilidade perante os restantes, elevando as chances de a troca se efetuar.

Ao longo deste documento, é descrito o processo tomado, isto é, a pesquisa levada a cabo para melhor compreensão do tema, levantamento dos requisitos, planeamento e criação de um protótipo.

Por fim, foram realizados testes de usabilidade sobre a solução criada, para avaliar o desempenho e a interface da solução criada.

**Palavras-Chave:** Sistema de Informação Geográfica, PostgreSQL, PostGIS, Webservice



## Abstract

The current society is distinguished by its consumerism and rapid development, which has resulted in the emergence of numerous unutilized items. People concerned about this urgency, on the other hand, have joined a collaborative economy in which goods and services are shared amongst themselves, reducing desperation.

With the widespread use of social networks, the emergence of communities and services managed in a distributed manner has grown. As a result, organizations such as Free Cycle and Community Exchange have emerged to aid in the search and exchange of goods.

Following the example of these two groups and using available technology, this dissertation documents the development of a Geographic Information System with the goal of providing a web-based platform for the exchange of goods and services. This new approach, in which geospatial information is a key focus, allows users to view available products based on their location, increasing the probability of discovering products nearby. Aside from that, a classification of the user is implemented based on the products and services requested and offered, in order to increase the user's credibility in the eyes of the others, increasing the likelihood of a deal occurring.

Throughout this work, the process is described, that is, the research carried out to gain a better understanding of the topic, the assessment of requirements, and the planning and creation of a prototype.

To conclude, usability tests on the created solution were conducted in order to evaluate its performance and interface.

**Keywords:** Geographic Information System, PostgreSQL, PostGIS, Webservice





## Contents

Acknowledgment	iii
Resumo	v
Abstract	vii
List of Figures	xiii
List of Tables	xv
Acronyms	xvii
Chapter 1. Introduction	1
1.1. Motivation	1
1.2. Context	1
1.3. Research Questions	2
1.4. Objectives	3
1.5. Investigation Method	3
1.6. Documentation structure	4
Chapter 2. Literature Review	5
2.1. Geographic Information Systems	5
2.1.1. Spatial Data	5
2.1.2. Spatial Database	6
2.2. Related Applications	8
2.2.1. Article 1 - <i>Web application using Google Maps API</i>	8
2.2.2. Article 2 - <i>Collaborative system for creating interactive and tourist routes using the Google Maps API</i>	8
2.2.3. Article 3 - <i>Animaps: A Geographic Information System for Animal Adoption</i>	8
2.3. Geographic Information Systems Repositories	9
2.3.1. <i>PostGIS</i>	9
2.3.2. <i>MySQL</i>	9
2.3.3. <i>CARTO</i>	10
2.3.4. Comparation	10
2.4. GIS Visualization	10
2.4.1. Leaflet	11
2.4.2. OpenLayers	11
	ix

2.4.3. Mapbox	12
Chapter 3. Design and Development	13
3.1. Resources	13
3.1.1. Development environment	13
3.1.2. Database	13
3.1.3. Visualization of Spatial Data	14
3.2. Application's Design	15
3.2.1. Functional Requirements	15
3.2.2. System Architecture	16
3.2.3. Wireframes	17
Chapter 4. Prototype	21
4.1. Database	21
4.1.1. PostGIS Extension	25
4.1.2. Database Connection	25
4.2. User Interface	25
4.3. Features	28
4.3.1. Account Creation	28
4.3.2. Login & Logout	30
4.3.3. Leaflet Map	30
4.3.4. Leaflet Plugins	31
4.3.5. Account Edition	33
4.3.6. User's Locations	34
4.3.7. Product Search	34
4.3.8. Product Request	35
4.3.9. Liking of products	36
4.3.10. User Subscription	37
4.3.11. Create and Edit Offers	38
4.3.12. Transactions Registry	39
4.4. Deployment	39
Chapter 5. System Evaluation	43
5.1. Usability test	43
5.2. Participant Profile	43
5.3. Results	44
Chapter 6. Conclusions	47
6.1. Overview	47
6.2. Research Answers	47
6.3. Acquired Skills	48
6.4. Challenges & Future Work	48
6.5. Final Appreciation	49

References	51
Appendix A. Usability Test	53
Appendix B. Feedback Quiz	55
Appendix C. Responsivity	57



## List of Figures

Figure 1	FreeCycle website	2
Figure 2	Community Exchange website	2
Figure 3	Design Science Research Methodology Process Model	3
Figure 4	Geographic map of 2020 EUA elections	5
Figure 5	GIS Generations (updated information from [1])	6
Figure 6	Different types of spatial data	7
Figure 7	Leaflet	11
Figure 8	OpenLayers	12
Figure 9	Mapbox	12
Figure 10	Browser Market Share Worldwide from Sept 2020 to Sept 2021 [2]	14
Figure 11	DB-Engines Ranking of Relational DBMS [3]	14
Figure 12	DB-Engines Ranking of Spatial DBMS [4]	14
Figure 13	Use-Case Diagram	15
Figure 14	Class Diagram	16
Figure 15	Component Diagram	17
Figure 16	Website's Layout Design	19
Figure 17	Website's Layout Design	20
Figure 18	Database Schema	22
Figure 19	Database Schema - users Table	23
Figure 20	Database Schema - confirmation_code Table	23
Figure 21	Database Schema - subscription Table	23
Figure 22	Database Schema - products Table	24
Figure 23	Database Schema - products_location Table	24
Figure 24	Database Schema - locations Table	24
Figure 25	Database Schema - trades Table	24
Figure 26	Database Schema - likes Table	24
Figure 27	Database Schema - messages Table	24
Figure 28	Website's Interface - Homepage	26

Figure 29 Website's Interface - Access process	26
Figure 30 Website's Interface - Search Header	27
Figure 31 Website's Interface - MyStore	28
Figure 32 Website's Interface - Message	28
Figure 33 Website's Interface - Profile	29
Figure 34 Website's Interface - Error page	29
Figure 35 Spiderfy Cluster Marker	32
Figure 36 Cluster and Edge Marker	32
Figure 37 ESRI Geocoding	32
Figure 38 Product Request bottom	36
Figure 39 Request Message	36
Figure 40 Like Button Status	37
Figure 41 Like Button Status	38
Figure 42 App Service	40
Figure 43 Azure Database for PostgreSQL server	40
Figure 44 SendGrid Service	41
Figure 45 Participants Age	44
Figure 46 Participants Gender	44
Figure 47 Feedback	46
Figure 48 Feedback Quiz	55
Figure 49 Feedback Quiz	56
Figure 50 Website's Interface for smarthphone	57
Figure 51 Website's Interface for smarthphone	58

## List of Tables

Figure 1	Number of functions available	9
Figure 2	Available Functions - MySQL v8.0	10
Figure 3	Number of functions available	11





## Acronyms

**AJAX:** Asynchronous JavaScript And XML

**API:** Application Programming Interface

**ArcGIS:** Aeronautical Reconnaissance Coverage Geographic Information System

**BLOB:** Binary Large Object

**CSS:** Cascading Style Sheets

**DB:** Database

**DSRM:** Design Science Research Methodology

**ESRI:** Environmental Systems Research Institute

**GIS:** Geographical Information System

**GUI:** Graphical User Interface

**HTML:** HyperText Markup Language

**IS:** Information System

**JSON:** JavaScript Object Notation

**NoSQL:** Non Structured Query Language

**PHP:** Hypertext Preprocessor

**RDBMS:** Relational Database Management System

**SQL:** Structured Query Language

**SRID:** Spatial Reference Identifier

**SSH:** Secure Shell

**UI:** User Interface

**UML:** Unified Modeling Language

**UX:** User Experience



## CHAPTER 1

### Introduction

#### 1.1. Motivation

One of the main problems that widely affects today's society is the excess of unused goods. This dilemma is relatively expectable since our society can be characterized by its consumerism and constant evolution, which can rapidly lead to a different appreciation of the object over time and its consequent obsolescence.

When this obvious consumerism is combined with population growth, it results in a never-ending cycle of underutilized wasted products. Although there are solutions such as OLX, eBay, Facebook Marketplace, and others that allow people to resell their goods, monetizing the products sometimes leads to underestimating/overvaluing them and the piece remaining unsold, resulting in the products being discarded.

To at least assist the community in reducing the impact of this situation, it may be beneficial to create a means of giving away products rather than selling. To that end, the purpose of this dissertation will be to design a geographic information system whose primary role will be the free exchange of goods and services.

#### 1.2. Context

The widespread use of social networks has resulted in the emergence of solutions, such as auction sites and second-hand markets, among others, which open up a range of new possibilities for mitigating the previously mentioned problem. However, as previously stated, product monetization introduces a new issue, as product evaluation varies from person to person.

In order to overcome these issues, as well as cope with economic difficulties, people all over the world have chosen to participate in a Sharing Economy [5]. Essentially, these individuals chose to participate in a socioeconomic system that differs from the traditional business model of corporations, in which assets and services are shared among individuals. Its essence is dependent on users' willingness to share their goods and communicate/trade with others. These are the two opposing forces that prevent the Sharing Economy from spreading even further.

In an effort to assist these goals, groups such as FreeCycle, [6], and Community Exchange, [7], emerged, and according to this last one, monetization "is at the root of most of the misery, suffering, and problems faced by humanity", [8]. So these two groups focus only on the reuse of products and created a simple website for sharing unwanted products, and services provided by one person that others may be interested in.



FIGURE 1. FreeCycle website



FIGURE 2. Community Exchange website

In an era in which digital information is becoming increasingly important, it is prudent to have a system that can process data in a meaningful and useful manner. According to Laudon [9], an information system can be defined as a set of interrelated components working together to gather, retrieve, process, store and distribute information in order to facilitate control, coordination, analysis, and decision-making in organizations. Aiming at a more reliable and less bureaucratic flow of information, the Information Systems can provide better stability, security in accessing information, integrity, and reliability of the information, and more.

With the same focus of FreeCycle and Community Exchange in mind, this dissertation will focus on developing a more versatile solution, with a Geographical Information System where the main innovation will be the geolocation of products and services, in order to better explore the interaction between users and their location, and increase the likelihood of a product/service being recycled. This system will be used to manage the transactions of users' goods and services, without any associated cost, with a web interface and a system for registering and maintaining the participants' profiles.

### 1.3. Research Questions

In carrying out this dissertation, where the main objective is the creation of a geographic information system with a web interface, it is intended to analyze how much it can contribute to the problem presented through the following questions:

- (1) What is the impact of a system based on geographic information has on the user interaction and perspective?
- (2) Is it relevant to consider the reputation of users to improve the perception of the usefulness and suitability of the system?

On the other hand, there are questions that need to be answered in order of a better understanding of the matter, related to the main components of these types of systems - the geolocation. As such, for personal knowledge this dissertation also intends to answer questions such as:

- (1) What is a Geographic Information System and what kind of data does it work with?
- (2) How to store, manipulate and manage geographic information in a distributed application?

## 1.4. Objectives

The main goals of this dissertation are:

- (1) Study the basics of geographic information and some of the existing repositories for this kind of data;
- (2) Definition of desired requirements and data structure for IS support (UML);
- (3) Creation of a geographic information system and a web interface to access and manage such information;
- (4) Evaluate the performance of the system.

## 1.5. Investigation Method

In this type of work, one cannot rush to programming without planning. It's necessary planning and idealization of the tasks be carried out from the beginning to the end.

As such, in order to maximize the functional performance of the system, its development will follow the Design Science Research Method. As shown in [10], this method has the main objective to offer better knowledge and understanding of the problem in question, in order to achieve better results. For this, the DSRM defines a set of procedures to be followed, as seen in Figure 3, in order to solve the problems.

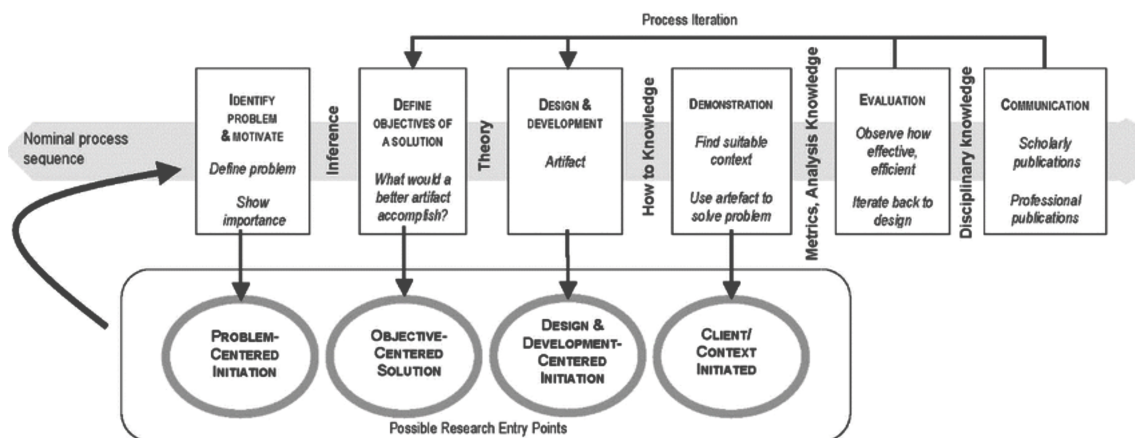


FIGURE 3. Design Science Research Methodology Process Model

Given the nature of this dissertation, where the research is based on the creation of an Geographic Information System, it is advisable to take a problem centered approach. With this view, the starting point will be at the very beginning of this method, with the identification of the problem and motivation, and then the definition of the objectives, which can be seen in the previous sub-chapters. After that begins the design and development of the solution, where it will be defined and constructed the architecture that the system will have as also his interface. When the solution is finished, the system will be evaluated through tests with users, thus obtaining their feedback and locating possible

flaws. Finally, the presentation of this dissertation will work as a way to submit and communicate our solution.

## **1.6. Documentation structure**

This dissertation is divided into six chapters that describe the main processes involved in developing the solution.

Starting with the introduction, the current chapter presents the main theme of the dissertation, as well as the objectives and method used.

The second chapter, Literature Review, describes the research that was conducted to better understand the main topic of this work, as well the existing solutions considered as examples and the technologies that could be used in the creation of the solution.

In the third chapter, Design and Development, is based on the planning and design for the solution are revealed.

The prototype chapter describes the created solution, that is, the elements that make it up, its primary functions, and the requirements for its deployment.

The following chapter, System Evaluation, describes the usability test performed on the solution, as well as the feedback received as a result of the same.

In the final chapter, a summary of the limitations and conclusions drawn from this dissertation is presented.

## CHAPTER 2

### Literature Review

In this chapter, the research that was carried out is presented, by topic, with the purpose of broadening the knowledge about Geographic Information Systems and the resources that should be used in their creation.

#### 2.1. Geographic Information Systems

Put simply, Geographic Information Systems are systems that allow the storage, analysis, manipulation, and visualization of geographic information, [1]. This type of system is used in computational cartography to create maps, and spatial and statistical methods can then be used to analyze attributes and geographic information. A well-known basic example of this type of analysis is the geographic maps created in the American presidential elections, where the counted votes and the percentage are presented, as well as which candidate is ahead in each state, Figure 4<sup>1</sup>.

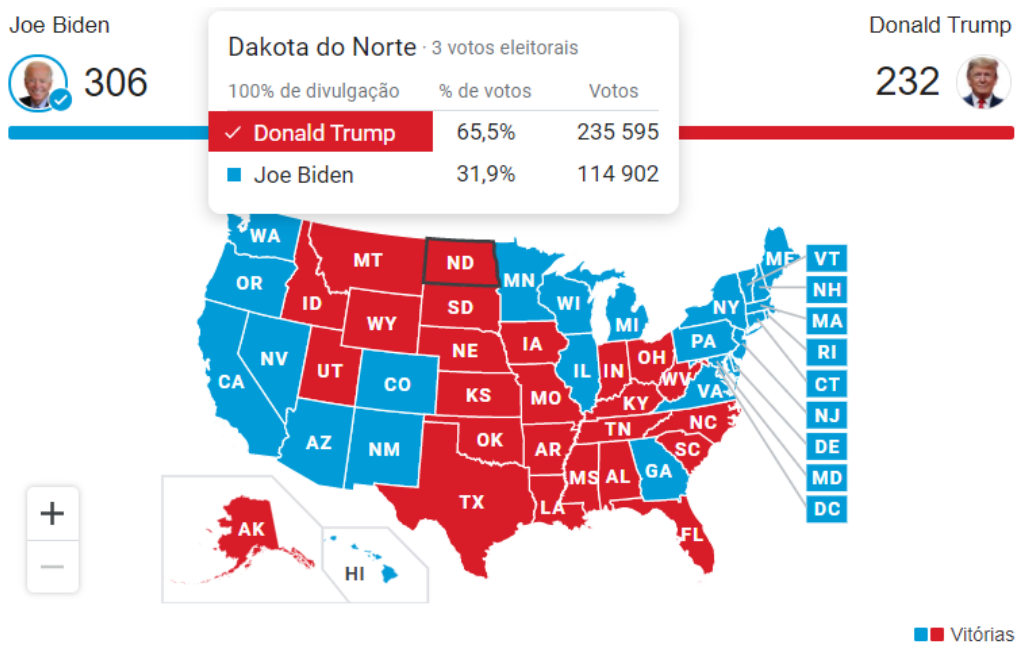


FIGURE 4. Geographic Map of 2020 EUA elections

But what is Geographic Information and how is it stored?

##### 2.1.1. Spatial Data

In this type of system, the keyword is Geography and it can be deduced that spatial information, also known as geographic information, is a type of information of an object's

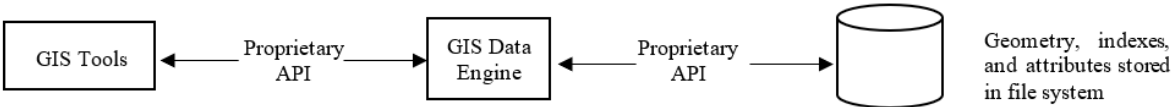
<sup>1</sup>Fonte: The Associated Press

location relative to the Earth’s surface. More specifically, it results from the combination of location data and other attributes/characteristics of the object, [1].

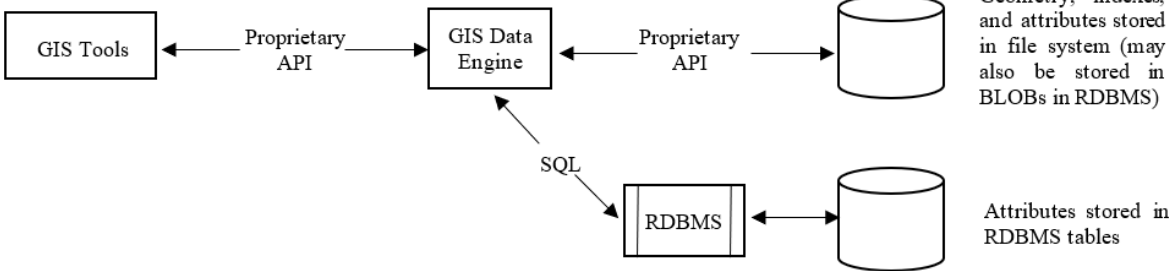
**2.1.2. Spatial Database**

As one of the most useful technology that had emerged, geographic information systems (GIS) are constantly evolving for over fifty years. The architecture of these types of systems, and the way they interact with spatial information, have undergone several transformations over time. Members of the geographic information community have seen this technology advance and it is possible to separate its progress in 4 generations, in Figure 5 (updated information from [1]):

First-Generation GIS:



Second-Generation GIS:



Third-Generation GIS:



Fourth-Generation GIS:

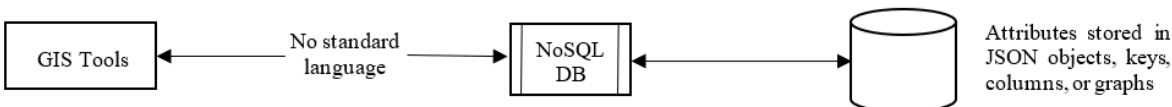


FIGURE 5. GIS Generations (updated information from [1])

- 1st generation - where spatial data was stored in flat files. This way there was no relational structure but rather a positional one, and GIS software was needed to interpret and manipulate the data;
- 2nd generation - separation of spatial data into spatial and non-spatial parts, where non-spatial attributes were stored in a relational database, separate from the rest;



- 3rd generation - creation of spatial databases, where spatial data started to be recognized as a relational object;
- 4rd generation - use of NoSQL databases after the realization that relational databases present some limitations in flexibility, scalability, slow queries and others. Although limited some NoSQL databases support spatial data.

These changes can be explained as the result of a simple change of perspective: it has shifted from an orientation centered on the GIS to an orientation centered on the database. This breakthrough idea has allowed to leave behind a workstation-based software to shift to data and tools hosted in the cloud or in servers.

A spatial database then allows the user to store and manipulate spatial data, like any other object. In addition, the ability to differentiate the types of spatial data, that are organized in a hierarchy, was added in order to be able to represent various geographical features, with different shapes. As it can be seen in Figure 6, these are the set of geometry types proposed by Open GIS Consortium, a worldwide community committed that defines standards for spatial data.

## Geometry Hierarchy

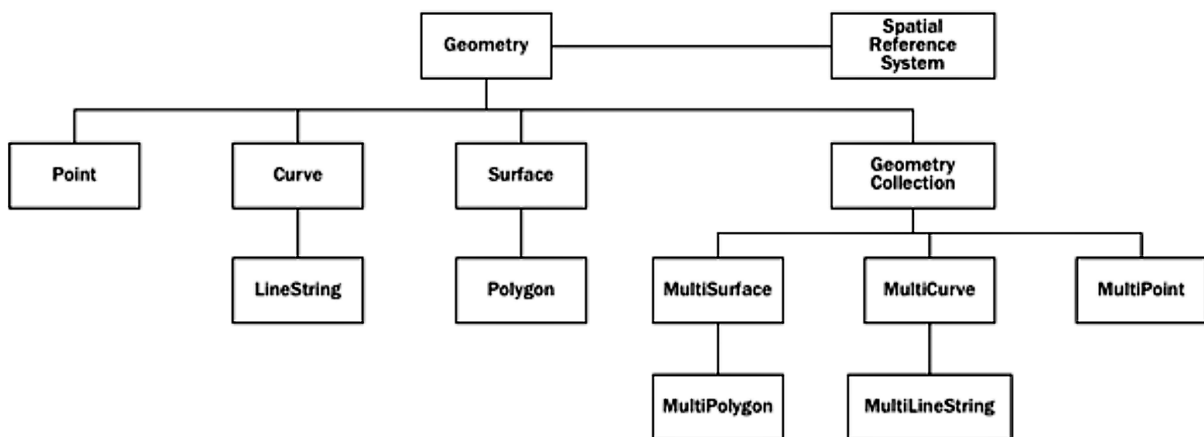


FIGURE 6. Different types of spatial data

When it comes to data manipulation, a spatial database provides the user a large number of functions that allow analyzing geometric components, determining spatial relationships and manipulating geometries. These functions can be divided into the following 5 categories:

- Conversion - conversion between geometries and external data formats;
- Management - manage the spatial information located in the tables;
- Consultation - consult the properties and measures of a geometric object;
- Comparison - comparison between two geometric shapes;
- Creation - creation of new geometric shapes.

## **2.2. Related Applications**

In this section, it'll be briefly analyzed some researchers's works and the resources they used, which will serve as a reference for this work. This search is mainly focused on the methods used to store and display geographic information.

### **2.2.1. Article 1 - *Web application using Google Maps API***

In [11], in order to conduct a study on geoprocessing using an API, thus demonstrating its usefulness and main features, this work provided a web interface, produced with technologies such as *PHP*, *HTML*, *Javascript*, *JQuery*, *Ajax* and *Json*, to the user, and an interactive map was provided. This map offered by the Google Maps API, displayed several signposted real estate locations (with description), which were entered by users through a form. All geographic information was stored in the *PostgreSQL* database, which, using the *PostGIS* extension, is able to store, manage and process geographic information. To test this application, a local server was used through WampServer.

### **2.2.2. Article 2 - *Collaborative system for creating interactive and tourist routes using the Google Maps API***

In [12], the idea of this work was to create a web system that integrates GPS technology with APIs for the management of tourist routes in a collaborative way between users. The system made it possible to create tourist routes through the points marked by the user on the map, thus creating a route to be followed and details regarding those same points using web content. For its realization, technologies such as *HTML*, *Javascript*, *JQuery*, *Ajax*, *Json*, and framework called "CodeIgniter" was used for the part related to *PHP*. For the display of the map, the Google Maps api was once again used, which also allows the capture of the user's approximate location (if the user allows). To store information about the routes created by users, *MySQL* was used, which has spatial extensions that allow the creation, storage and analysis of geographic data.

### **2.2.3. Article 3 - *Animaps: A Geographic Information System for Animal Adoption***

In [13], work dedicated to animals, creating a website in order to help their adoption and donation. The implementation of this project allowed users to create an account and to provide information about animals, either for adoption and donation or lost and found, or for advertising events and fairs about them. In addition, the use of geolocation was used, where the information entered by users was displayed on a map with a visualization option. The creation of this page used *HTML5*, *CSS3*, *Javascript* and *PHP* and the *MySQL* database. The Google Maps API was also used to incorporate the map and the Geocoding API to convert the addresses entered by users into coordinates (latitude and longitude), which, supposedly, were stored in the mysql database.

As it can be seen, there is a recurring use of the Google Maps API in creating and displaying maps on web pages. However, with constant updates to the privacy terms and

other issues, this API may not be desirable. As such, it is recommended to look for other viable solutions.

### 2.3. Geographic Information Systems Repositories

With some solutions listed in the previous section, it is essential to carry out a more detailed analysis of these methods used, and others that are free for use which it may come to use.

#### 2.3.1. *PostGIS*

As a free and open-source extender for the object-relational database PostgreSQL, PostGIS increases its management capabilities by adding geospatial types (Figure 6) and functions to enhance spatial data handled within a relational database structure. Regarding the available functions (Table 1, provided by [14]) for spatial data, a common set of functions defined by the Open GIS Consortium are implemented together with additional useful functions making PostGIS one of the software that offers more functions. With a language similar to SQL allows performing spatial analysis and typical queries on spatial data making it easy to manage and use compared to other GIS software, according to [15].

However, this solution does not allow the visualization of spatial data and, as such, it will be necessary to combine another method such as the Google Maps API to represent this data.

TABLE 1. Number of functions available

Feature	PostGIS
Supported Geometry Types	Point, Polygon, LineString, MultiPoint, MultiPolygon, MultiLineString, GeometryCollection, CircularString, CompoundCurve, CurvePolygon, MultiCurve, MultiSurface
Constructors	14
Accessors and Editors	76
Spatial Reference System	3
Input functions	31
Output functions	17
Spatial Relationships	21
Measurement	27
Overlay	10
Processing	25
Other types	8

#### 2.3.2. *MySQL*

To add/remove, access, and process the information in a structured collection of data, a database management system such as MySQL it's needed. With relational databases, MySQL is one of the most known Open Source software that offers a very fast, reliable,

scalable, and easy to use solution. MySQL implements spatial extensions following the specification of the Open GIS Consortium and implements a subset of the SQL with geometry types environment proposed by them (Figure 6) to enable the generation, storage, and analysis of geographic features, according to [16].

Such as PostGIS, this solution does not allow the visualization of spatial data and, as such, it will be necessary to combine another tool for such purpose.

TABLE 2. Available Functions - MySQL v8.0

Feature	MySQL
Supported Geometry Types	Point, Polygon, LineString, MultiPoint, MultiPolygon, MultiLineString, GeometryCollection
Constructors	10
Accessors and Editors	24
Spatial Reference System	2
Input functions	32
Output functions	6
Spatial Relationships	17
Measurement	6
Overlay	4
Processing	7
Other types	1

### 2.3.3. *CARTO*

As a solution to the problem of data visualization presented above, presented by [17], CARTO is an open, powerful, and intuitive platform. It is a cloud computing platform that provides a geographic information system, web mapping, and spatial data science tools. With its database created using PostGIS, CARTODB supports advanced PostGIS features, allowing basic PostgreSQL expressions to be used and with the help of Deck.gl, for the creation of maps, the user can generate maps with different types viewing.

### 2.3.4. Comparison

PostGIS and MySQL, these are the two spatial data repositories that are presented above and in order to better understand the characteristics of each one, Table 3 was created, to compare the available functions in each repository.

As it can be seen from the Table 3 and analyzed from [14] and [16], PostGIS provides a greater amount and variety of data types and spatial functions, making this a more desirable solution to our problem leaving only the choice of data representation. To solve this, CARTO appears, basing the database on PostgreSQL, providing its own maps for data visualization

## 2.4. GIS Visualization

With a solution that operates on the physical locations of users and available products, a way of visualizing those same data that is interactive and dynamic for users is required.

TABLE 3. Number of functions available

Feature	MySQL	PostGIS and CARTO
Supported Geometry Types	Point, Polygon, LineString, MultiPoint, MultiPolygon, MultiLineString, GeometryCollection	Point, Polygon, LineString, MultiPoint, MultiPolygon, MultiLineString, GeometryCollection, <b>CircularString</b> , <b>CompoundCurve</b> , <b>CurvePolygon</b> , <b>MultiCurve</b> , <b>MultiSurface</b>
Constructors	10	<b>14</b>
Accessors and Editors	24	<b>76</b>
Spatial Reference System	2	<b>3</b>
Input functions	<b>32</b>	31
Output functions	6	<b>17</b>
Spatial Relationships	17	<b>21</b>
Measurement	6	<b>27</b>
Overlay	4	<b>10</b>
Processing	7	<b>25</b>
Other types	1	<b>8</b>

This section contains a collection of Javascript libraries for maps which could be used in the creation of a solution.

#### 2.4.1. Leaflet

Leaflet, [18], is the most widely used open-source JavaScript library for creating mobile-friendly interactive maps. This is a free and open-source JavaScript library for creating interactive maps. A small but functional library with an easy-to-use API that works on all browsers and platforms. It also has a plethora of plugins available, which means that many functionalities can be added. It is also supported by some of the world's leading companies, including GitHub, Flickr, Facebook, Etsy, and many others.



FIGURE 7. Leaflet

#### 2.4.2. OpenLayers

OpenLayers,[19], is a feature-rich, modular, high-performance library for displaying and interacting with maps and geospatial data that supports a wide range of commercial and free image and vector tile sources, as well as the most popular open and proprietary vector

data formats. It offers an API for developing rich web-based geographic applications, similar to Google Maps and Bing Maps.



FIGURE 8. OpenLayers

### 2.4.3. Mapbox

Mapbox, [20], is a location data platform that powers many popular apps' maps and location services. Mapbox is an embed dynamic, interactive, customizable map library for webpages and mobile devices that combines vector tiles and 3D rendering technology, making it one of the most advanced mapping solutions on the web right now.



FIGURE 9. Mapbox

## CHAPTER 3

### Design and Development

This chapter describes a variety of resources and tools that aided in the quick and easy development of the website, as well as strategy for its design and construction.

#### 3.1. Resources

For the development of this application, a literature review was conducted with the goal of expanding knowledge not only about the major subject of this dissertation, but also about the resources available for its appropriate development. Based on this research, the resources to be used in the website's key technical components were chosen.

##### 3.1.1. Development environment

Two essential development tools were required for website development: Visual Studio Code and WampServer [21].

Visual Studio Code is a lightweight but powerful source code editor that combines the simplicity of a source code editor with advanced developer tooling, and it is one of the most widely used code editors. It was possible to use it to develop the *PHP*, *CSS*, and *Javascript* files that support our solution.

WampServer is a collection of open source web development tools that allow to run a local Apache server and test web or *PHP* applications. With these two components, the website under creation, as well as all of its functionalities, could be deployed locally and viewed in a browser.

The browser used in these testing was Google Chrome, which is widely used by people all around the world according Statcounter's study, as shown in Figure 10.

##### 3.1.2. Database

In terms of data storage, PostgreSQL version 11 was chosen to store and manage the website's data. This decision was made based on the considerable number of features available to help developers building applications, which have allowed this DBMS to be acknowledged as one of the preferred ones (Figure 11). Furthermore, one of its extension, PostGIS, offers support for geographic information and several functions that put it high on the geographical database rankings, as shown in Figure 12.

The PostgreSQL administration software, pgAdmin, was used to access, create, and alter the database tables. This tool is an open source administration and development platform for PostgreSQL that has a web-based GUI that enables for quick and straightforward handling over a secure connection (ssh) to local and remote servers.

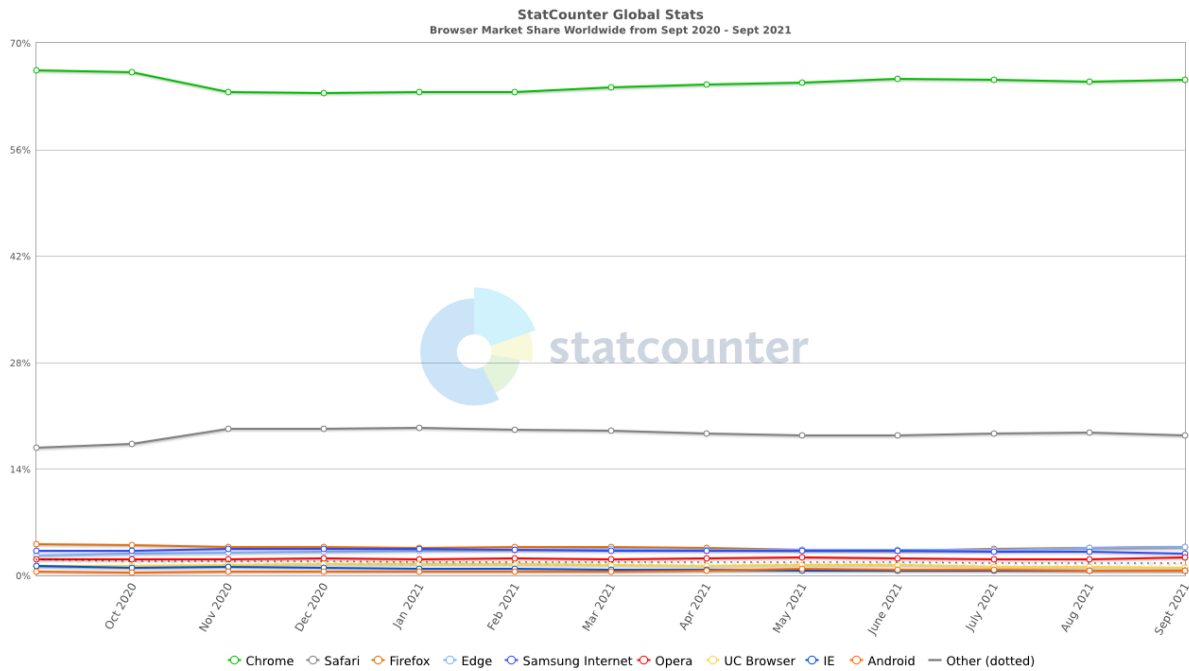


FIGURE 10. Browser Market Share Worldwide from Sept 2020 to Sept 2021 [2]

Rank			DBMS	Database Model	Score		
Oct 2021	Sep 2021	Oct 2020			Oct 2021	Sep 2021	Oct 2020
1.	1.	1.	Oracle +	Relational, Multi-model	1270.35	-1.19	-98.42
2.	2.	2.	MySQL +	Relational, Multi-model	1219.77	+7.24	-36.61
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model	970.61	-0.24	-72.51
4.	4.	4.	PostgreSQL +	Relational, Multi-model	586.97	+9.47	+44.57
5.	5.	5.	MongoDB +	Document, Multi-model	493.55	-2.95	+45.53
6.	6.	↑ 8.	Redis +	Key-value, Multi-model	171.35	-0.59	+18.07
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	165.96	-0.60	+4.06
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	158.25	-1.98	+4.41
9.	9.	9.	SQLite +	Relational	129.37	+0.72	+3.95
10.	10.	10.	Cassandra +	Wide column	119.28	+0.29	+0.18

FIGURE 11. DB-Engines Ranking of Relational DBMS [3]

Rank		DBMS	Database Model	Score	
Oct 2021	Sep 2021			Oct 2021	Sep 2021
1.	1.	PostGIS	Spatial DBMS, Multi-model	31.64	+0.93
2.	2.	SpatiaLite	Spatial DBMS, Multi-model	1.96	+0.05
3.	3.	GeoMesa	Spatial DBMS	0.76	+0.05
4.	4.	H2GIS	Spatial DBMS, Multi-model	0.17	+0.00
5.	5.	SpaceTime	Spatial DBMS, Multi-model	0.00	±0.00

FIGURE 12. DB-Engines Ranking of Spatial DBMS [4]

### 3.1.3. Visualization of Spatial Data

Leaflet was chosen for client-side geographic data visualization. Leaflet has a robust documentation project behind it, making it an excellent choice for novices, as there are many



community-contributed examples available on the Internet, as well as many examples on the project homepage.

### 3.2. Application's Design

Apps and websites, like any other product or service, must be planned and well structured. As a result, before beginning the website's creation, an analysis and design of the website's basic concept was performed. This process enabled a deeper grasp of the functional requirements and system architecture.

#### 3.2.1. Functional Requirements

The development of a website follows various development stages, both technical and strategic.

Functional requirements are the set of functionalities that make up the website. The use-case diagram in Figure 13 was created to identify the main functions of the solution. Thus, a user must be able to:

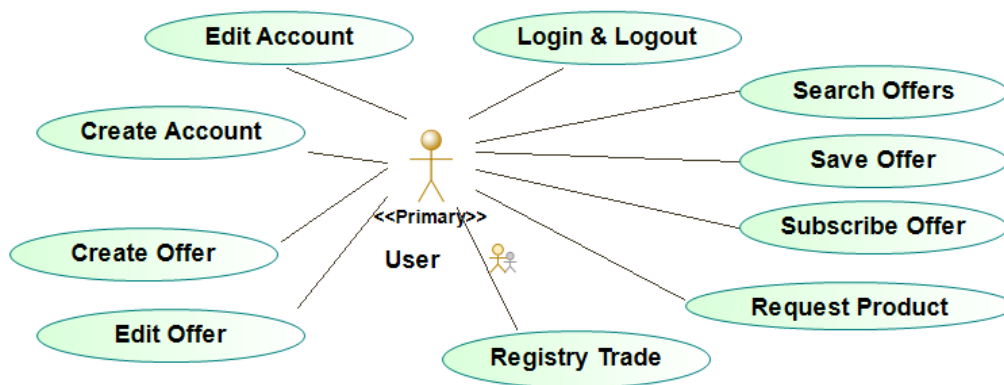


FIGURE 13. Use-Case Diagram

- Create Account:** The user must have the ability to establish a new account;
- Edit Account:** The user must have the ability to edit and delete his account;
- Login & Logout:** The user must be able to access the website and log in and out;
- Create Offer:** The user must be able to add a product to the market, which can be either an offer or a request;
- Edit Offer:** The user must be allowed to edit and delete the product after it has been created;
- Search Offer:** The user must be able to search for products that are available for offer or that are being requested;
- Save Offer:** The user must be able to save the offers that he finds appealing;
- Subscribe User:** Users must be able to subscribe to another user;
- Request Offer:** The user must be able to request the product offered;
- Register Trade:** An exchange must be able to be registered by the user.

These requirements define the fundamental elements of the solution and will serve as the foundation for its development.

### 3.2.2. System Architecture

Architectural representations enable the interpretation of a wide range of perspectives, with a focus on specific characteristics. Some viewpoints were investigated in order to better understand the problem at hand, and are thus represented below.

Figure 14, the Class Diagram [22], provides better view of the logic of our solution. A user, acting as an offerer, has one or more products that he or she desires to give away, which might be items or services, and then creates an advertisement describing that product and selecting one or more delivery locations. Another user, on the other hand, acts as a searcher of goods and services. The user's rating rises with each exchange, whether giving or receiving an item or service.

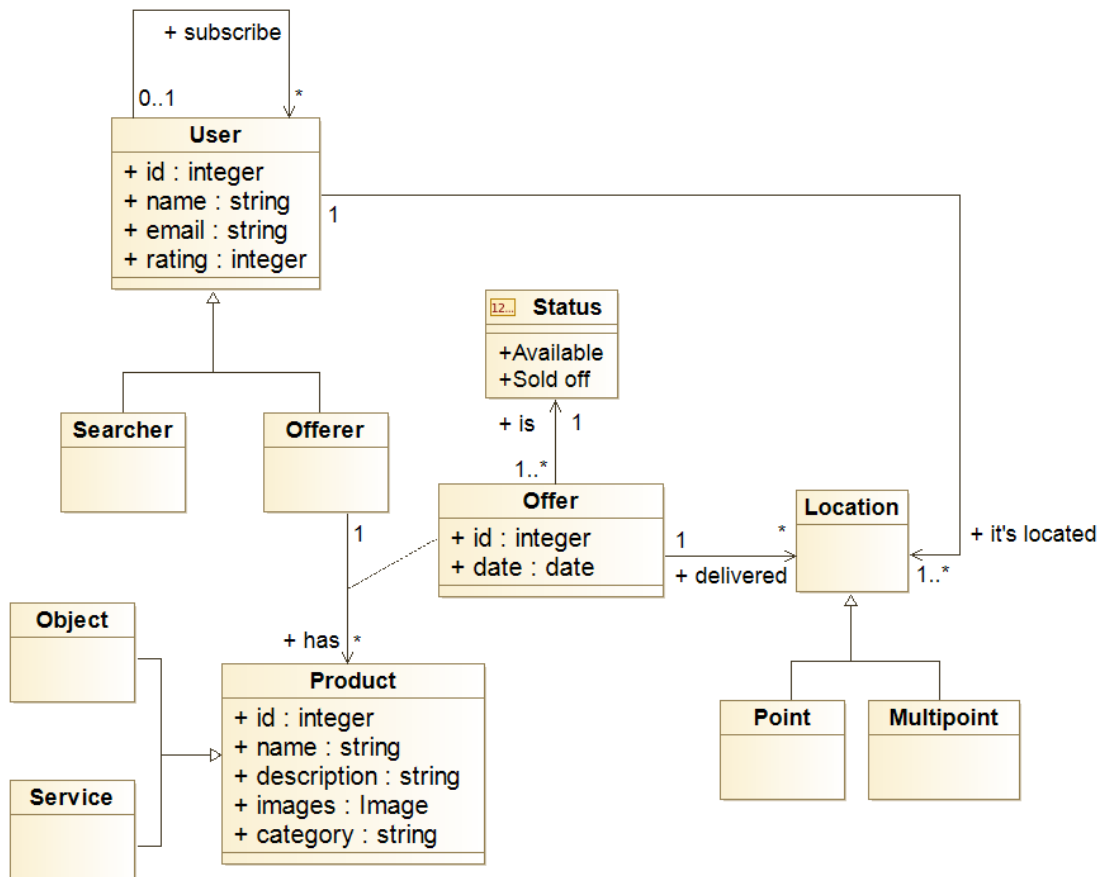


FIGURE 14. Class Diagram

The Component Diagram [23], Figure 15, allows us to have a better understanding of the link between the solution's primary components. As this solution relies on Geographic Information, a Spatial Database must be included among the main components. Furthermore, the system is built on a client-server architecture, with the server providing essential functionality to any client that connects to it.

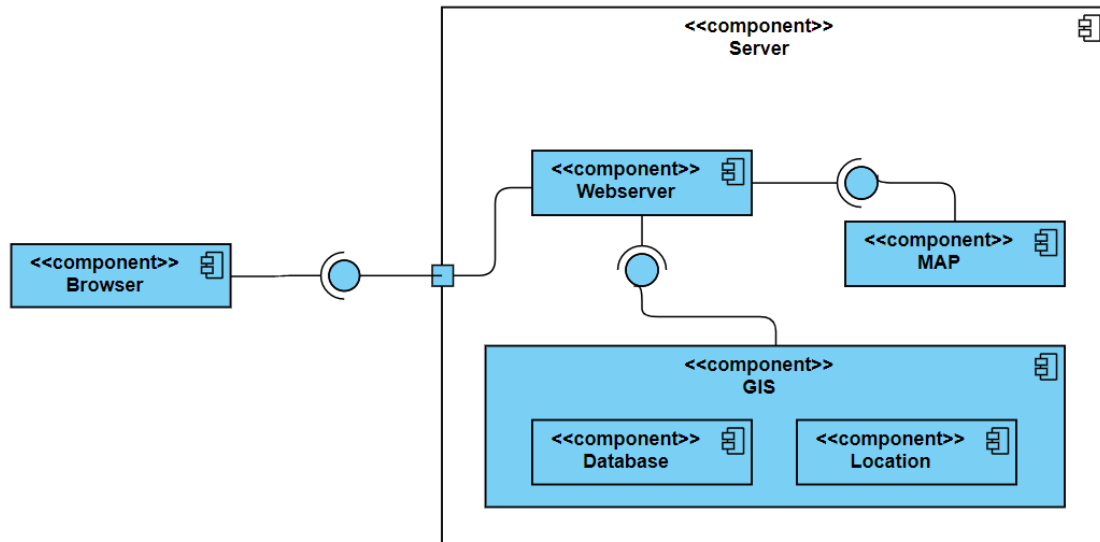


FIGURE 15. Component Diagram

### 3.2.3. Wireframes

The final phase in the Design process is to create a user interface for the solution. This concept refers to the product’s basic planning based on the preceding phases, beginning with basic concepts for its implementation. As these are not technical in nature, it is easy to receive feedback and make necessary adjustments rapidly from visual ”drafts,” which eases and guide the implementation of the prototype. Adobe XD [24], a rapid and advanced UI/UX design tool, was used to create the Layouts below (Figure 16 and 17).

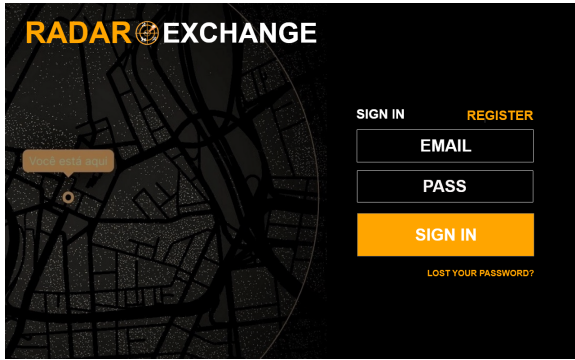
Figure 16a illustrates the website’s initial page, where the user can register and login, as well as find some information about the website. After successfully logging in, the user will be directed to the platform’s *Search* header. This header is made up of two pages: *Ads* and *Members*. The first one, allows users to search for products and services using a map (Figure 16b) or a list (Figure 16c), with the option to filter the results using the filter on the left. In both alternatives, products are represented by cards that, when clicked, will display a window (Figure 16d) with product’ and service’ information. The *Members* page, figure 16e, allows the user to perform a search on the platform’s registered users and their products.

Regarding the header *MyStore* (Figure 16f), as the name hints, it represents the user’s store, and, like the previous header, it is an aggregation of two pages. The *Inventory*, where you can add, edit, and remove own products and services. As in the previous header, products and services are represented by cards that, when clicked, open a form for product addition/edition (Figure 16g). There is also the *Registry* page, Figure 16h, where the users can register and confirm their transactions using the form shown in Figure 17a.

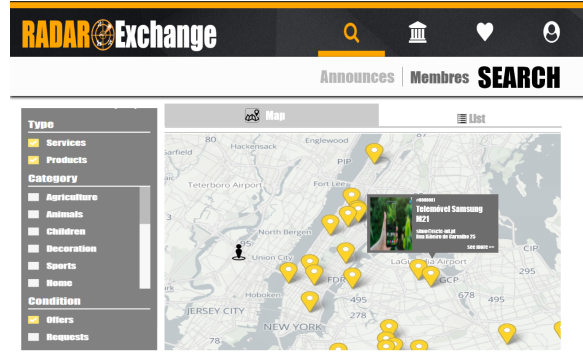
In the Header *Favorites* (Figure 17b), the user is presented with his favorite products and services, as well as his subscriptions, which are represented by cards which when clicked, display a window with the user’s complete information would be displayed.

Finally, there is the *Profile* header (Figure 17c), where the user can find his personal information. This header is an aggregation of two pages: *Profile* and *Logout*. Starting with the last, by clicking on the option, the user will log out of the platform. The *Profile* page will contain the user's personal information, and the user will be able to navigate through the various menu options and edit his general data, transaction location (Figure 17d), avatar (Figure 17e), and store statistics (Figure 17f).

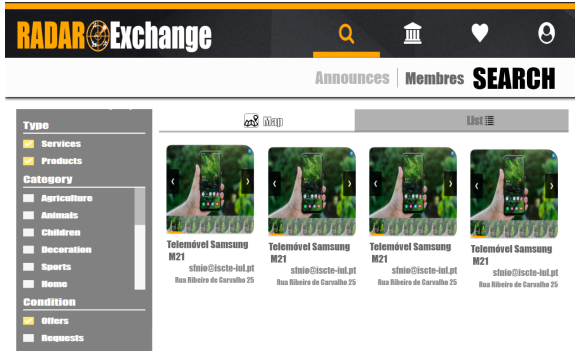
As previously stated, these wireframes are not the final version of the user interface, but rather an indication of how information and functionalities will be presented to the user, with changes made along the way due to feedback.



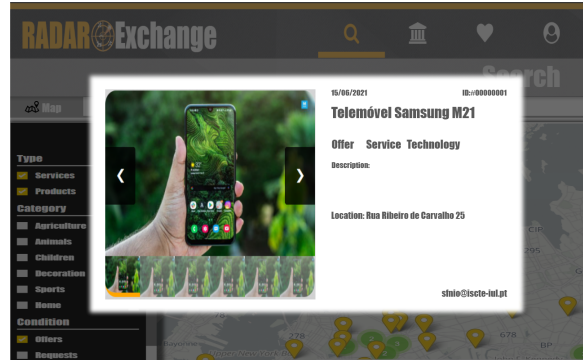
(A) Home Page



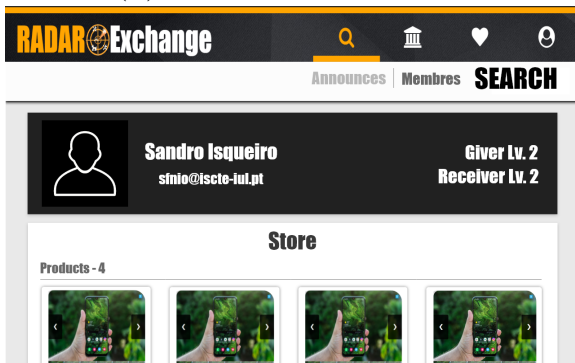
(B) Search Page - map mode



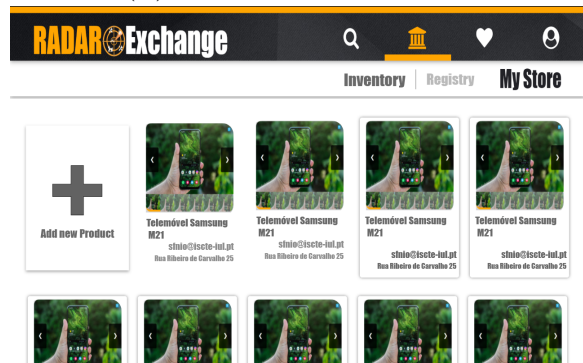
(C) Search Page - list mode



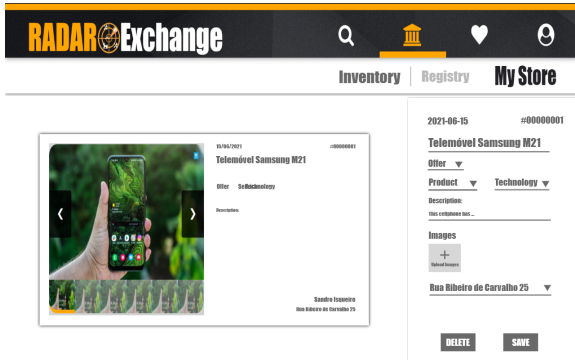
(D) Search Page - ads view



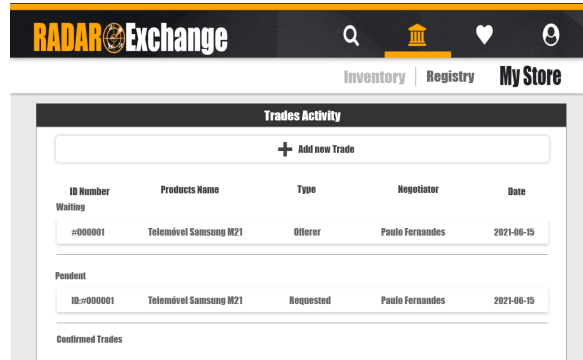
(E) Search Page - member view



(F) MyStore Page - Inventory list

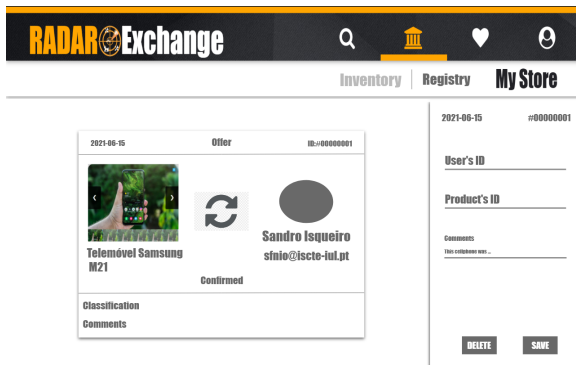


(G) MyStore Page - Inventory form

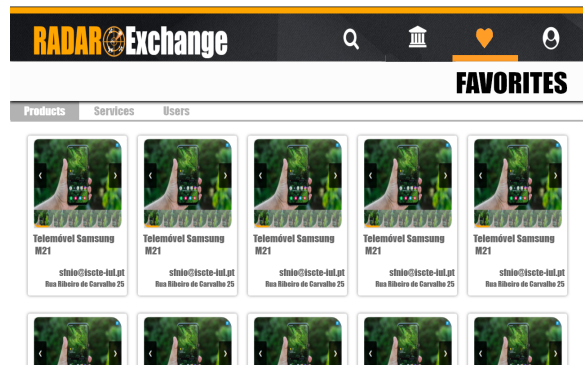


(H) MyStore Page - Registry list

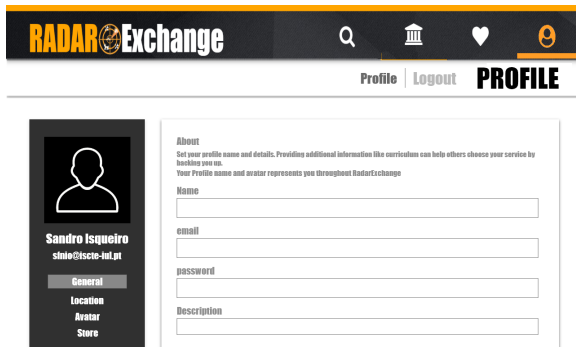
FIGURE 16. Website's Layout Design



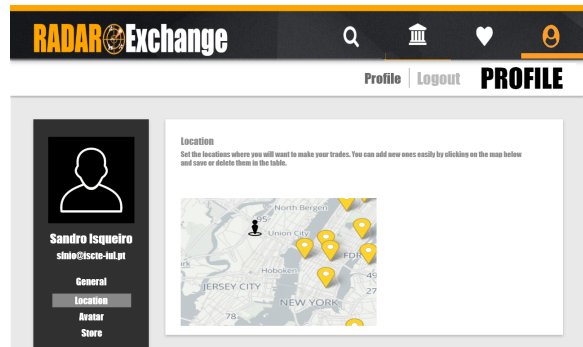
(A) MyStore Page - Registry form



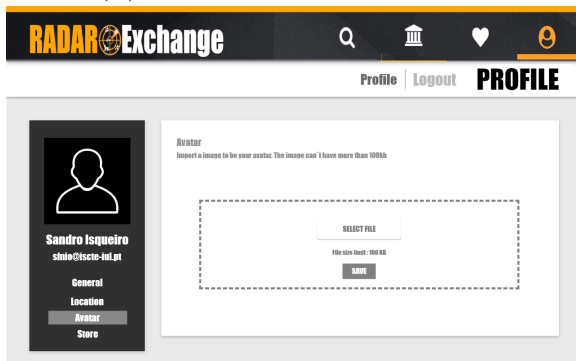
(B) Favorites Page



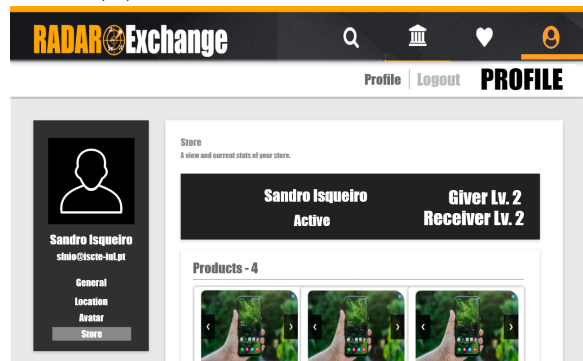
(C) Profile Page - General info



(D) Profile Page - locations info



(E) MyStore Page - avatar



(F) Profile Page - store info

FIGURE 17. Website's Layout Design

## CHAPTER 4

# Prototype

This chapter is broken into four sections that detail the conceived elements and the final design of the solution. We begin with the database created to support the site, then move on to the prototype's UI and features, and lastly its deployment.

### 4.1. Database

When developing an application or website, it is critical to pay close attention to how the frontend communicates with the backend. However, the construction and design of the database, initially, are more crucial. The relationships between the data will lead to the creation of the database schema. A database schema is an abstract design that specifies how the data will be stored in a database. It determines the data organization as well as the relationships between tables in a database.

As discussed in Chapter 3, PostgreSQL11 was chosen as the database to secure and manage the website's data, assisted by pgAdmin4 which will be used to access the database and perform the necessary actions. The database that supports the solution was set up in a simple and quick manner using this tool, following Entity Relationship Diagram, Figure 18, created based on the Class Diagram (Figure 14).

As shown in Figure 18, the database is made up of nine tables: *users*, *confirmation\_code*, *products*, *locations*, *products\_location*, *likes*, *subscriptions*, and *trades*, which are described further below.

The table *users*, Figure 19, is used to store the user's personal information. Its primary key, *user\_id*, is used as a foreign key in the remaining tables, acting as a link between the data associated with that user. There are also the attributes *name*, *email*, *password*, which are filled with the data entered during account creation and used for login. The attributes *description*, *rating\_offer*, *rating\_request* and *avatar* are used to store additional information about the user, e.g., a small introduction for the other users, as well as the levels used in the system of calculating contributions that was implemented, and the image that the user chose to be his avatar, respectively. The last one, *active*, is a boolean that indicates whether the account is active or not.

The *confirmation\_code* (Figure 20), table located next to the *users* table is meant to store the confirmation codes generated for each user, establishing a one-to-one relationship with the user table. Aside from its primary key, *id\_conf\_code*, there are also the attributes *code*, which are assigned to the confirmation code, and *user\_id*, which acts as a foreign key, operating as a link between the user to whom the code was assigned.



FIGURE 18. Database Schema

There's also the *subscriptions* table, Figure 21, which establishes a many-to-one association with the user table. This table keeps track of every subscription that one user makes to another. To that end, in addition to the primary key, *subs\_id*, the attributes that round out this table are *subscriber\_id* and *subscription\_id*, both of which indicate the id of the user who subscribed, as well as whom it subscribes.

With a many-to-one relationship and the same importance as the table *users*, the table *products*, Figure 22, stores all products entered by users. This table contains the primary key, *product\_id*, as well as the attributes that describe the product, such as *title*, *proposal*, *type*, *category*, *description*, and *product\_date*. Aside from these attributes, there is also



public
<b>users</b>
user_id integer
name character varying(40)
email character varying(100)
password character varying
description character varying(200)
rating_offer integer
rating_request integer
avatar text
active boolean

FIGURE 19. Database Schema - users Table

public
<b>confirmation_code</b>
id_conf_code integer
code integer
user_id integer

FIGURE 20. Database Schema - confirmation\_code Table

public
<b>subscriptions</b>
subs_id integer
subscriber_id integer
subscription_id integer

FIGURE 21. Database Schema - subscription Table

*dataimage*, which stores the images data inserted by the user, and *status*, which informs the user about the product's availability. Finally, the foreign key *offerer\_id* indicates to whom the product belongs.

With the objective of storing users' locations, the table *locations*, Figure 24, was created with a many-to-one relationship with the table users. In this table, in addition to the primary key *location\_id*, there are attributes such as *address* and *coordinates*, which contain the geographical information about the location selected by the user, and a foreign key, *users\_id* that indicates which user belongs that location.

In order to indicate which locations the product is available in, a link was created between the *product* and *location* tables. However, this connection appears to be a many-to-many association, and, as such, the creation of an associative table, *products\_location*, that indicates the locations of each product was required, Figure 23. This table is made up of its primary key, *prod\_loc\_id*, as well as its foreign keys, *location\_id* and *product\_id*.

The final three tables complete the database, and they are *trades*, *likes*, and *messages*, and they have a many-to-one relationship with the table users. The *trades* table, Figure 25, was created to track transactions between users and, as a result, to allow users to be classified according to their record of offers and requests. The table contains the primary key *trade\_id*, as well as the attributes *requester\_id*, *offerer\_id*, *product\_id*, and *trade\_date*, which indicate which users participated in the transaction, as well as the product traded and the date of the transaction. There are also the attributes *validated*, which were created to prevent the registration of unconfirmed transactions, with the validation of the same being required by the user who requested the product, as well as the attributes *liked* and *comment*, which indicate the user's satisfaction with the product.

public
<b>products</b>
product_id integer
title character varying
proposal character varying (40)
type character varying(40)
category character varying
description character varying
product_date date
dataimage text[]
status character varying
offerer_id integer

FIGURE 22. Database Schema - products Table

public
<b>products_location</b>
prod_loc_id integer
location_id integer
product_id integer

FIGURE 23. Database Schema - products\_location Table

public
<b>locations</b>
location_id integer
address text
coordinates geometry
users_id integer

FIGURE 24. Database Schema - locations Table

public
<b>trades</b>
trade_id integer
requester_id integer
offerer_id integer
product_id integer
validated integer
liked text
comment text
trade_date date

FIGURE 25. Database Schema - trades Table

public
<b>likes</b>
like_id integer
fond_id integer
product_id integer

FIGURE 26. Database Schema - likes Table

public
<b>messages</b>
message_id integer
sender_id integer
receiver_id integer
topic_id integer
message_text text
date timestamp without time zone

FIGURE 27. Database Schema - messages Table

The *likes* table, Figure 26 is used to track the user's desired products. As a simple table, the *likes* table is made up of its primary key, *like\_id*, and its secondary keys, *fond\_id* and *product\_id*, which indicate the user and the product in question.

Finally, the *message* table, Figure 27, was created to keep track of user conversations. This table is made by its primary key, *message\_id*, as well as the attributes that make up a message, such as *sender\_id* and *receiver\_id*, which indicate the sender and receiver, *topic\_id*, which indicates the product about which the users are conversing, *message\_text*, which is the body of the message, and *date*, which is the date the message was sent.

#### 4.1.1. PostGIS Extension

It is important to report that, in order to take advantage of the PostGIS extension, it was necessary to create it, as can be seen in Listing 1.

---

```
CREATE EXTENSION postgis;
```

---

LISTING 1. PostGIS Extension

#### 4.1.2. Database Connection

The connection to the database, which is one of the most crucial aspects, enables for data query and entry. This connection is accomplished using SSH, a network protocol that encrypts all incoming communication in order to eliminate spying, data sequestration, connection failure, and other types of attacks or malfunctions.

---

```
$conn = pg_connect("dbname='RadarExchange'  
    user='*****' password='*****'  
    host='*****'  
    port='5432'");  
if (!$conn) { echo 0; }
```

---

LISTING 2. Database Connection

As may be seen in Listing 2, after the creation, the connection to the data base is checked. In the event of an error, this response is handled on the client side by redirecting the client to the error page.

#### 4.2. User Interface

When developing the solution presented in this dissertation, the goal was to make the user experience straightforward and intuitive, requiring minimal effort on the user's side to get a satisfactory usage experience.

With this in mind, the Wireframes shown in Chapter 3.2.3 were created and were made available to a few users in order to obtain input on their appearance. Furthermore, various parts of the functions and the overall process were thoroughly reviewed, which resulted in the modification of some features previously shown in Wireframes.

Following the sequence of actions, the user is welcomed by the Homepage when they access the website, Figure 28. After some initial improvements in terms of layout and information provided, this page includes information about the purpose of this service, how it works, and how to contact the website managers, in addition to links to register and login to the platform. To attract the attention of new users, a display area of some available items (with minimal information) has been added.

To improve site security, a more rigorous process for platform access was implemented, pictured in Figure 29. When creating an account, the user must fill out the form and, if the submitted data passes validation, the user is directed to the confirmation page, where

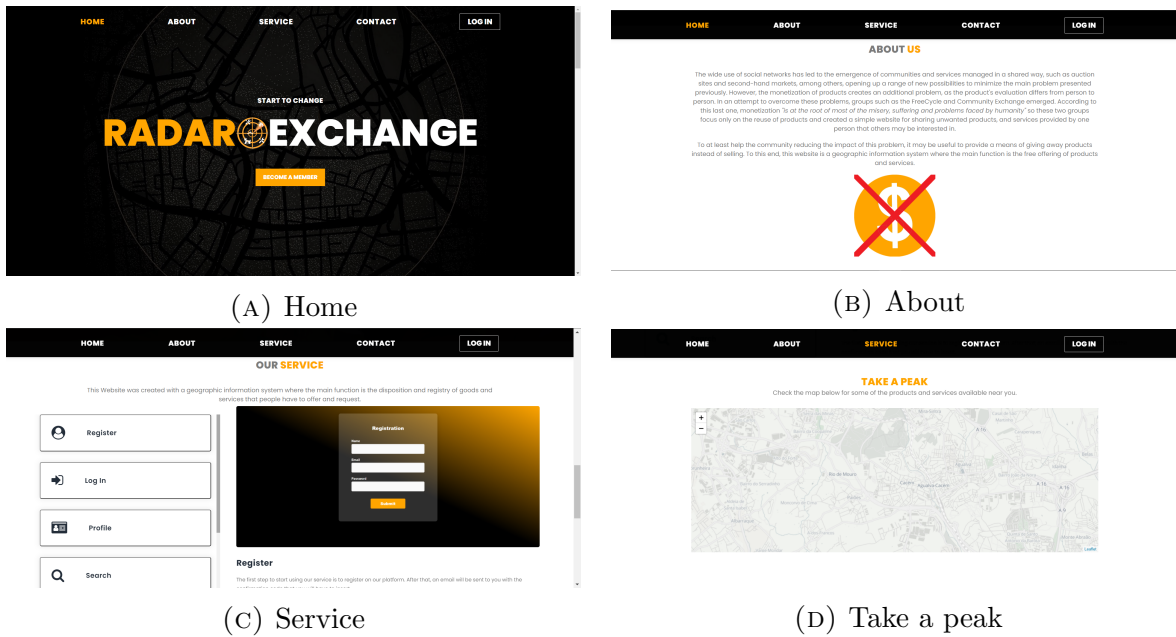


FIGURE 28. Website's Interface - Homepage

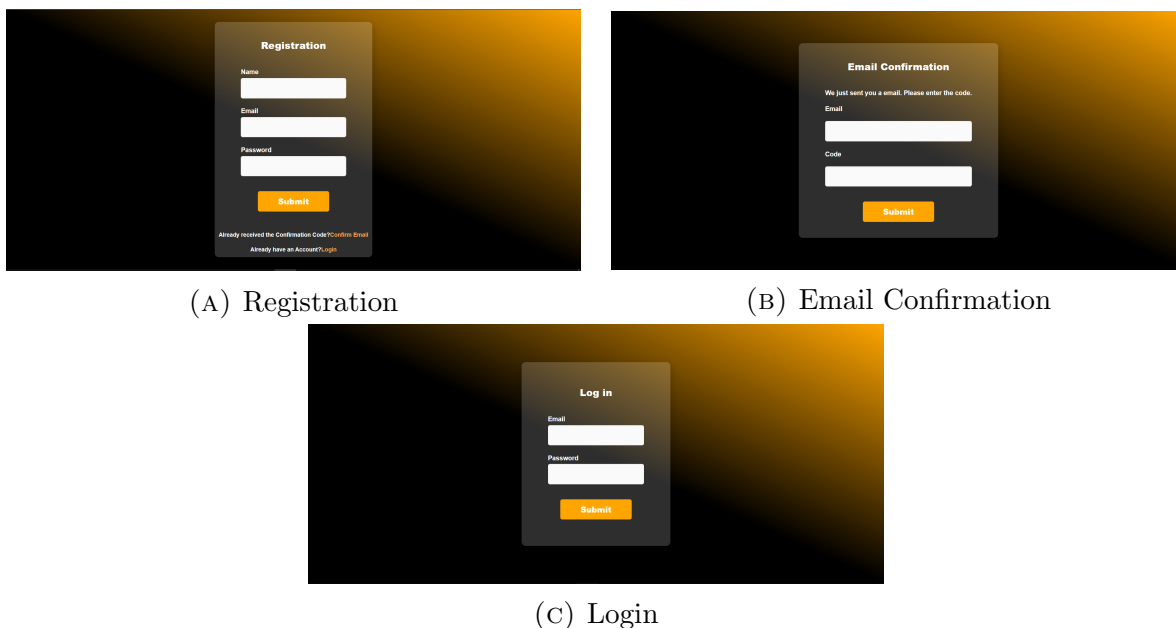


FIGURE 29. Website's Interface - Access process

he must input the code that was sent to the email address specified when the account was created. Since this website's objective is to enable transactions of products and services between users, this extra step avoids the creation of fraudulent accounts and contributes to its reliability. After properly entering the code, the user is able to log in to the platform.

When user logs in, he is sent to the platform's Search page, Figure 30, which has two pages: Ads and Members. The Ads page, as described in Section 3, allows the search for products and services by their and the user's location. This search can be carried out using the map or the list, and it may also filter the products and services seen. The information about the product chosen by the user is shown in the form of a window,



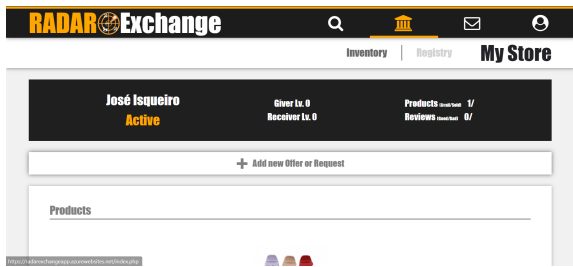
FIGURE 30. Website's Interface - Search Header

which provides all relevant information about it as well as the option to apply for it. The Members page displays a list of all users on the site, as well as information on the items they are offering or requesting.

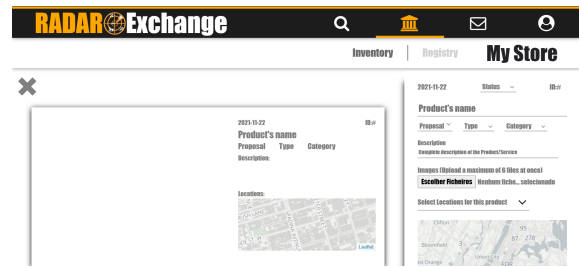
The *MyStore* header, Figure 31, was one of the parts that stayed consistent with the original design, with its content divided into two pages: Inventory and Registry. The Inventory page is a list of all products and services offered by the user, with the option to add, edit and delete products at any time. The Registry page is dedicated to the registration of the user's exchanges. The user must register any transactions he does in order to raise his level and get reputation.

Another modification to the original design is the Message header, Figure 32. This page contains all of the occurred dialogues related to the products that have been requested by and to the user, allowing for discussion between the users to exchange information about the product and plan a possible transaction.

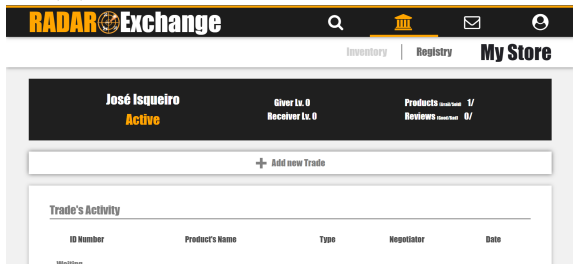
Finally, there have been some design adjustments to the Profile header. It contains the user's favorites and subscriptions, in addition to the user's personal information and the places where the user is available to make the transactions. It also contains the logout and delete account options.



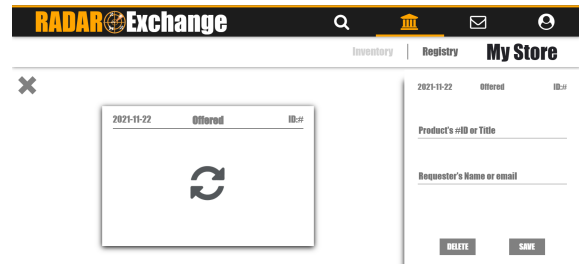
(A) MyStore - Inventory product's list



(B) MyStore - Inventory product's form



(c) MyStore - Trades activity



(d) MyStore - Trades form

FIGURE 31. Website's Interface - MyStore

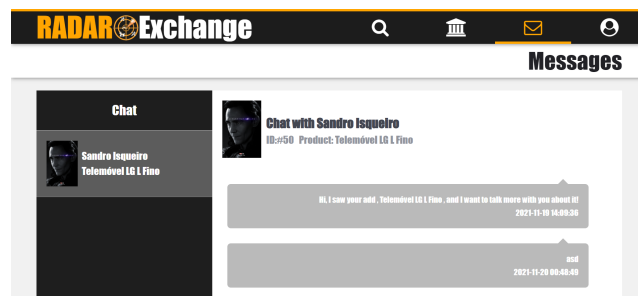


FIGURE 32. Website's Interface - Message

In addition, if the server is unable to interact with the database, the user will be sent to an error page, Figure 34.

As a web design strategy, Responsive design [25] seeks to make web pages appear effectively across a number of devices ranging from small to large display sizes in order to assure usability and satisfaction.

This type of design was achieved in the developed solution through the use of media queries, which allow the page to utilize alternative CSS style rules based on the features of the device on which the site is being presented, as seen in Appendix C.

### 4.3. Features

During the solution's development, the Use Case Diagram, Figure 13, was taken into account. This section describes the mechanisms and procedures used to implement these fundamental tasks, as well as other features that make the solution more complete.

#### 4.3.1. Account Creation

The creation of a user account is a complex process. As shown in Figure 29a, the user only needs to enter his name, email address, and password. However, in order to make the

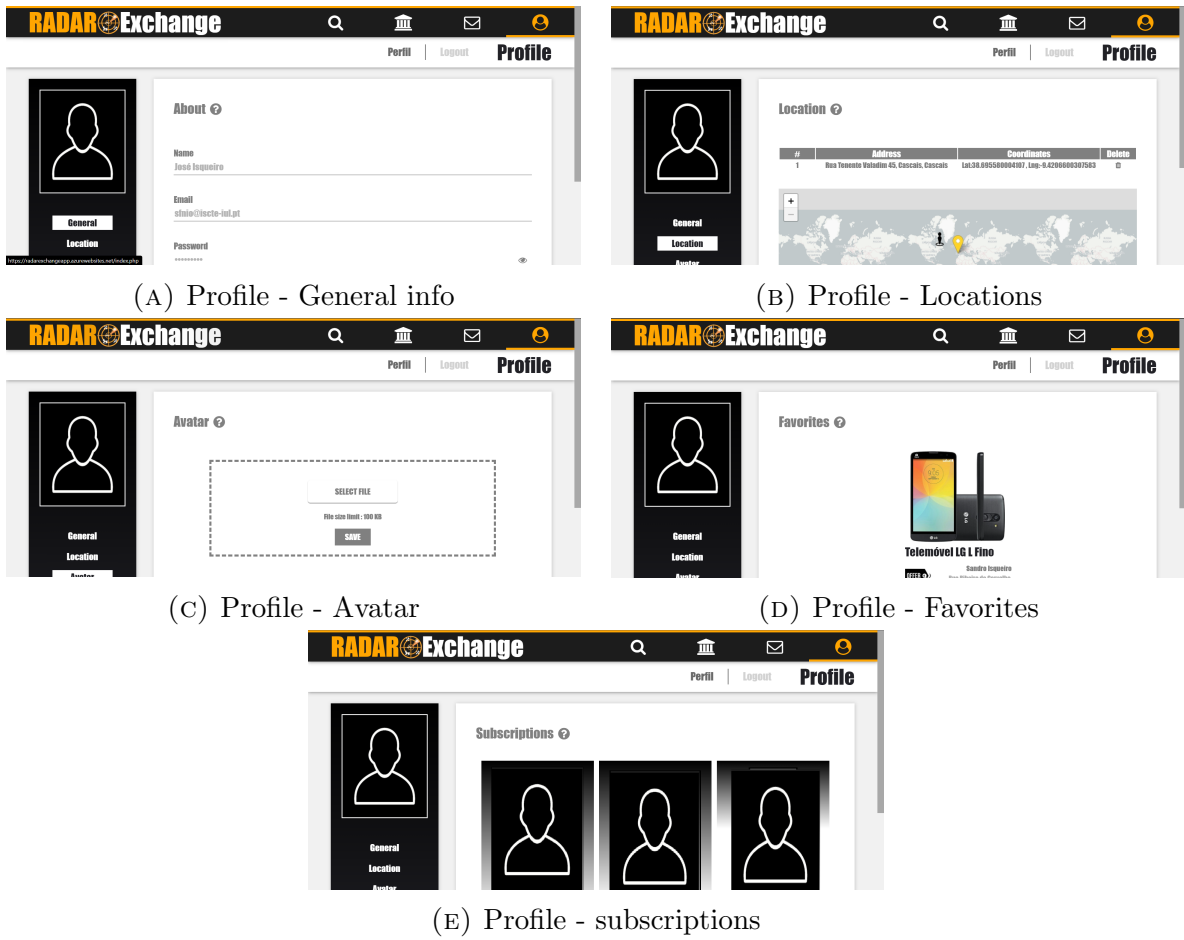


FIGURE 33. Website's Interface - Profile

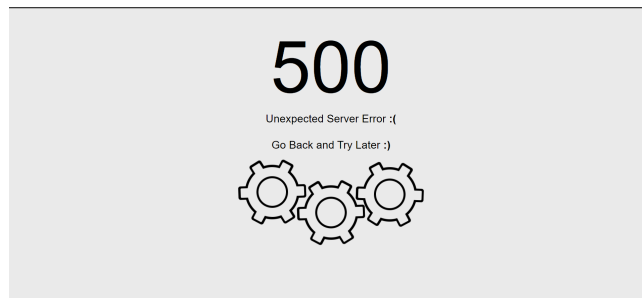


FIGURE 34. Website's Interface - Error page

platform more secure and to prevent the creation of duplicate or false accounts: processes to alert to such situations are carried out on the server's side.

By using AJAX, a technology for querying web servers from a web page, it is easy to validate the arguments without leaving the page. The parameters given by the user are then transmitted to the *newprofile.php* service, which checks if the email address is already in use by another user. If everything is in order, an inactive account with an encrypted password is created.

The user is then redirect to the *Email Confirmation* page, Figure 29b, after calling the service *email.php*. This service creates and saves a unique confirmation code for the user,

which is subsequently sent to the supplied email address. Using once again AJAX, when the user submits the form, the *codeconfirm.php* service checks to see if the code entered is the same as the one created for the email address in question. If the sequence supplied corresponds to the code provided to the user, the account is enabled and the user may log in.

### 4.3.2. Login & Logout

In contrast with creating a new account, login and logout are simple processes to implement, defined in the files *login.php* and *logout.php*.

The credentials supplied in the login form, in Figure 29c, are checked in the database using AJAX without leaving the page. If the credentials are approved, as well as the account's active state, the user's information is kept in a session and will be shared across the platform.

When a user logs out, the system destroys the session that was generated for the user and redirects the user to the website's homepage.

### 4.3.3. Leaflet Map

As mentioned in Chapter 3, the javascript Leaflet library, which allows for a quick implementation on a web page, was employed. To use this library, it is necessary to include the Leaflet CSS and Javascript files in the page's head section, as seen in Listing 3.

---

```
<link rel="stylesheet"
  href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
  integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUK
LsCC5CXdbqCmblAshOMAS6/keqq/sMZM19scR4PsZChSR7A==" crossorigin=""/>
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
  integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448h0EQiglfqkJs1NOQV44c
WnUrBc8PkA0cXy20w0v1aXaVUearIOBhiXZ5V3ynxwA==" crossorigin=""></script>
```

---

LISTING 3. Leaflet's CSS and Javascript files

The map can be initialized by providing the *id* of the *div* where the map will be displayed and configuring its view to a set of geographical places and zoom level. The tile images that will be utilized to make the map must also be specified.

---

```
var mymap = L.map('mapid', {zoomControl: false}).setView([40.771,
  -74.0], 15);
L.tileLayer('https://tiles.stadiamaps.com/tiles/alidade_smooth/
{z}/{x}/{y}{r}.png', {attribution:false}).addTo(mymap);
```

---

LISTING 4. Map Creation



---

```
<link rel="stylesheet" href="style/MarkerCluster.css" />
<link rel="stylesheet" href="style/MarkerCluster.Default.css" />
<script src="script/leaflet.markercluster-src.js"></script>
```

---

LISTING 5. ClusterMarker CSS and Javascript file

---

```
var markers = L.markerClusterGroup({ spiderfyOnMaxZoom: false,
    showCoverageOnHover: false, zoomToBoundsOnClick: true });

markers.on('clusterclick', function (a) { a.layer.spiderfy(); });
```

---

LISTING 6. MarkerCluster Creation

#### 4.3.4. Leaflet Plugins

Despite the fact that Leaflet focuses on a core set of resources, there are a number of plugins [26] that enhance the user's interaction with the map. As a result, some plugins were added to enhance the functionality of the system.

To prevent situations where the markers overlap even on the current zoom level, which hinders interacting with them, the plugin *MarkerCluster* [27] was utilized. This plugin supports marker clustering, which is a strategy for grouping together markers that are close to each other on the current zoom level. Respectively, Listing 5 and 6 show the necessary files to implement these plugins and how variable *markers* was created as a Cluster of markers. The Cluster plugin included the spiderfy availability event, which allows for the display of overlapping markers in the shape of a web, Figure 45.

To provide more information on what is surrounding the user, *EdgeMarker* [28], a Leaflet plugin which allows to indicate Markers, Circles and CircleMarkers that are outside of the current view by displaying indicators at the edges of the map, was employed. Listings 7 and 8 show the files required to implement the plugins, as well as how variable *edgeMarkersLayer* was produced as edgemarkers for each marker that is positioned in a specific radius, outside the region seen by the user, respectively.

---

```
<script src="script/Leaflet.EdgeMarker.js"></script>
```

---

LISTING 7. EdgeMarker Javascript file

The outcome of using these two plugins can be seen below in Figure 46.

---

```

var edgeMarkerLayer = L.edgeMarker({
  findEdge : function(mymap){return L.bounds([0,0],
    mymap.getSize());},
  icon: L.icon({
    imageUrl : 'images/edge-arrow-marker2.png',
    clickable: true,
    iconSize: [48,48],
    iconAnchor: [24, 24]
  })
}).addTo(mymap);

```

---

LISTING 8. Edge Marker

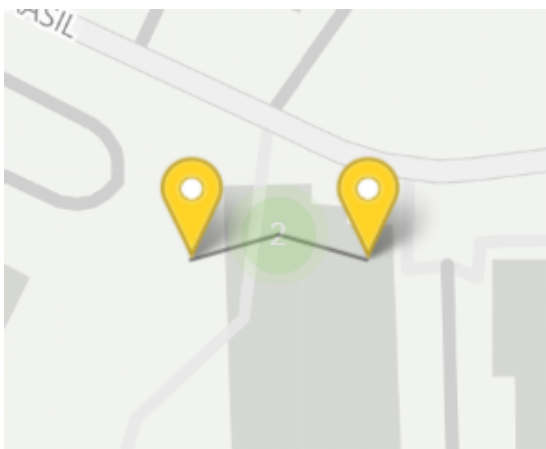


FIGURE 35. Spiderfy Cluster Marker

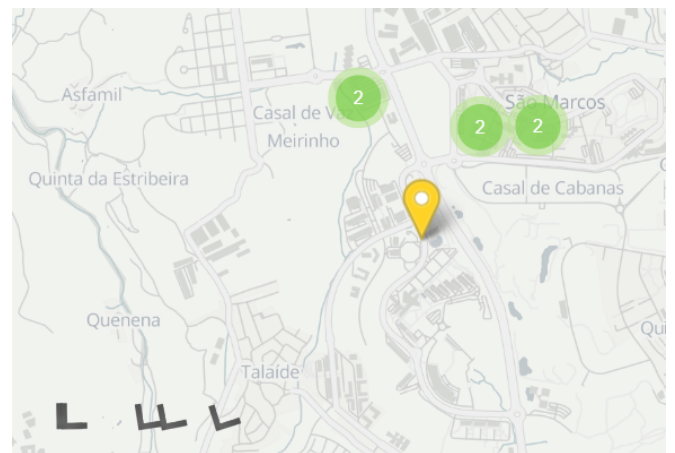


FIGURE 36. Cluster and Edge Marker

In this solution, one more additional plugin was used. ESRI Leaflet [29] is a small set of JavaScript tools for interacting with ArcGIS location services. This tool allowed the geocoding of coordinates and addresses from a geolocated point, as shown in Figure 37.



**Address:** Avenida das Forças Armadas, Lisboa, Lisboa  
**Coordinates:** 38.74709905449215, -9.155286807261604

FIGURE 37. ESRI Geocoding

---

```
<script src="https://cdn-geoweb.s3.amazonaws.com/esri-leaflet/
1.0.0-rc.2/esri-leaflet.js"></script>
<script src="https://cdn-geoweb.s3.amazonaws.com/esri-leaflet-geocoder/
0.0.1-beta.5/esri-leaflet-geocoder.js"></script>
<link rel="stylesheet" type="text/css" href="https://cdn-geoweb.s3.
amazonaws.com/esri-leaflet-geocoder/0.0.1-beta.5/esri-leaflet-geocoder
.css">
```

---

LISTING 9. ESRI Javascript and CSS file

---

```
var geocodeService = new L.esri.Services.Geocoding();
mymap.on('click', function(e){
    geocodeService.reverse(e.latlng, {}, function(error, result){
        $('#addresslocation').text(result.address + ", " +
            result.neighborhood + ", " + result.city );
        $('#latlocation').text(result.latlng.lat);
        $('#lnglocation').text(result.latlng.lng);
        marker1.setLatLng(e.latlng).bindPopup(result.address).openPopup();
    });
});
```

---

LISTING 10. Reverse Geolocation

To accomplish this, it was necessary to include the ESRI Javascript and CSS files, seen in Listing 9, as well as the inclusion of an `EventListener()` function from Leaflet, which performs reverse geocoding on the coordinates of every selected location in the map and determines its address, printing it on the web page, Listing 10.

#### 4.3.5. Account Edition

The ability of the user to edit his or her account is a feature that is necessary in web and mobile applications.

This functionality can be found in the Profile section, which contains the user's personal information as well as his or her favorites and subscriptions. Personal information is divided into two sections: *general* and *avatar*.

The general information is presented in a form, as shown in Figure 33a, where the user's name, email address, and password are displayed. Any changes submitted are sent as arguments to the *savegeneralinfo.php* service via AJAX, where the data present in the database is updated. If a user wishes to delete his or her account, the service *deleteaccount.php* is invoked, where the status of the account, represented by the active column of the users table in the database, is set to false.

The avatar section, seen in Figure 33b, allows the user to upload an image to serve as his or her avatar on the platform. This process is carried out by updating the stored image data in the database via the *saveuploadimage.php* service, which is then used throughout the platform.

### 4.3.6. User's Locations

As the foundation of the developed solution, in the profile separator, the user has the ability to select the locations where he is available for product exchanges with others.

As shown in Figure 33c, the menu *locations* of the profile separator has a map whose implementation complied to the requirements outlined in Chapter 4.3.3. In addition, the ESRI plugin was implemented in order to identify the locations selected by the user as a trading point.

When a user presses the save locations button, the point's coordinates are sent as arguments to the *savelocation.php* service, along with the address determined by the plugin. In addition, the ESRI plugin was applied in order to geocode the locations selected by the user as a trading point.

To be able to store the coordinates in the data base as spatial coordinates, it is necessary to process them, Listing 11. This procedure is completed in two steps:

- (1) Using the PostGIS constructor `ST_POINT (X,Y)`, create the geometric point with the given coordinates;
- (2) Next, identify the Spatial Reference Identifier (SRID). In this case, the SRID 4326 was used, which corresponds to "longitude/latitude on the WGS84 spheroid," the spacial reference system most commonly used in cartography, geodesy, and satellite navigation.

---

```
INSERT INTO locations (users_id, address, coordinates) values ($iduser
, '$address', ST_SetSRID( ST_Point( $lat, $lng), 4326))
```

---

LISTING 11. Spatial data processing

### 4.3.7. Product Search

One of the primary functions of the developed solution is the search for products. Since the core focus of the solution is spatial data, it is critical that this feature focuses on how to make spatial data available to the user.

As a result, the search for products is available as a map, as shown in Figure 30a, resorting to Leaflet and its previous mentioned plugins for displaying results: ClusterMarker and EdgeMarker. Following the map's creation described in Section 4.3.3, the next step is to add product location markers to the map. AJAX was once again used, this time to perform a search for all available products and services via the *products\_geojson.php* service, which returns the required information in JSON format. Based on the information gathered, a marker is created for each product's location, with a popup (Listing 12). It is important to note that in order to process the coordinates into the desired geometric format, the function `ST_AsGeoJSON(geometry data)` must be used, which converts the data to a GeoJSON object.

---

```

$.ajax({
  type: "POST",
  url: "products_geojson.php",
  success:function(result){
    var obj=JSON.parse(result);
    for (feature of obj.features) {
      for (i = 0; i < feature.properties.coordinates.length; i++) {
        markers.addLayer(L.marker([feature.properties.
          coordinates[i].coordinates[0],feature.properties.
          coordinates[i].coordinates[1]},{icon: goldIcon}
        ).bindPopup(...));
      }
    }
  },
  error: function(result) {
    window.location.replace("500ServerError.php");
  }
});
mymap.addLayer(markers);

```

---

LISTING 12. Marker Creation

The map's view is updated based on the user's location using Leaflet's `locate` function, allowing the user to always see the products and services that are closest to him, as seen in Listing 13.

---

```

window.onload = function() {mymap.locate({setView: 8, watch: true})
  .on('locationerror', onLocationError)};

```

---

LISTING 13. Update map view to user's location

#### 4.3.8. Product Request

Another key function is product requesting. This functionality is available to the user through the product information window (Figure 38), which, when selected, calls, using AJAX, the service *email.php*. This service, which was previously used to send confirmation codes, is now used to send an email notifying the product's owner that there is interest in the product.

Aside from sending this email, a message is also "sent" to the product's owner via the platform. This message is generated automatically and saved in the database's *messages* table, thereby initiating a conversation between the two users (proprietary and solicitant) and establishing communication between them, Figure 39.

If this procedure does not generate errors, an alert will be displayed to inform the user that the product has been requested and a message will be sent to the owner. Additionally, the product is saved in the user's favorites.



FIGURE 38. Product Request bottom

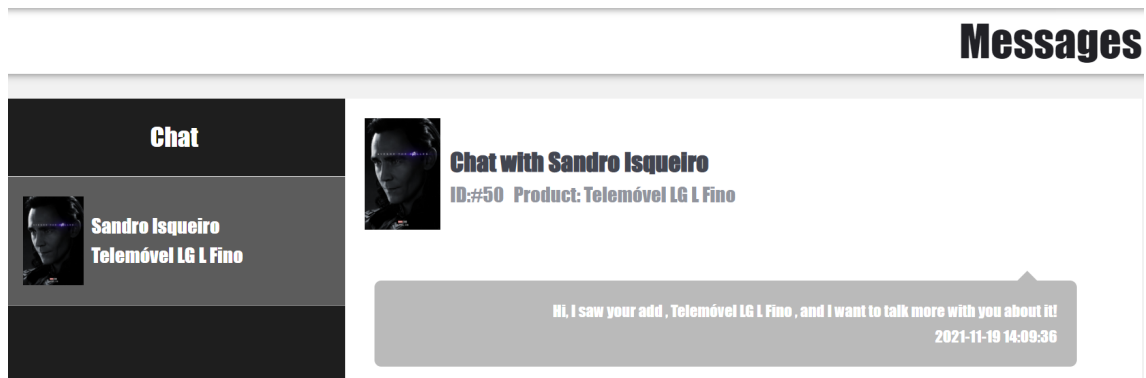


FIGURE 39. Request Message

#### 4.3.9. Liking of products

The function of adding a product to the user's favorites is a very straightforward operation, which is based on the status of the *like button* that the user can see and modify in the product's information window.

The function *mylike*, in Listing 14, is responsible for initiating this process when the like button is pressed. In this function, using AJAX to call the service *likeproduct.php*, supplying the arguments needed to make the desired changes in the database's *likes* table. These arguments are dependent on the current state of the like button, which indicates whether the product should be liked and the corresponding data should be registered in the *likes* table, or whether the product should be disliked and the corresponding data should be removed from the table.

After successfully implementing the changes to the database, the class of the button is changed to reflect the current state, Figure 40.

---

```

function mylike(id){
    $.ajax({
        url : "likeproduct.php",
        type : "POST",
        data : (document.getElementById("modallike").getAttribute("class")
            == "fa fa-heart-o fa-2x") ? {do:"like", idprod:id, iduser:<?php
            echo $_SESSION['id_user'];?>} : {do:"dislike", idprod:id,
            iduser:<?php echo $_SESSION['id_user'];?>},
        success:function(result){
            if(result==1){
                if(document.getElementById("modallike").getAttribute("class")
                    == "fa fa-heart-o fa-2x"){
                    $("#modallike").removeClass("fa-heart-o fa-2x");
                    $('#modallike').addClass("fa-heart fa-2x");
                }else{
                    $('#modallike').removeClass("fa-heart fa-2x");
                    $("#modallike").addClass("fa-heart-o fa-2x");
                }
            }
            else{ window.location.replace('500ServerError.php'); }
        }
    });
}

```

---

LISTING 14. Mylike Function



FIGURE 40. Like Button Status

#### 4.3.10. User Subscription

Subscribing to a user is as simple as adding a product to the favorites, and is available on the platform's member page.

In the same way as the function *mylike* is responsible for starting the process once the user selects the Follow button. This function, Listing 15, invokes the *subscribe.php* service through AJAX and sends the necessary arguments for registering or unregistering a subscription in the database's *subs* table. One of the required arguments is the text displayed on the button that informs whether the subscription is created or removed (if it already exists).

After successfully implementing the changes to the database, the text and class of the button is changed to reflect the current state, Figure 41.

---

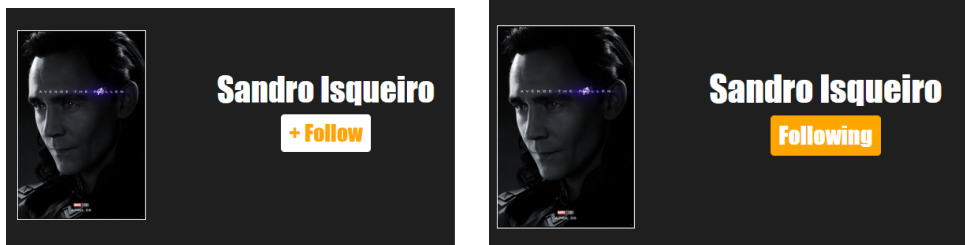
```

$(".follow-button").click(function subscribe(){
    $.ajax({
        url : "subscribe.php",
        type : "POST",
        data : ($("#follow-button").text() == "+ Follow") ? {do:"follow",
            iduser:<?php echo $_REQUEST["id"];?>, idfollower:<?php echo
            $_SESSION['id_user'];?>} : {do:"unfollow", iduser:<?php echo
            $_REQUEST["id"];?>, idfollower:<?php echo
            $_SESSION['id_user'];?>} ,
        success:function(result){
            if(result==1){
                if ($("#follow-button").text() == "+ Follow") {
                    $("#follow-button").text("Following");
                    $("#follow-button").toggleClass('active');
                }
                else{
                    $("#follow-button").text("+ Follow");
                    $("#follow-button").toggleClass('active');
                }
            }
            else{ window.location.href="500ServerError.php"; }
        }
    });
});

```

---

LISTING 15. Subscribe Function



(A) User unsubscribed

(B) User subscribed

FIGURE 41. Like Button Status

#### 4.3.11. Create and Edit Offers

Other key functions of the developed solution include product creation and subsequent edition. These features are available in the MyStore page, Figure 31a, where all of the user's products are displayed.

On this page, the user has the option of creating a new product by clicking the 'Add New Product' button, which displays the product creation form, Figure 31b. There are several fields in the form to detail the product that the user must fill in, namely the location selection. This is one of the most important fields since it determines where the user intends to trade the product. When the form is opened, a search is performed via



the service *getuserlocations.php* for the locations specified by the user in his profile. These locations will, in turn, serve as an option in the location selection field.

If all of the fields are filled out correctly, they are sent as parameters to the *saveproduct.php* service, which then inserts the data into the database table *products*.

The procedure is similar for product edition. When the user selects a product to edit, an AJAX search is performed, calling the service *editproduct.php*, which will search for information about the product in question, completing the form's fields with the same names. The same process as product creation is carried out when the user finishes editing. However, the service *saveproduct.php* performs an update to the data in the database.

If the user decides to delete the product, the service *deleteproduct.php* is invoked, which deletes the data from the product table in the database.

#### 4.3.12. Transactions Registry

Finally, in order to give support to a reputation system for users based on the transactions that occur, the transaction registry was added.

This functionality, shown in Figure 31c, displays a list of the transactions executed by the user, which have three states: waiting, pendent, and confirmed. When a user exchanges a product or service, he must register the transaction using the form in Figure 31d, indicating the product exchanged and the user with whom the transaction was made. Upon submission of the form, the service *createtrade.php* is invoked to register in the data exchange database, which remains pending until the other user confirms, in order to prevent false registrations, that would falsify the user's rating.

All transactions that need confirmation are placed in the "waiting" area, where the user must confirm whether or not the transaction were completed and the status of the product. If the transaction was successful, the data is sent to the *confirmtrade.php* process, which updates the trade table in the database, confirming the transaction and completing the registration process. If the transaction was rejected, the *textitdeletetrade.php* service would be used to remove the transaction's registry. All confirmed transactions are now listed in the "confirmed" section.

The user rating process is very straightforward, with 1 point always being added to *rating\_offer* or *rating\_request*, depending on the user's position in the transaction in the *confirmtrade.php* service. Based on practical experience and observation of various mechanisms that use level classification, it was determined that the level of the user was exponentially proportional to the number of transactions executed. As a result, this level classification is made using simple equations  $\lfloor \log_2(\textit{rating\_offer}) \rfloor$  and  $\lfloor \log_2(\textit{rating\_request}) \rfloor$ .

#### 4.4. Deployment

Finally, in order to carry out an evaluation of the solution, as described in the following chapter, it was necessary to deploy it, and Microsoft Azure was used to accomplish this.

Microsoft Azure [30] is a cloud computing-based platform for running applications and services, consisting of a number of different tools. Its a service that allows enterprises

and developers to buy the processing and storage capabilities of Microsoft data centers to apply to their business.

To deploy the solution on Azure, an App Service Web App, Figure 42, was created, which allows to swiftly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. After configuring the service to support our solution (PHP 7.4), all that remained was to import it into the server. Consequently, an Azure database

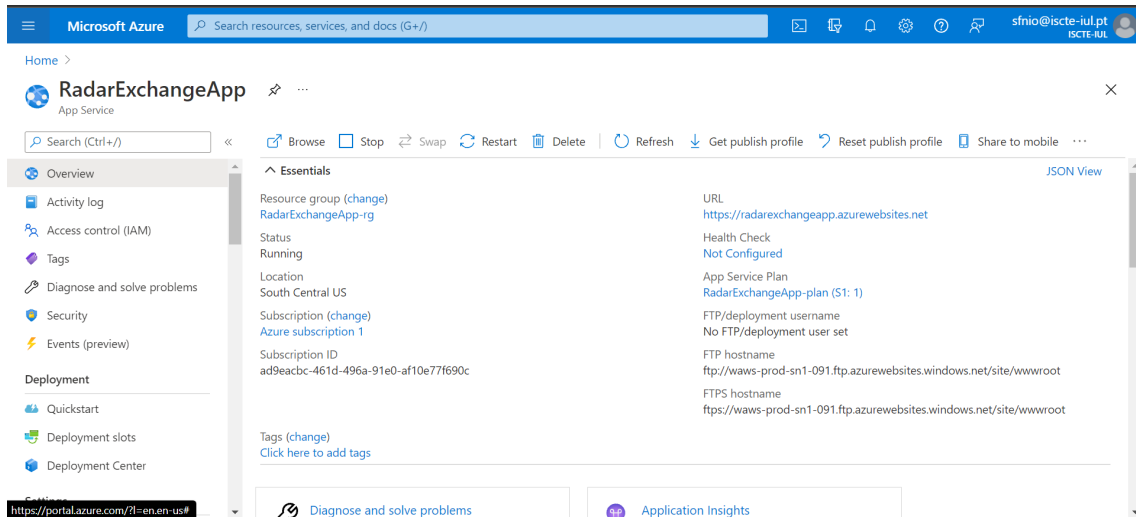


FIGURE 42. App Service

on a PostgreSQL server was required, into which the constructed model could be imported. After configuring the database, it was necessary to change the connection parameters in the PHP files so that an SSH connection could be established while consulting, inserting, and changing data in the database.

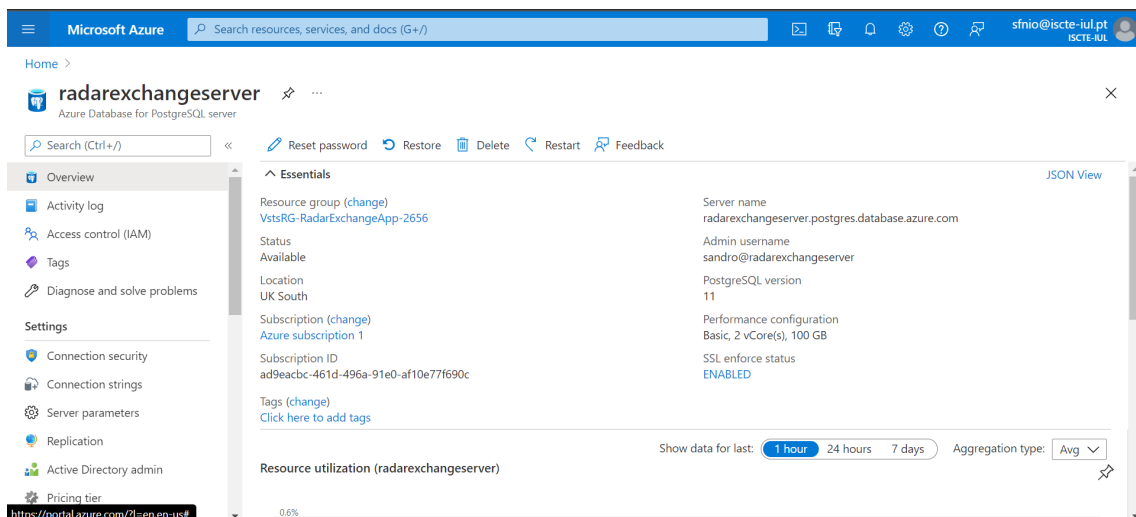


FIGURE 43. Azure Database for PostgreSQL server

Furthermore, the creation of the *SendGrid* service [31], Figure 44, was required in order to complete the validation of the users' emails and the sending of notifications to the same. SendGrid for Azure enables scalable email delivery.

Microsoft Azure Search resources, services, and docs (G+)

Home > RadarExchange SaaS

Search (Ctrl+/) Refresh Change plan Edit recurring billing View billing Cancel subscription Change Azure subscription

Product and plan details [View all your SaaS subscriptions](#)

Twilio SendGrid - Free 100 - Monthly  
SendGrid

✔ Your subscription was configured successfully. Select [Open SaaS Account on publisher's site](#) to navigate to the publisher's external website

[Open SaaS Account on publisher's site](#)

Free 100 - Monthly	Billing term & price	Created on	Current term	Recurring billing
✔ Subscribed Try it out! Integrate fast and explore features with 100 emails/day forever. <a href="#">Change plan</a> <a href="#">View this product in Marketplace</a> <a href="#">Terms of use</a>   <a href="#">Privacy policy</a>	Monthly Pricing not available	10/26/2021	Starts 10/26/2021 Ends 11/25/2021	On, renews on 11/26/2021 <a href="#">Edit recurring billing</a> <a href="#">View billing</a> Azure subscription: Azure subscription 1 <a href="#">Change Azure subscription</a> Resource group: radarexchangeapp-rg

FIGURE 44. SendGrid Service



## CHAPTER 5

### System Evaluation

This chapter's goal is to document the tests performed on the developed prototype in order to observe usability in task completion, obtain feedback, and analyze user experience in order to improve the product in the future.

#### 5.1. Usability test

The primary objective of the Usability Test [32] is to gain knowledge about potential problems encountered by users while using the solution, as well as any flaws in the design of the solution and/or functionalities that are missing or are not considered necessary, by analyzing one's experience. To obtain these data, the plan entails conducting usability tests, assigning tasks to users, monitoring them, and analyzing their feedback. The solution was then deployed on Azure services to conduct this tests.

As previously stated, tasks were assigned to participants in these tests in order to evaluate all of the functionalities included in the implemented solution. Among the tasks are:

- (1) Account creation;
- (2) Email confirmation;
- (3) Login;
- (4) Profile edition;
- (5) Insert localizations;
- (6) Create a product;
- (7) Product Search;
- (8) Make a product request and view the created conversation;

Following the completion of the tests, it was requested that the participants fill out a questionnaire in which they provide their feedback.

#### 5.2. Participant Profile

A usability test was conducted on 19 people aged 16 to 59 years, 12 of whom were male and 7 were female, to evaluate the developed solution. In general, no selection criteria were used for the participants, however, to make the feedback more critical, some of these participants are associated with the information technology field.

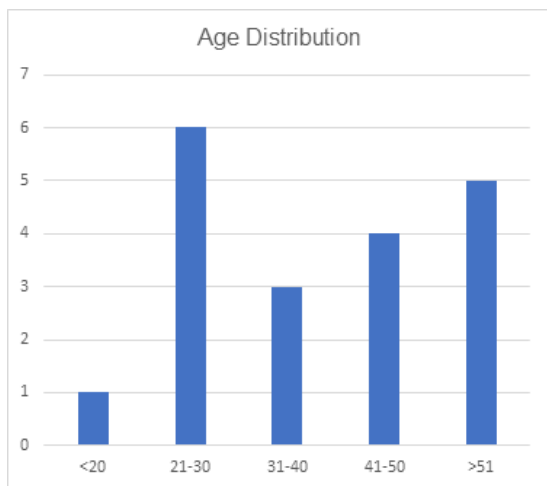


FIGURE 45. Participants Age

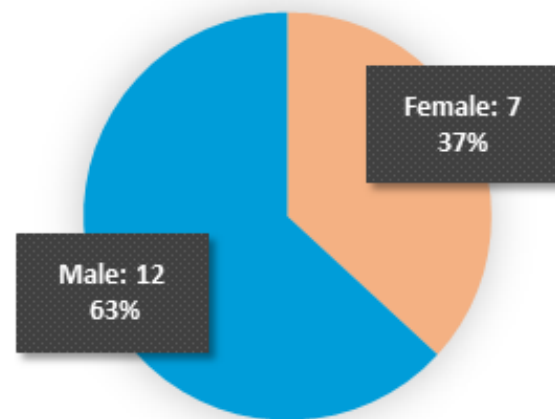


FIGURE 46. Participants Gender

### 5.3. Results

Participants were first introduced to the dissertation's theme before being given the prototype and various tasks to complete. Following the conclusion of the tests, the participants responded to a questionnaire, available in the Appendix B. The information from their responses was combined to form the graph in Figure 47.

In terms of the user interface, the majority of participants were pleased with the presented design, claiming that the solution has a user-friendly and simple-to-use interface. The solution also has a good distribution of information for the participants, making it easy and quick to understand.

However, some participants pointed out flaws in the presented interface, claiming that it had an unrecognizable font style and an insufficient size in some places. These errors were later investigated, and it was determined that they were caused by the use of web browsers other than Google Chrome, such as Safari, and operating systems other than Windows. This evaluation was able to detect a misrepresentation of the styles used between the browsers and operating systems, resulting in a different view of the interface.

In terms of the solution's functionality and performance, once again, the majority of participants were able to complete all of the tasks that were assigned, demonstrating the system's correct functioning, that is, the implemented functionalities perform the desired process. However, unlike the interface, several participants, who have a general desire for fast and efficient applications, were dissatisfied with the solution's performance, criticizing the delay in carrying out the tasks, due to a significant delay during certain processes of the solution, specifically the process of saving the information.

Following the submission of these responses, another investigation was conducted to determine the source of the problem, and it is estimated that the delay presented is due to the moment when the service requires a database query. Taking this into consideration, the cause of the delay could be the large physical distance between the server (USA) where

the solution is located, the database (United Kingdom), and the user, as the solution did not exhibit such issues during its development on localhost. No entanto, os testes são adicionados para confirmar essa hipótese.

In short, the solution received a mixed reaction from the participants; that is, while the interface design is very simple, pleasant, and appealing, the solution's performance raised some constructive criticisms in order to fully meet the needs and interests of the target audience. To add up, users had the opportunity to list some features they thought would improve the solution. In this list, there are functions that can improve the solution when it comes to the capabilities, such as:

- The addition of a search box for searching for locations by name;
- The addition of links in the user's location to redirect the map's view to the selected coordinates;
- The addition of new icons to differentiate product categories.

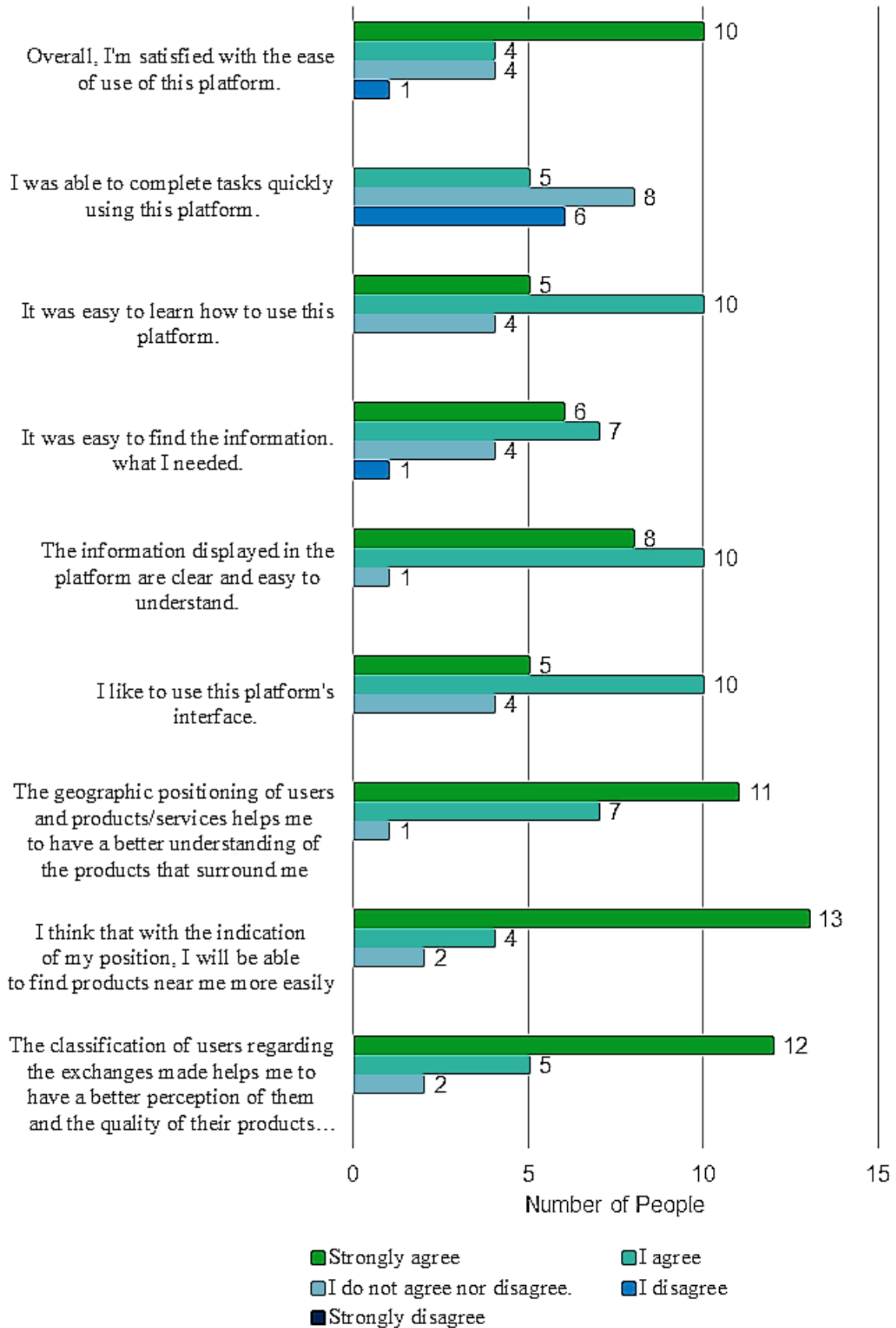


FIGURE 47. Feedback



## CHAPTER 6

### Conclusions

In this last chapter, a review of the completed work is performed, along with responses to research questions listed in the introduction, as well as acquired abilities, encountered problems, and future work.

#### 6.1. Overview

It was proposed that a geospatial information access system with a web interface would be developed. This system will function as a platform for users who, by adopting a collaborative economy, will be able to announce products and services that they wish to share with others, using the user's location as a starting point. This system will allow to differentiate itself from existing solutions, such as FreeCycle and Community Exchange, by making available to users the products and services that surround it in its current location.

The adopted method for the creation of the proposed solution included several key steps, with the following goals as its foundation: the study of geospatial information and the resources available for its manipulation and storage, the definition of requirements and the system's structure, the creation of the geospatial information system and its corresponding web interface, and the evaluation of the system's performance.

The solution was created using PHP, HTML, and CSS, with support from PostgreSQL's database and its spatial data extension, PostGIS. In general, the system was well received by the users that completed its evaluation, emphasizing the use of geolocation for product and service display. The proper operation of the services, as well as some system performance issues, were also discussed.

#### 6.2. Research Answers

In this dissertation's introduction, several research questions were raised with the goal of assessing the innovations introduced in the created solution that distinguished it from others.

The first question to be addressed is the impact that the solution, as a system based on geographic data, has on the user's perspective and interaction with it. This question was answered during the analysis of the answers given by the participants in the usability test questionnaire, where it was stated by an overwhelming majority that this type of system allows a better perspective of the products that surround the user, making it easier to find them.

The second question, regarding user classification, is also addressed in the questionnaire. Again, the majority of participants believe that categorizing uses based on transactions will help them have a better understanding of them, as well as the quality of the products and services available.

Based on the question, it was feasible to confirm the previous viewpoints, given that many popular programs use this type of information system, making it a waste for both the FreeCycle and the CommunityExchange not to use it.

### **6.3. Acquired Skills**

The development of this dissertation and the proposed solution were a chance to put the academic knowledge to practice.

During the many stages of the solution's development, it was possible to mature the knowledge and processes required for its building, such as identifying the system's requirements and designing its architecture and interface.

The topic of the dissertation and the new concepts implemented in contrast to existing solutions, that is, spatial data, compelled the conduct of a study that proved to be extremely useful for better perception, management, and use of existing resources (PostGIS) for these same data.

It also allowed for a better and deeper understanding of several of the programming languages studied in the Master's degree, notably PHP, HTML, CSS, JavaScript, and SQL.

### **6.4. Challenges & Future Work**

Despite the fact that the proposed solution has been completed, there are improvements that can be implemented in future work.

It will be necessary to address the problems detected during usability tests, specifically the system delay and the difference between the layout as shown in by the various browsers. In relation to the reported delay, it has already been hypothesized that the origin of the problem may be due to the availability of the application's resources on various servers. To put this theory to the test, all resources would need to be placed on a single server and the solution's functionality would need to be examined in these conditions. Concerning the second issue, it would be necessary to conduct research to justify this behavior. Nonetheless, one solution would be to change the sources used for a universal.

Apart from the suggestions made by users to add new features, challenges that needed to be dealt were discovered during the development of the solution. One of these issues is the storage of images in the database. The option of storing picture data in a database is not the best option. Instead, server storage is preferred because it utilizes less server resources and minimizes the amount of time and data sent to and from the database.

## **6.5. Final Appreciation**

Overall, the development of this dissertation presented an interesting and challenging topic on which to work. Its successful conclusion allowed the use of acquired knowledge in the classroom, as well as the expansion and expansion of the same.



## References

- [1] P. Ramsey and M. Leslie. 2. introduction — introduction to postgis. [Online]. Available: <https://postgis.net/workshops/postgis-intro/introduction.html>
- [2] Statcounter global stats - browser, os, search engine including mobile usage share. [Online]. Available: <https://gs.statcounter.com/>
- [3] Db-engines ranking. [Online]. Available: <https://db-engines.com/en/ranking>
- [4] Db-engines ranking. [Online]. Available: <https://db-engines.com/en/ranking/spatial+dbms>
- [5] Sharing economy - wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/Sharing\\_economy](https://en.wikipedia.org/wiki/Sharing_economy)
- [6] Freecycle. Freecycle.org. [Online]. Available: <https://www.freecycle.org/>
- [7] C. Exchange. Community-exchange.org. [Online]. Available: <https://www.community-exchange.org/home/>
- [8] —. Why we need a new exchange system — community exchange system. [Online]. Available: <https://www.community-exchange.org/home/how-it-works/why-we-need-a-new-money/>
- [9] K. C. Laudon and J. P. Laudon, *Management Information Systems*.
- [10] I. M. Alexandre, “Design science research,” University Lecture, 2020.
- [11] P. Schmitt, “Aplicação web utilizando api google maps.”
- [12] K. R. L. SOUSA, “Sistema colaborativo para criação de roteiros turísticos e interativos utilizando a api google maps.”
- [13] T. S. Batista, V. S. Batista, and V. I. de Faria, “Animaps: Um sistema de informação geográfica para a adoção de animais.”
- [14] PostGIS. Postgis.net. [Online]. Available: <https://postgis.net/docs/manual-3.1/reference.html>
- [15] M. Altaweel. Gis lounge. [Online]. Available: <https://www.gislounge.com/what-is-postgis/>
- [16] MySQL. Mysql :: Mysql 8.0 reference manual :: 11.4 spatial data types. [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/spatial-types.html>
- [17] CARTO. Unlock the power of spatial analysis — carto. [Online]. Available: <https://carto.com/>
- [18] Leaflet — an open-source javascript library for interactive maps. [Online]. Available: <https://leafletjs.com/>
- [19] Openlayers - welcome. [Online]. Available: <https://openlayers.org/>
- [20] Maps, geocoding, and navigation apis & sdks — mapbox. [Online]. Available: <https://www.mapbox.com/>
- [21] Wampserver. [Online]. Available: <https://www.wampserver.com/en/>
- [22] Class diagram - wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/Class\\_diagram](https://en.wikipedia.org/wiki/Class_diagram)
- [23] Component diagram - wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/Component\\_diagram](https://en.wikipedia.org/wiki/Component_diagram)
- [24] Adobe xd — ferramenta de colaboração e design da interface/experiência do usuário rápida e avançada. [Online]. Available: <https://www.adobe.com/pt/products/xd.html>
- [25] Html responsive web design. [Online]. Available: [https://www.w3schools.com/html/html\\_responsive.asp](https://www.w3schools.com/html/html_responsive.asp)
- [26] Plugins - leaflet - a javascript library for interactive maps. [Online]. Available: <https://leafletjs.com/plugins.html>

- [27] Leaflet.markercluster 0.1 released - leaflet - a javascript library for interactive maps. [Online]. Available: <https://leafletjs.com/2012/08/20/guest-post-markerclusterer-0-1-released.html>
- [28] Sharing economy - wikipedia. [Online]. Available: <https://github.com/ubergesundheit/Leaflet.EdgeMarker>
- [29] Esri leaflet. [Online]. Available: <https://esri.github.io/esri-leaflet/>
- [30] Serviços de computação na cloud — microsoft azure. [Online]. Available: <https://azure.microsoft.com/pt-pt/>
- [31] Email delivery service. [Online]. Available: <https://sendgrid.com/>
- [32] What is usability testing? (and what it isn't) — hotjar. [Online]. Available: <https://www.hotjar.com/usability-testing/>

## APPENDIX A

### Usability Test

No âmbito da Dissertação que estou atualmente a realizar, gostaria que efetuasse um teste de usabilidade ao protótipo, e que no final comentasse sobre a experiência. Obrigado!

- (1) Começando o teste, aceda ao website carregando neste link:  
<http://radarexchangeapp.azurewebsites.net/>
- (2) Encontra-se neste momento na página de acolhimento do RadarExchange. Tente compreender o intuito do produto navegando um pouco pela página.
- (3) Agora que tem informação inicial sobre a plataforma, vamos experimentar as funcionalidades. Para tal, criará uma conta. Após inserir os dados, um email ser-lhe-á enviado com o código de confirmação que deverá de seguida inserir na página para a qual será encaminhado. Após ter efetuado com sucesso este processo, pode efetuar o login.
- (4) Após efetuar o login, irá abrir-se a plataforma do RadarExchange no separador Search, ao qual voltaremos depois. Primeiro vá ao separador Profile. Como pode ver, este separador contém a sua informação pessoal, bem como os seus favoritos, subscrições e a opção de Logout.
- (5) Através do menu lateral poderá navegar pelas opções disponíveis e, por agora, altere o seu nome e selecione uma ou mais localizações onde pretende trocar os seus produtos. Verifique sempre que as alterações foram guardadas.
- (6) Depois de concluir as mudanças no Profile, mude agora para o separador MyStore. Crie um novo anúncio para oferecer um produto no Inventory. Preencha todos os campos do formulário devidamente. Ao guardar o produto, deverá poder vê-lo de imediato criado.
- (7) Imaginemos agora que você trocou o seu produto com alguém. Depois de alterar o estado do seu produto para indisponível, adicione uma nova troca no seu registo em Registry de atividade indicando o devido produto e o utilizador com quem trocou (sfnio@iscte-iul.pt).
- (8) Tendo concluído o anúncio do produto, vá agora à procura de produtos e serviços no separador Search. Neste separador poderá ver todos os produtos e serviços que estão a ser oferecidos ou requisitados por alguém. Deve agora procurar por um produto em específico: “Telemóvel LG L Fino”. Para tal, deve usar os filtros fornecidos e o modo de visualização que preferir.
- (9) Após ter conseguido encontrar o produto em questão, efetue o Request do produto. Neste processo, é lhe mostrado uma mensagem de aviso que o produto foi requisitado. Entre na página do utilizador que está a oferecer o produto e passe

a segui-lo. Verifique que uma conversa sobre o produto foi criada no separador Messages e que o utilizador se encontra na sua lista de Subs.

(10) Por último, faça logout.



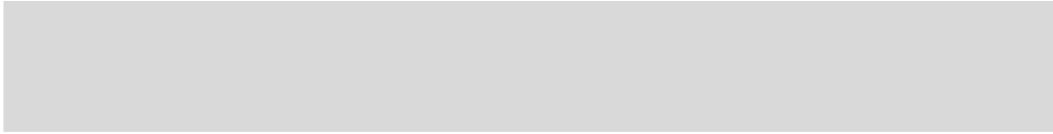
APPENDIX B

Feedback Quiz

		Discordo fortemente				Concordo fortemente	
		1	2	3	4	5	NA
1	No geral, estou satisfeito/a com a facilidade de uso desta plataforma.						
2	Consegui completar as tarefas rapidamente usando esta plataforma.						
3	Foi fácil aprender a usar esta plataforma.						
4	Foi fácil encontrar as informações de que precisei.						
5	As informações exibidas na plataforma são claras e fáceis de entender.						
6	Gosto de usar a interface desta plataforma.						
7	O Posicionamento geográfico dos utilizadores e dos produtos/serviços ajudam-me a ter uma melhor perceção dos produtos que me rodeiam						
8	Acho que com a indicação do meu posicionamento conseguirei encontrar produtos perto de mim mais facilmente						
9	A classificação dos utilizadores relativamente às trocas feitas, ajuda-me a ter uma melhor perceção sobre os mesmos e qualidade dos seus produtos e serviços						

FIGURE 48. Feedback Quiz

**Existe alguma informação que gostaria de ter visto e que não tenha sido apresentada neste teste?**



**Tem alguma sugestão ou comentário quanto à apresentação da informação e sobre o design do protótipo?**



FIGURE 49. Feedback Quiz

## APPENDIX C

### Responsivity

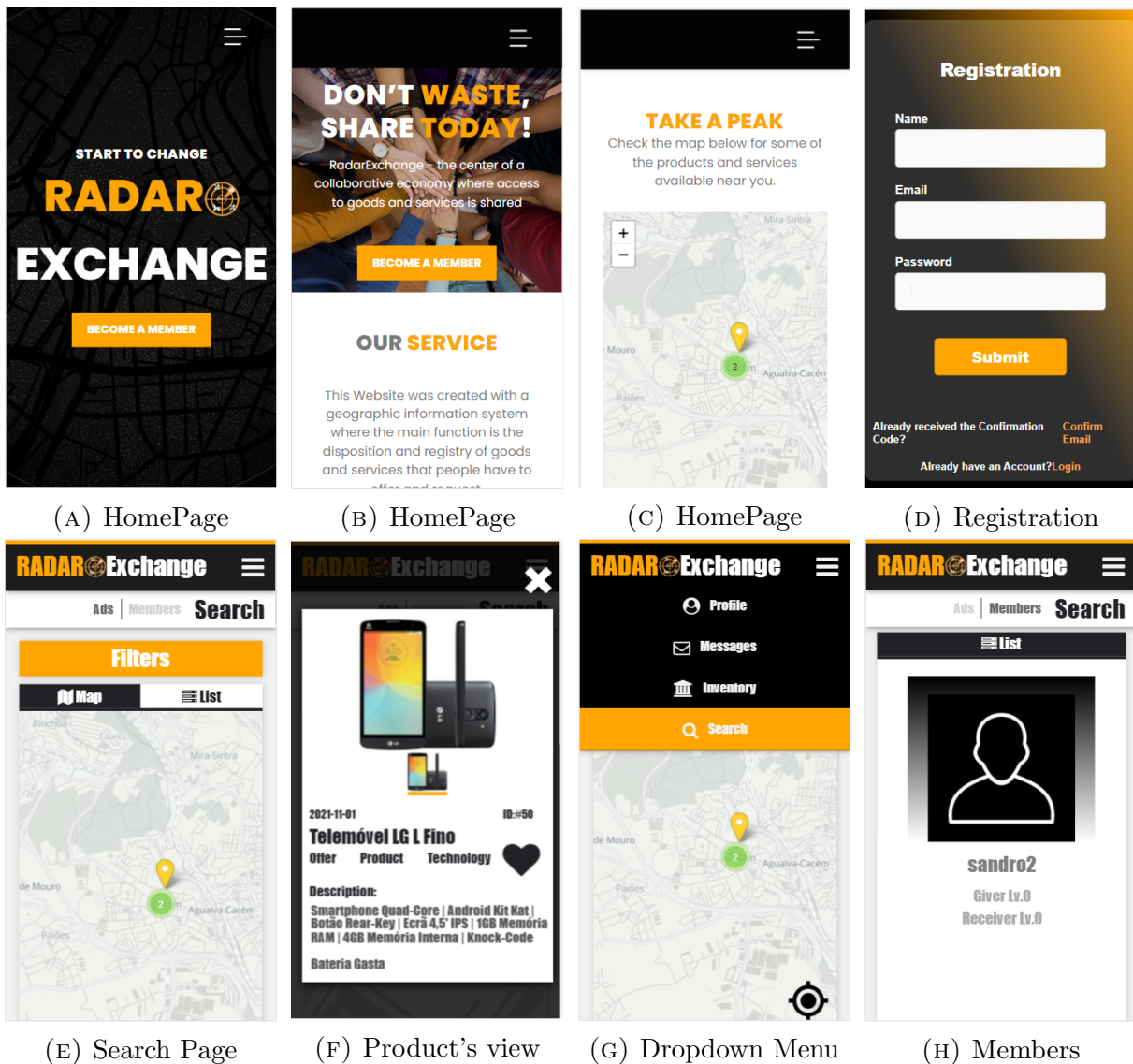
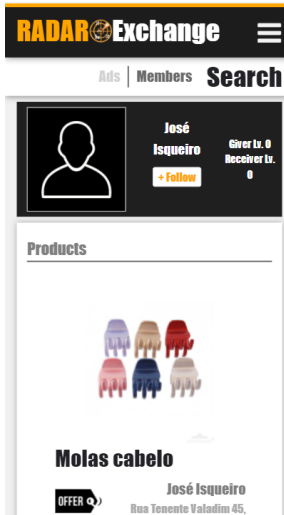
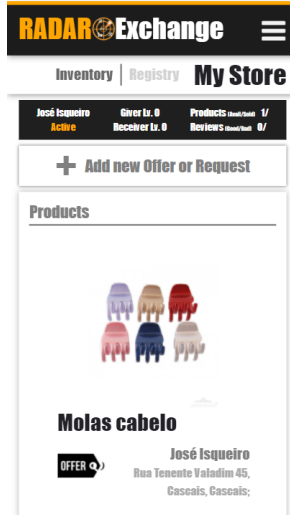


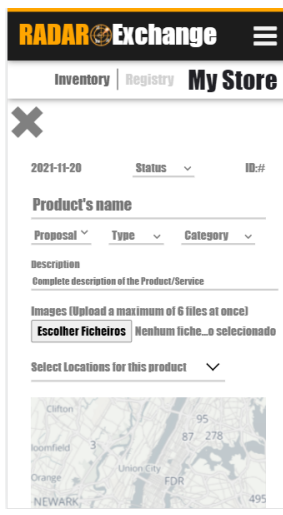
FIGURE 50. Website's Interface for smartphone



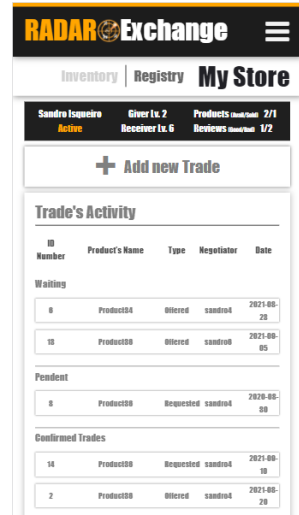
(A) Member's page



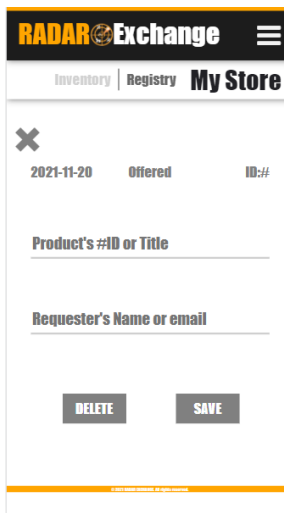
(B) Inventory List



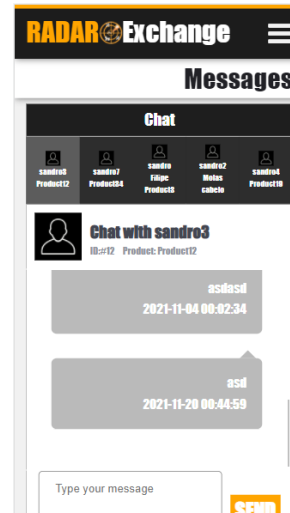
(C) Inventory form



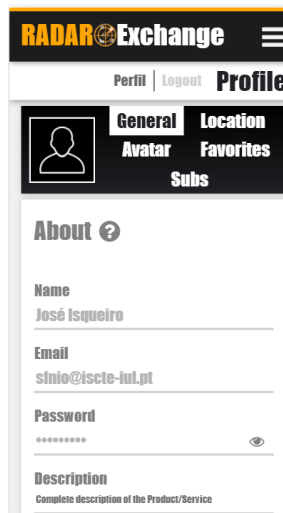
(D) Registry list



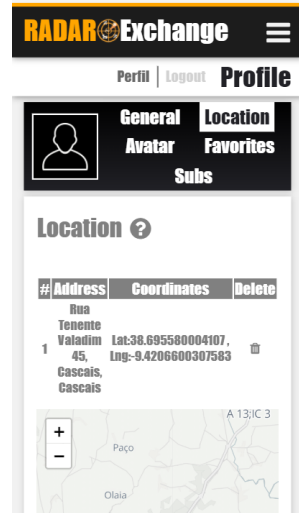
(E) Registry form



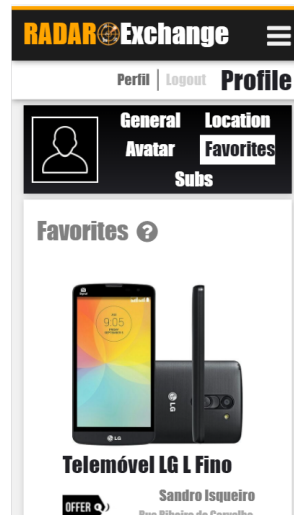
(F) Message Page



(G) Profile - General



(H) Profile - Locations



(I) Profile - Favorites

FIGURE 51. Website's Interface for smartphone