Universidade da Beira Interior
Departamento de Informática

Nº 245 - 2022: *APPSClone*

Made by:

Duarte Arribas

Advisor:

Carlos Barrico, PhD

Co-Advisor:

Rui Fernandes, PhD

13th April, 2022

Contents

Contents i

List of Figures v

List of Tables vii

1 Introduction 1
- 1.1 Scope . . . 1
- 1.2 Motivation . . . 2
  - 1.2.1 Global Navigation Satellite System (GNSS) . . . 2
    - 1.2.1.1 Positioning . . . 4
    - 1.2.1.2 Precise positioning - error correction techniques 5
    - 1.2.1.3 GNSS post processing . . . 8
    - 1.2.1.4 How GNSS exists in day-to-day lives . . . 8
    - 1.2.1.5 GNSSs around the globe . . . 9
  - 1.2.2 Reference frames . . . 9
- 1.3 Problem and objectives . . . 9
- 1.4 Approach . . . 10
- 1.5 Report outline . . . 10

2 State of Art 13
- 2.1 Introduction . . . 13
- 2.2 GDGPS . . . 13
- 2.3 APPS . . . 14
  - 2.3.1 File Uploads . . . 15
  - 2.3.2 Processing Modes . . . 15
  - 2.3.3 Output . . . 15
  - 2.3.4 Precise positioning in APPS . . . 16
- 2.4 Conclusion . . . 16

3 Software Engineering 17
- 3.1 Introduction . . . 17
- 3.2 Pre-Specification . . . 17
  - 3.2.1 Product Vision . . . 17
  - 3.2.2 Personas . . . 18

3.2.3 Scenarios . . . 18
3.2.3.1 Scenario 1: . . . 19
3.2.3.2 Scenario 2: . . . 19
3.2.3.3 Scenario 3: . . . 20
3.2.4 User Stories . . . 20
3.3 Specification . . . 20
3.3.1 Functional Requirements . . . 20
3.3.2 Non-Functional Requirements . . . 23
3.4 System Architecture . . . 23
3.4.1 Layered Diagram . . . 25
3.4.2 Pages Diagram . . . 25
3.4.3 Database Modelling . . . 25
3.4.3.1 Conceptual Model . . . 25
3.4.3.2 Logic Model . . . 25
3.4.3.3 Normalization . . . 25
3.5 Conclusion . . . 25

4 Technologies 27
4.1 Introduction . . . 27
4.2 APPS Python API . . . 27
4.2.1 The APPS class . . . 28
4.2.1.1 Check if everything is working . . . 28
4.2.1.2 Upload a RINEX file . . . 29
4.3 Conclusion . . . 30

5 Design 31
5.1 Introduction . . . 31
5.2 Corporate Image . . . 31
5.2.1 Color . . . 31
5.2.2 Logo . . . 31
5.3 Interface Design . . . 31
5.3.1 Wireframes . . . 31
5.3.2 Mockups . . . 31
5.4 Conclusion . . . 32

6 Development 33
6.1 Introduction . . . 33
6.2 Development . . . 33
6.3 Conclusion . . . 33

7 Tests 35

7.1 Introduction 35
7.2 Functional tests 35
7.2.1 Unit tests 35
7.2.2 Feature tests 35
7.2.3 System tests 35
7.3 User tests 35
7.3.1 Work tests 35
7.3.2 UI tests 36
7.3.3 UX tests 36
7.4 Performance and Load tests 36
7.5 Security tests 36
7.6 Conclusion 36

8 Conclusion **37**
8.1 Conclusion 37
8.2 Future work 37

Bibliography **39**

A Models **43**
A.1 Sequence models 43

List of Figures

1.1 How the three GNSS segments relate to each other[7] 3
1.2 One satellite's location and its distance to the receiver 5
1.3 Two satellite's location and their distance to the receiver 6
1.4 Three satellite's location and their distance to the receiver. The three circles converge to one point-the location of the receiver . . . 6
1.5 Four satellite's used for trilateration around a location on earth. The first satellite locates the receiver somewhere on the **sphere** (top left of the figure); the second satellite locates the receiver somewhere on the intersection **circle** (top right of the figure); the third satellite locates the receiver on **two possible points** (bottom left of the figure); the fourth satellite locates the **exact position** of the receiver (bottom right of the figure) [9] 7
1.6 Sprint planning on Trello: the board on the left represents the sprints, and the board on the right represents a specific sprint 11

3.1 The persona Maria 18
3.2 The persona Stefan 19

A.1 Sequence diagram for the password recovering system 44

List of Tables

1.1 Details of the four main GNSSs in the world 9

3.1 Functional requirements of the APPSClone 23

Code samples list

4.1 APPS API access settings file 27
4.2 Install the APPS API through pip 28
4.3 Instantiating the APPS class 28
4.4 The APPS class 28
4.5 The `ping` method 29
4.6 The `upload_gipsy` method 29

Acknowledgments

To be written last.

Abstract

To be written last.

Keywords

GNSS, GPS, positioning, precise positioning, rtk, ppp, post-processing.

Acronyms

API	Application programming interface
APPS	Automatic Precise Positioning Service
DB	Database
GDGPS	Global Differential Global Positioning System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical user interface
HCI	Human-computer interaction
HTTP	Hypertext Transfer Protocol
ISP	Internet service provider
JPL	Jet Propulsion Laboratory
NASA	National Aeronautics and Space Administration
NRT	Near Real Time
OOP	Object-oriented programming
OTP	One-time password
PNT	positioning, navigation and timing
PPP	Precise point positioning
PT	Propagation time
RINEX	Receiver Independent Exchange format
RTK	Real-time kinematic
UI	User interface
UX	User experience
VC	Version control

Style Guide

In order to provide a rich and evident reading experience, this text attends to a particular style. It is based on my view of how the content will be the most comprehensible and easy to read.

The following are the typographic and usage conventions that are used in this report:

Bold	Used for simple emphasis, User interface (UI) elements and important elements;
Italic	Used for loanwords, the first introduction of a concept and chapter's abstracts;
`Monospaced`	Used for inline code, filenames, directory names, host names, commands and others;
Blue	Used for hyperlinks;
$Italic\ \ spaced$	Used for inline mathematical equations;
"Quotation Marks"	Used for odd, nonstandard use of words.

All images, tables, long equations and code blocks will have captions; if the previous are too long, they'll be placed in appendices.

This report won't use the Oxford Comma[1] when enumerating lists.

Chapter

1

Introduction

This chapter presents a brief introduction to the project and justifies how and why it was developed. Includes a foundation of the main concepts that glean on GNSS and precise positioning and introduces the initial problems, congregating possible solutions, goals and approaches.

1.1 Scope

This report is the foundation of the **APPSClone project**, intending as both a guide to how the project was implemented and as a linear follow-up to understanding the work done.

Cloning NASA is the title of the chosen project. It foresees a study of GNSSs and the APPS Application programming interface (API), which will result in a clone-based application that supports features related to the original interface.

The curricular unit of *Projeto* requires the merging of distinct study areas and technologies with new learning experiences. The following are the areas of study that the project will focus on:

- **Programming and web programming**: Integral to the whole project, as it'll be the basis of the front and backend of the web application;
- **Math**: Needed to understand basic GNSS concepts and basic programming;
- **Report writing**: Imperative for the proper writing of this report;
- **Version control (VC)**: Used to manage and backup versions of the code;
- **Clean code**: All code written in this project must follow correct programming practices to produce clean and concise code[2];

- **Documentation**: Documentation is imperative for the correct understanding of the code;
- **Networking**: Needed for website management and deployment;
- **Imperative, Functional and Object-oriented programming (OOP)**: The code of the app will apply the three paradigms;
- **Databases**: The web app will implement a database;
- **Multimedia and Human-computer interaction (HCI)**: Basis for the design of the Graphical user interface (GUI) and the exceptional User experience (UX);
- **Algorithms, data structures and automata theory**: Needed for some programming challenges of the web app;
- **Operating systems and System Administration**: Imperative for the deployment of the web app;
- **Software engineering**: Crucial for the planning of the whole project and their, consequent, testing;
- **Cybersecurity**: Used to assure the security of the web app;
- **GNSS**: The main topic of the project

The needed technologies will be discussed late,r in chapter 4, which starts on page 27.

1.2 Motivation

Space agencies have undergone considerable advancements in Geo-spatial location, to the point that some individuals' social lives, jobs, entertainment now depend on it. It is a vast subject with lots of areas of study, which are abstracted to the regular user by a simple user interface on a smartphone or computer.

1.2.1 Global Navigation Satellite System (GNSS)

A *satellite navigation system* is a system that uses a network of satellites to provide Geo-spatial positioning, navigation and timing (PNT)[3], by allowing receivers to determine their location[4] (longitude, latitude and altitude) using

the timing and orbital information transmitted by the satellites and received by the antennas[5].

A satellite navigation system with global coverage is called a *GNSS*. GNSSs are made of three main components, called *segments*[6]:

- *Space segment*: consists of an array of GNSS satellites orbiting about 20,000 km above the earth. Each satellite broadcasts a signal that provides its identity, time, orbit (location) and status;
- *Control segment*: consists of an array of ground-based master control stations, data uploading stations and monitor stations. The monitor stations monitor the satellites signals and status, relaying that information to the master control stations, which make corrections to the orbit of the satellites, so that they can be send to them by the data uploading stations;
- *User segment*: consists of equipment that processes the received signals from the GNSS satellites and use them to derive location and timing information.

Figure 1.1 represents the interaction of the three GNSS segments. The icons represent each system that belongs to the specific segment.

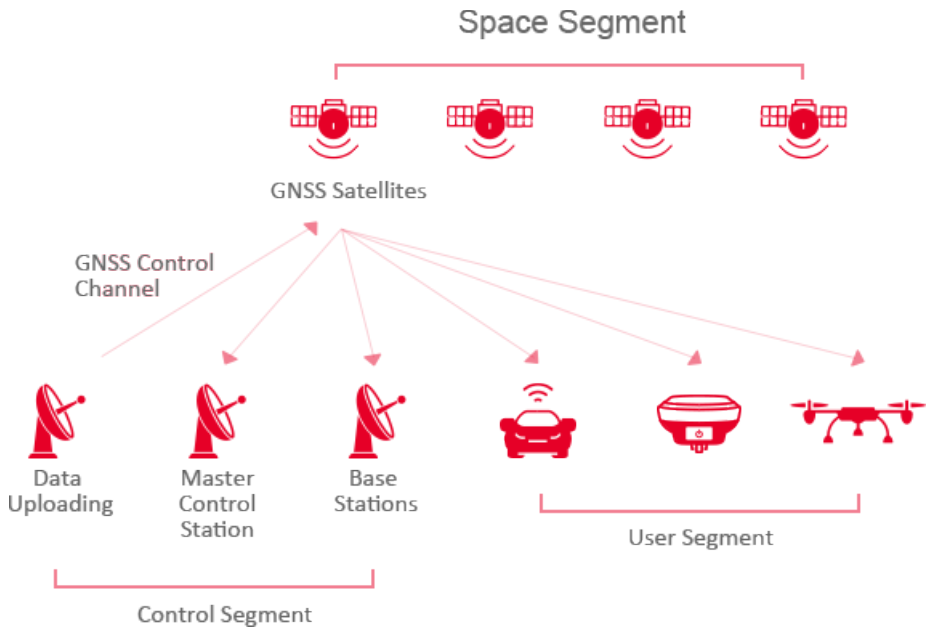

Figure 1.1: How the three GNSS segments relate to each other[7]

1.2.1.1 Positioning

GNSS satellites contain very accurate clocks, which are very important, in that they are used to determine how much time it takes a GNSS signal to travel from satellites to receivers, the Propagation time (PT). That is very important, as the distance between the receiver and the satellite can be calculated by multiplying the PT by the speed of light (c), as shown in the equation 1.1[8].

$$\text{distance} = PT * c \tag{1.1}$$

The time PT can be calculated as long as both the time the signal left the satellite (t_0) and the time the signal reached the receiver (t_1) are known. The equation 1.1 shows how to calculate the PT[8].

$$PT = t_1 - t_0 \tag{1.2}$$

The receiver will receive the distances and orbit of, usually, four satellites, so that it can determine its location using a method called *trilateration*.

Trilateration works by reducing the possible amount of points the receiver can be in. In two dimensions, knowing the location of three satellites and the distance between them and the receiver, it's possible to conclude the position the receiver is in. Figure 1.2 displays one satellite.

The receiver now knows it's located at one point on that circle, but at which point? There's no way to tell exactly where the receiver is located at with only one satellite, but what if two are used? Figure 1.3 explores exactly that.

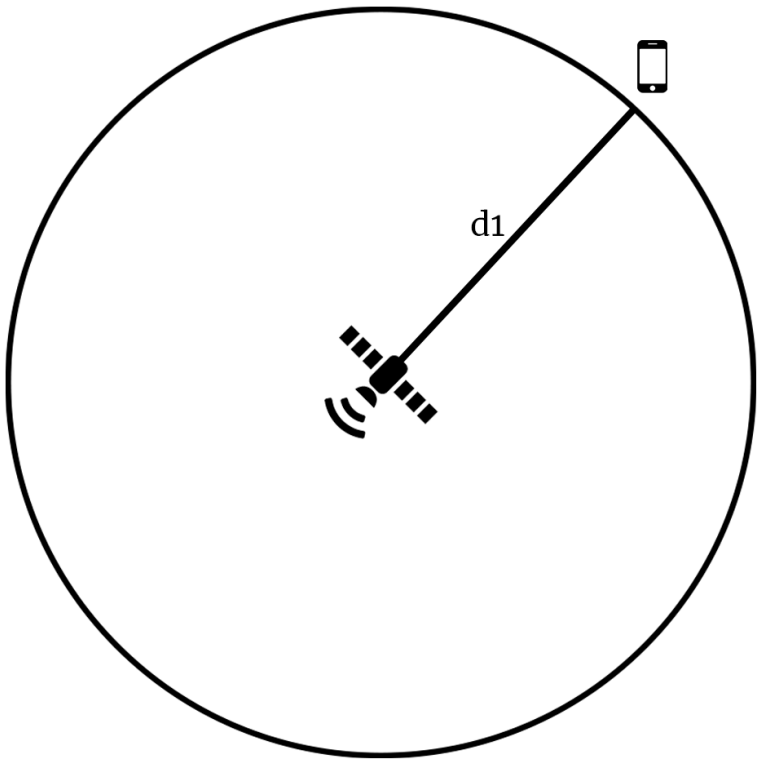

Figure 1.2: One satellite's location and its distance to the receiver

Now, the location of the receiver is immensely more accurate; there are only two possible locations for the receiver. Unfortunately, there's no way to reduce it to one location without adding a third satellite, as shown in figure 1.4.

Because receivers don't have very accurate clocks, there will be an error when calculating the location of the receiver; to correct this error, the receiver will advance or delay its clock until all of the ranges converge in one point.

In three dimensions, the ranges are shown as spheres and not circles. Because of that, a fourth satellite will be needed to find the location of the receiver. Figure 1.5 shows how trilateration in three dimensions needs four satellites.

1.2.1.2 Precise positioning - error correction techniques

Even though the errors from the **not so** very accurate clocks from the receiver were corrected, there are still other errors that may occur, which can be di-

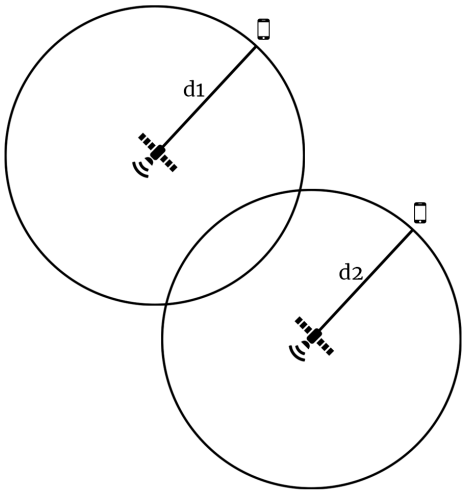

Figure 1.3: Two satellite's location and their distance to the receiver

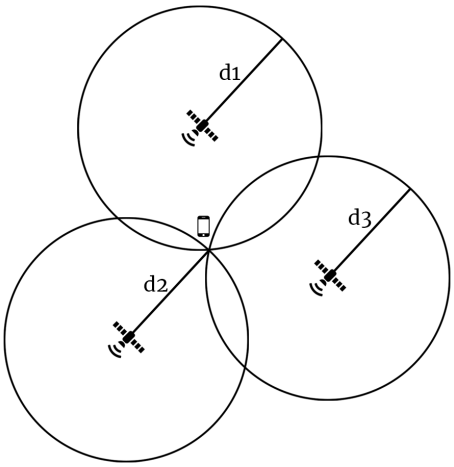

Figure 1.4: Three satellite's location and their distance to the receiver. The three circles converge to one point-the location of the receiver

vided in two categories[8]:

- **Natural cause errors**: errors that were not introduced on purpose. E.G.: inaccuracy due to the refraction of the satellite signal, in the process of passing through the ionosphere and troposphere;
- **Man-made errors**: errors that were introduced on purpose. E.G.: government's Selective Availability methods.

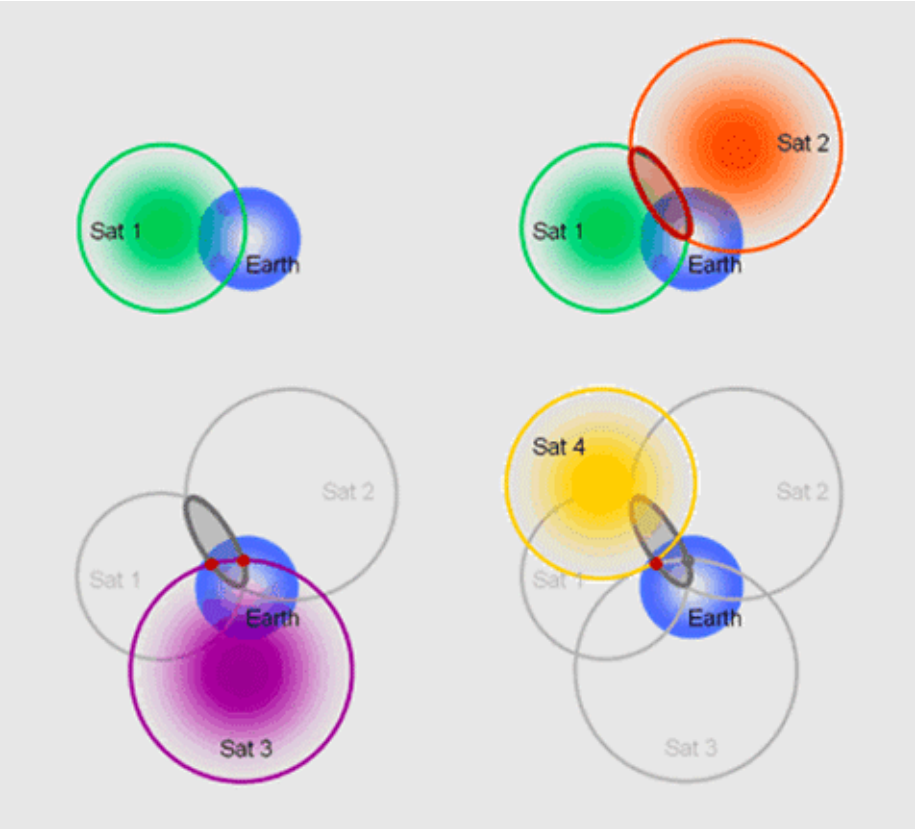

Figure 1.5: Four satellite's used for trilateration around a location on earth. The first satellite locates the receiver somewhere on the **sphere** (top left of the figure); the second satellite locates the receiver somewhere on the intersection **circle** (top right of the figure); the third satellite locates the receiver on **two possible points** (bottom left of the figure); the fourth satellite locates the **exact position** of the receiver (bottom right of the figure) [9]

Precise positioning is the process of mitigating these errors in order calculate the exact location of a receiver as accurately as possible. To achieve this, several *positioning techniques* were developed.

Real-time kinematic (RTK) is a positioning technique that removes most systematic errors, such as satellite clock and orbit errors, troposphere and ionosphere effects and others.

Here, a base station with known coordinates is needed. It'll send its coordinates and satellite observations to estimate its position with a high degree of accuracy (2-3cm).

Precise point positioning (PPP) is a positioning technique that removes or minimized GNSS system errors using, unlike RTK, only one receiver.

It charges a fee to mitigate and correct the errors, estimating its position with a high degree of accuracy (usually a few decimeters, but can go up to 3 cm of accuracy).

1.2.1.3 GNSS post processing

Some positioning methods, such as those referred in subsection 1.2.1.2 require some sort of post processing tools and algorithms to mitigate and correct their errors[10].

Tools process the data and make corrections to it. This data is most often represented in Receiver Independent Exchange format (RINEX) files, which contain raw satellite navigation system data relative to a specified interval of time. The standard format for the name of these files is `ssssdddh.yyt`, where:

- **ssss** is the marker name of the ground station where the recording was performed;
- **ddd** is a three digits number indicating the day of the year when the recording started;
- **h** is a letter indicating the hour of the day when the recording started;
- **yy** is a two digits number indicating the year when the recording started;
- **t** is a letter indicating the type of the RINEX file. Most common types are:
  - **o**: observation data files, which contain satellites' position data;
  - **d**: compressed observation data files;
  - **n**: navigation data files, which contain Global Positioning System (GPS) ephemeris data;
  - **g**: navigation data files, which contain GLONASS ephemeris data;
  - **l**: navigation data files, which contain GALILEO ephemeris data;
  - **m**: meteorological data files.

1.2.1.4 How GNSS exists in day-to-day lives

GNSS is vastly used in consumer-grade products. GNSS receivers are routinely integrated into smartphones, web services and more, to support applications that display maps and calculate best routes, traffic and others. Examples of products that take advantage of GNSS are:

- Google maps;
- Apple maps;
- Car GPS navigators.

1.2.1.5 GNSSs around the globe

At the time of writing, there are four GNSSs in the world[11] owned by different political organizations:

- GPS;
- GLONASS;
- Galileo;
- BeiDou.

Table 1.1 displays who owns each GNSS, the year each started their operation and how many satellites it currently has in orbit.

GNSS name	Location	Operation start	Satellites in orbit
GPS	USA	1978	31
GLONASS	Russia	1993	27
Galileo	EU	2016	22
BeiDou	China	2000	22

Table 1.1: Details of the four main GNSSs in the world

1.2.2 Reference frames

Reference frames are abstract coordinate systems with an origin, orientation and scale specified by set of reference points, i.e., geometric points whose position is identified both mathematically and physically.

1.3 Problem and objectives

Many applications post-process GNSS information for precise positioning. One of these applications' firewall (described in section 2 on page 13) has been blocking Internet service provider (ISP)s all across the world, mainly generic ISPs, such as Vodafone or MEO in Portugal.

The goal of this project is to build a web application to provide GNSS data upload and post-processing, to achieve precise positioning information. All the frontend and backend of the website are to be built from source, allowing for a new interface to be used with the existing content.

1.4 Approach

This project was carefully planned using a variation of scrum[12], designed for use within this project.

The outline planning phase started on the 3rd March, 2022, where the project was split up into seven sprints:

1. **Pre-Specification sprint**: the introduction chapter (chapter 1) and the State of Art chapter (chapter 2) are written;
2. **Specification sprint**: the requirements part of the software engineering chapter (chapter 3) is written;
3. **System architecture**: the system architecture part of the software engineering chapter (chapter 3) and the technologies chapter (chapter 4) are written. Technologies have to be learned here;
4. **Design**: the design chapter (chapter 5) is written;
5. **Implementation**: the implementation chapter (chapter6) is written;
6. **V&V**: the tests chapter (chapter 7) is written;
7. **Deployment**: the project and report are reviewed and deployed.

The sprints are planned on Trello, as shown in figure 1.6.

1.5 Report outline

This document was carefully designed to be read sequentially; therefore, every chapter is minutely written with clarity and follow-up in mind.

The report is, therefore, divided into four non-delimited parts:

- **Document organization minutiae**: explains how the document is structured and helps the reader understand how to read the document;

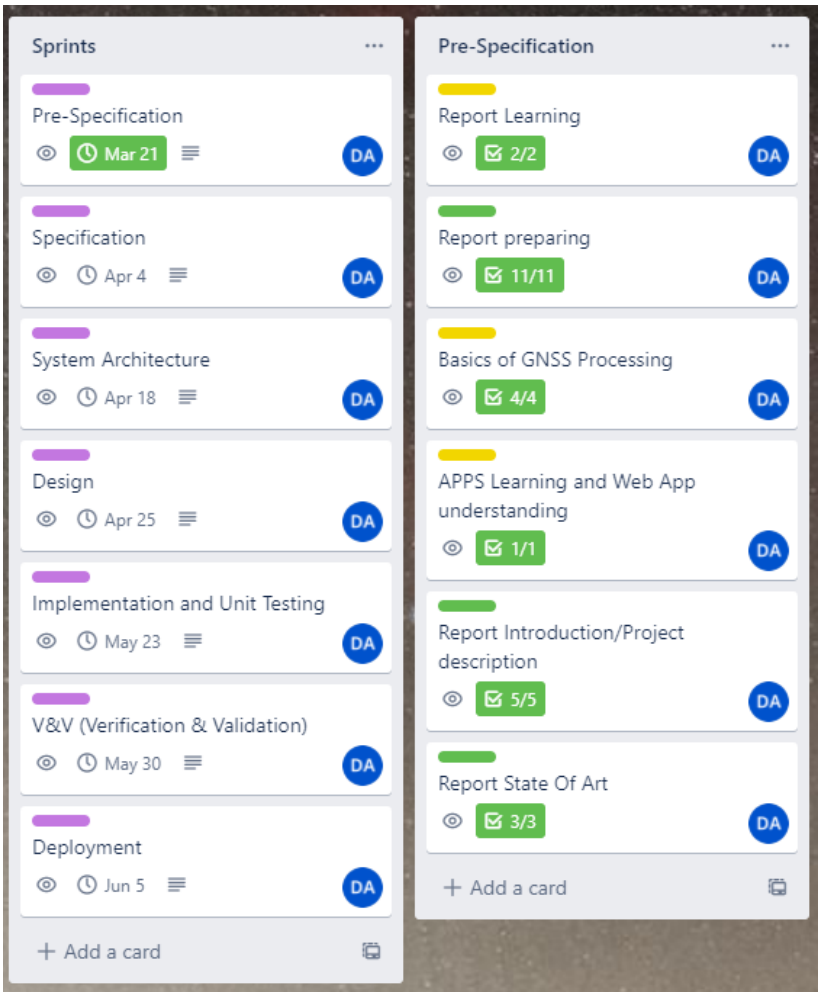

Figure 1.6: Sprint planning on Trello: the board on the left represents the sprints, and the board on the right represents a specific sprint

- **Work identification**: presents how the work was organized, carefully planned and what methods were used to arrange a smooth work environment;
- **Learning experience**: offers an insight at what was learned during the whole project;
- **Project**: outlines the project itself and how it was made.

Notwithstanding the previous list, this text was thoroughly grouped into eight different chapters. Each chapter contains content that is related to each other

and, thus, must be unified. The following list delineates a summary of the eight chapters:

Chapter 1 - Introduction Motivates the reader for the usage of GNSS and how the correction of errors introduces precise positioning of GNSS receivers. Also, explains how the idea of the project has surged and how it integrates with the whole course;

Chapter 2 - State of Art Explores the cutting edge technology that already exists for GNSS post-processing and the associated problems with it;

Chapter 3 - Software Engineering Covers the planning process for the project. Includes the pre-specification, which comprises the product vision, personas, scenarios and user stories, the specification, which comprises, functional requirements, non-functional requirements and their specification alongside diagram models and system architecture, which comprises a layered and pages diagram and goes into detail how the database for the project is modeled;

Chapter 4 - Technologies Presents the used technologies, how they work and the process that's going to be used to apply them to the project;

Chapter 5 - Design Introduces the project to a graphical level. Comprises the corporate image (including logos and colors) and the interface design of the website (in which wireframes and mockups will be used to draft the implementation);

Chapter 6 - Development something;

Chapter 7 - Tests Uses functional, user, performance and loading and security tests to justify the correct functioning of the software;

Chapter 8 - Conclusion Presents in a SWOT fashion what the project accomplished and what it didn't. Gives room for future work to be built upon this one.

Chapter

2

State of Art

This chapter presents the state of the art products and how they are already being used for GNSS post processing. Presents an introduction to the NASA GDGPS tools, such as the APPS tool, which the project will focus on.

2.1 Introduction

Plenty of raw GNSS satellite measurement data is collected and stored (see reference 1.2.1.3) as they are not required in real-time[13]. As a matter of fact, some precise positioning techniques used in the post-processing of this information may take a considerable amount of time to reach the desired amount of accuracy[14]:

> Static PPP needs 60 min to reach five-centimeter horizontal accuracy at the 95% confidence level and 24h to achieve one-centimeter horizontal accuracy at the 95% confidence level.

There are several online tools for GNSS post-processing[15] and all work in different ways. This chapter will focus on one of this tools and will explain how it's used and how its API can be operated and re purposed for a new app.

2.2 GDGPS

The Global Differential Global Positioning System (GDGPS) is a complete, highly accurate and extremely robust real-time GNSS monitoring and augmentation system[16]. As mentioned in section 1.2.1.2 on page 5, this system provides several products and services for error correction and, subsequently, accurate position estimation.

GDGPS was developed at the Jet Propulsion Laboratory (JPL) by National Aeronautics and Space Administration (NASA) in support to the latter's demanding terrestrial, airborne, and spaceborne operations[16]. The system has far-reaching consequences in accurate and timely assessment of the magnitude and structure of earthquakes, as well as the magnitude and direction of resulting tsunamis[17].

Employing large number of reference receivers and innovative network architecture, it uses real-time data post-processing software for specific tasks, such as the previous concerned precise positioning. Among the vast array of services and products it provides[16]:

1. Data management for High performance GPS tracking networks;
2. Precise real-time positioning and orbit determination;
3. GPS performance monitoring and situational assessment;
4. Real-time environmental monitoring;
5. Commercial differential systems;
6. Assisted GPS data services.

2.3 APPS

The Automatic Precise Positioning Service (APPS) is a precise positioning GNSS post-processing tool (encompassing the service of item 2 of section 2.2), which accepts GPS files to apply precise positioning technology to estimate the position of the related GPS receivers, whether they are static, in motion, on the ground or in the air; its main features are:[18]

- Static point positioning for static receivers;
- Kinematic point positioning for receivers in motion;
- Access to real-time GPS orbit and clock products;
- Fast post-processing results.

The following subsections will focus on analyze the tool, describing it and illustrate how it is used. The information that will be shown in the following subsections will be adapted from the official APPS website (see reference [18]).

2.3.1 File Uploads

The service supports RINEX 2, RINEX 2.11 and RINEX 3 format files for the uploading of the GNSS data; these files may have arbitrary filenames, with measurement rates up to one second.

These files can be uploaded via three means:

- Manually (through the website);
- Using the python API;
- Using APPS as a web service via Hypertext Transfer Protocol (HTTP).

2.3.2 Processing Modes

The files can be processed in two different modes:

- **Static**: assumes that the GPS measurements were taken at a static antenna;
- **Kinematic**: assumes that the GPS measurements were taken at an antenna in motion.

They can also be processed with two different latency options:

- **Near Real Time (NRT)**: the results will be less accurate, but the processing time will be faster;
- **Most accurate**: the results will be the more accurate, but the processing time will be slower.

2.3.3 Output

After processed, information about the GNSS analysis and the precise positioning of the receiver will be described. The following are the files generated as output:

- XYZ file: includes time, (x,y,z) coordinates, together with their formal errors, using the ITRF2014 reference frames[19];
- GIPSY TDP file: includes time, (x,y,z) coordinates, using the GIPSY's TDP format.
- Summary file: includes run status (success/error), key output values and list of output;

- Access information: URL of output files and password;
- Instant positioning for web users, including (x,y,z) coordinates, Latitude, Longitude, Height.

2.3.4 Precise positioning in APPS

The APPS uses a PPP based technique (see subsection 1.2.1.2 on page 5) implemented in JPL's GIPSY-OASIS software for error correction, which estimates:

- Position coordinates;
- Receiver clock states as white noise with updates every measurement epoch;
- Zenith wet delay as a random walk with variance of $3mm2$ per hour;
- Wet delay gradient as a random walk with variance of $0.3mm2$ per hour;
- Phase ambiguities as real numbers.

2.4 Conclusion

As mentioned in section 1.3 of chapter 1, there exist several tools for GNSS post processing and precise positioning. One of those tools is the APPS tool; this tool's firewall, however, is blocked my many ISPs, hence the need for a new interface, that uses its API.

The already existing interface is meant to be upgraded, while maintaining its core features and structure, so that existing users can browser the application without any prior training.

Chapter

3

Software Engineering

This chapter describes the agile process carried out during the development of the product (APPSClone) in this project. It'll explain how it was conceived and planned before its actual implementation. It is divided into three main sections: the pre-specification section explains the user view of the product, the specification explains the developer view of the product and the system architecture section will formulate how the different parts of the system will integrate together; these last two sections will be used in the development of the system.

3.1 Introduction

A well developed system must be planned in advance first. Because this system is planned in an agile process, this chapter will derive the culmination of concludes a work altered

3.2 Pre-Specification

It is very important to understand what the users want from the product that is being built. It's also imperative to understand how the users will use the system and how it can be better assembled to accord to the users' needs and to have a better user experience.

3.2.1 Product Vision

A generic vision on the product that is being built is a solid foundation to the whole project. The following product vision follows the Moore's product vision template.

FOR generic ISP users, **WHO** need accurate location estimation of GPS receivers, the **APPSClone** is a **WEB-BASED SERVICE**, **THAT** accepts GPS measurement files and applies the most advanced GPS positioning technology from NASA's JPL to estimate the position of the GPS receivers. **UNLIKE** the original APPS, **OUR PRODUCT** aims at a more user friendly and easy to use interface, which won't block generic ISPs from connecting to it.

3.2.2 Personas

The personas will introduce the target audience that will be using APPSClone. Figures 3.1 and 3.2 show the two personas of the APPSClone.

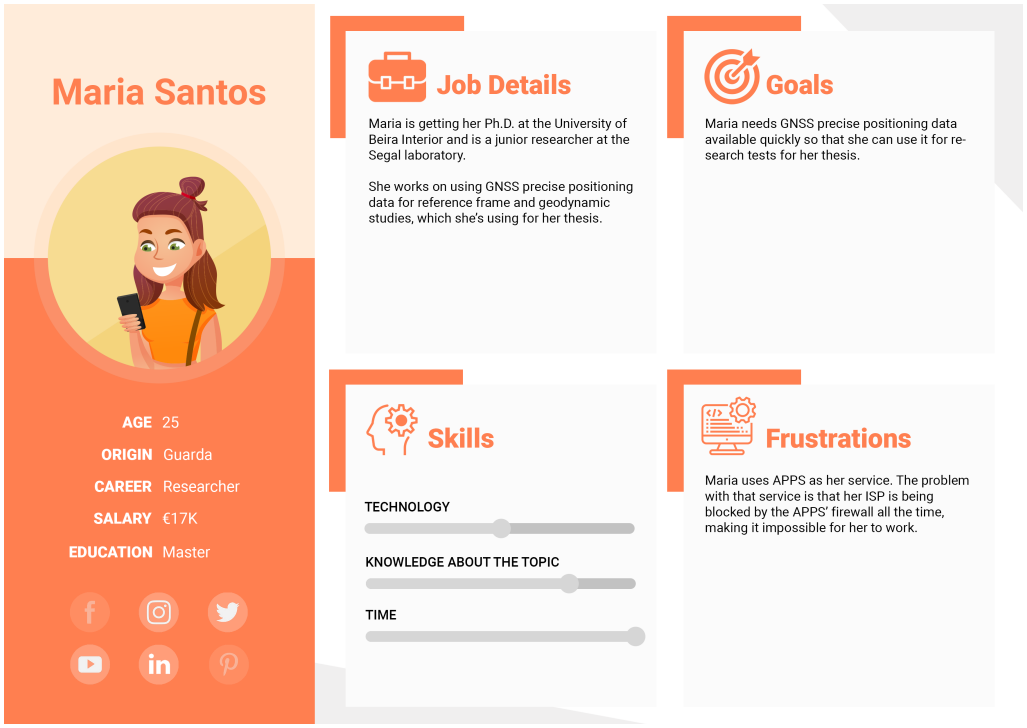

Figure 3.1: The persona Maria

3.2.3 Scenarios

Scenarios are specific use-cases that personas (see subsection 3.2.2) will follow to access a certain feature of the website. They're not meant as complete definitions and will evolve into the user stories, defined in the subsection 3.2.4.

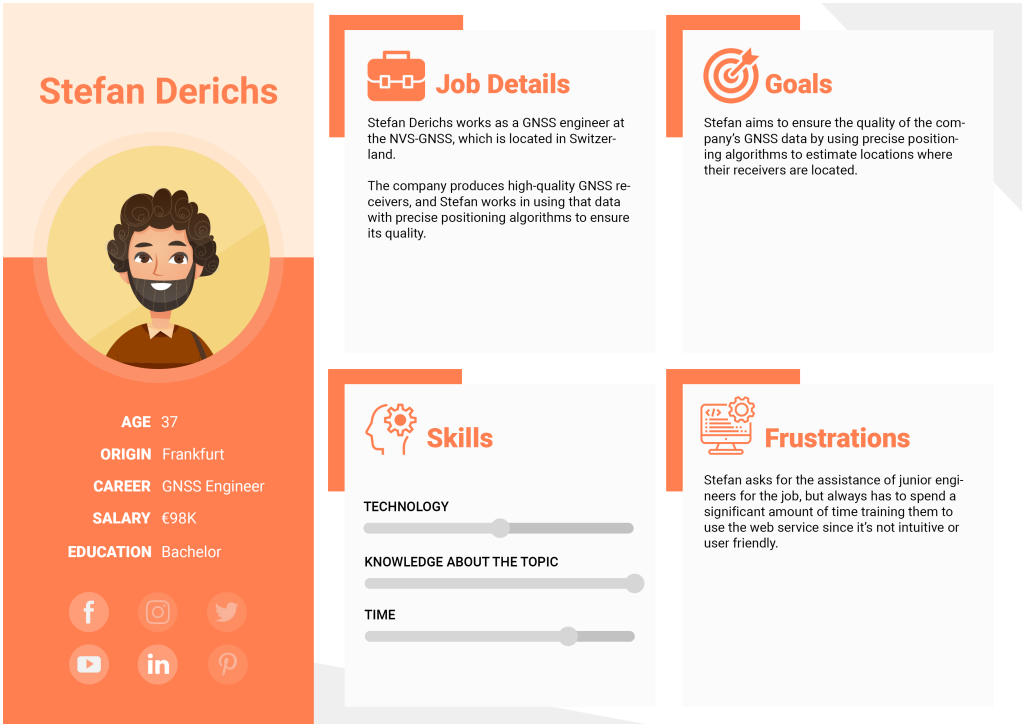

Figure 3.2: The persona Stefan

3.2.3.1 Scenario 1:

Maria needs the precise positioning of some data she got from GPS receivers. Previously, she would have to send an email for the APPS team to be able to access the blocked website (see figure 3.1). Maria didn't consider the process of having to write the email acceptable at best, however the time she had to wait for the website to be unblocked was dreadful and really slowed down her work.

Because of that, she now uses APPClone to upload her RINEX files and, after choosing the appropriate settings, gets the precise location of the receiver that generated those files. That location is, then, presented on the map for easy observation.

3.2.3.2 Scenario 2:

Stefan is training junior engineers to work with the APPS. The interface is not very user friendly and unintuitive, thus it takes a lot of time to train them.

With the APPSClone Stefan can now train new engineers faster and, thus, save money and time.

3.2.3.3 Scenario 3:

Maria needs to access the output files from the post-processing of the RINEX files. Previously, she had to download them. Now, she can either choose to download them or see them in the actual website.

3.2.4 User Stories

User stories check the priority of features the users may want from the website. They're meant as guides to create functional requirements and for the developer to understand what should be developed first. The following are the users stories of APPSClone:

1. As a GNSS researcher and user of the website, I want to get the exact location of where the GNSS receivers received the raw satellite information;

2. As a GNSS researcher and user of the website, I want to see the exact location of where the GNSS receivers received the raw satellite information on a map;

3. As a GNSS researcher trainer, I want junior researchers to understand how the web service works quicker, so they can start using it as a tool in their research, saving time and money to the company.

3.3 Specification

The developer should have a clear and unambiguous guide on how the product should be developed. Its specification aims to guide the development by displaying what it should or not have and how it connects its features.

3.3.1 Functional Requirements

Functional requirements explain everything that should be developed in the software. For each functional requirement, a rationale is added to explain why the feature is needed in the system. Some of these requirements may need extra explaining, which is covered by either sequence or activity models. Further, these requirements will be grouped into use cases.

Functional requirement	**Rationale**

1: The system shall have a login system, as an authenticated mechanism using an username/email and a password.	The system needs to save each set of PPP output files for each user.
2: The system should have a way to recover the password, using their email (see figure A.1 from appendix A for a model representation).	If the user forgets their password or the account is compromised, they should be able to recover it.
3: The system should allow the user to stay logged in after leaving the page.	Upon logging in, the user should have an option to stay logged in, even after the website is closed and reopened, so that they don't have to authenticate again. This is useful for users who don't share a computer and want faster access to the website's content.
4: The system shall have a registration system, where users can create their account, by giving some of their personal information (name, username, email, password, affiliation, address).	To be able to access some of the features of the web service (such as their saved PPP output files), the users need to be registered in the database, so that they can login to their account.
5: The system should only allow account creation after confirmation with their email (the process is similar to the system in requirement 3.1).	As a security measure, people should only be able to associate their emails to the web service, if they confirm they own the email.
6: The system should allow the login and register password to be shown and hidden.	If the user is unsure if they're typing the correct password, they can show it for a brief moment and, then, hide again.
7: The system should allow the user to authenticate with google.	The user may want to register and further log in with their google account to save time.

8: The system shall have a user profile for logged in users, where the user can change its profile picture, name, email, password, affiliation and address).	The user may want to change its credentials.
9: The system shall display information about the website and about its features.	The user may want to know more about the web service before actually starting to use it.
10: The system should have an interactive tutorial on how to use the website.	To facilitate user training, the website should provide a way of teaching the users to use the web service.
11: The system shall have a contact system without authentication.	If the users wish to contact with the owners of the website, they may do so, without having to log in.
12: The system shall accept the upload of RINEX files for future post-processing.	The files need to be processed and, thus, must be on the server.
13: The system shall associate uploaded RINEX files with the user who uploaded them.	The user should be able to access previous uploaded files.
14: The system shall allow the user to change the parameters of post-processing, namely, Job Name, Email Notify, Access, Use Case, Processing Mode, GNSS Orbit & Clock State, Troposphere Model, Ocean Loading, Model Tides, Elevation Dependant Weight, Elevation Angle Cutoff and Solution Period.	Some of the parameters of the post-processing should be able to be changed by the user.
15: The system shall process the RINEX file, upon approval of the parameters, producing the results, which can be seen and/or downloaded.	This is the main feature of the web service. It's meant to produce the needed results.
16: The system shall display a map location of the coordinates that result from the RINEX file post-processing.	The user may want to have a brief overview of where the receiver collected the satellite data.

17: The system shall allow the viewing, download and deletion of the previous submissions of the user that is logged in.	The user may want to view, download and remove previous submissions multiple times.
18: The system should provide all of its content in both the English and Portuguese languages.	The English language is an international language and the APPSClone aims to be used anywhere. The Portuguese language is used, because the project is made in a Portuguese environment and many people who may want to use it know Portuguese; also, the author also knows Portuguese, ensuring good grammar and orthography.
19: The system shall impose a quota of storage for submissions of each user.	In order to not overflow the server's storage, it's important to keep a quota of usage for each user, so that they understand that they should delete their submissions, if they present to not be important.

Table 3.1: Functional requirements of the APPSClone

3.3.2 Non-Functional Requirements

Non-Functional requirements aim to restrict the service to some accords. These restrictions often involve extra development to make the software run as it is supposed to.

3.4 System Architecture

Content

Non-Functional category		Non-Functional requ
Usability		The system shall not impose a learning/training time of more th
		The system shall provide support f
Performance		The maximum satisfactory response time to be experienced on
Space		The system shall not occupy mor
Dependability		The system shall be ava
		The system shall be available to every
Security	Secure protocols	The system shall use the \textbacksla
		The system shall use the \textbacks
		The system shall use the \textbackslash{
	Response headers	The system shall use the \textbackslash
		The system shall use the X-Content-Typ
		The system shall use the X-Frame-O
		The system shall use the X-XSS-Prot
		The system shall use the Set-Cookie c
	Attack protection	The system shall be protected against \text
		The system shall be protected a
		The system shall be protected against
		The system shall be protected against
	User protection	The system shall be protected a
		The system shall sanitize a
		The system shall have all user p
		The system shall encourage stro
		The system shall not show full personal information on t
		The system shall log every human-
	Failure protection	Backup the database every two we
Environmental	Browser support	The system shall support the following browsers: Google Chr
	Mobile support	The system shall support mobile devices, meaning it shoul
Development		The system shall be developed using HTML5, CSS3, JS (EC

3.4.1 Layered Diagram

Content

3.4.2 Pages Diagram

Content

3.4.3 Database Modelling

Content

3.4.3.1 Conceptual Model

Content

3.4.3.2 Logic Model

Content

3.4.3.3 Normalization

Content

3.5 Conclusion

Content

Chapter

4

Technologies

4.1 Introduction

Content

4.2 APPS Python API

In order to access the APPS web service, the Python API presents to be the most versatile (because it contains every function needed to apply all functionalities to the service[20]) and simple (because the interface is already abstracted, containing a number of classes designed to take the complexity out of interfacing with APPS[21]).

The APPS service needs authentication through the online APPS web service (at https://pppx.gdgps.net/) at least once, to download a settings file, which contains all information needed to tell the library how to find the APPS web service and how to authorize the requests the library makes against the user account[22]. The settings file will look something like the following:

```
{
    "portal": "http://pppx.gdgps.net",
    "api_path": "api/user",
    "key": "50052f46-bb9e-11e8-b79c-60f81dca17e2",
    "secret": "UkdjcmBsCl1oWScwbCJVW3N4MyMKXld4R3BLVVsoZDM="
}
```

Code sample 4.1: APPS API access settings file

The **key** and **secret** will authenticate a specific user and allow them to use that APPS through the command line. Moreover, the APPS library will sign all the API requests with the previous credentials to keep the data private and secure[23].

Besides the settings file, the API library will need to be downloaded using the pip library for python:

```
pip3 install https://pppx.gdgps.net/static/lib/gdgps-apps-1.0.0b0.tar.gz
```

Code sample 4.2: Install the APPS API through pip

4.2.1 The APPS class

The APPS class is the main interface for interaction with APPS. It greatly simplifies usage of APPS by automatically handling tasks such as data caching to speed up and reduce the amount of back and forth communication between the local machine and the APPS servers. APPS is comprised of more than one server, which adds complexity to interfacing with it. This class abstracts away the complexity of interfacing with APPS by automatically selecting the current fastest APPS processing server to upload data to and by keeping track of where the data is located and how to retrieve it[23].

The following will instantiate the class APPS, which will give access to every methods needed to use the web service:

```
from gdgps_apps.apps import APPS

apps=APPS(parameters)
```

Code sample 4.3: Instantiating the APPS class

It's defined as:

```
class gdgps_apps.apps.APPS(settings_file,portal,api_path,label,key,
    secret,download_directory,trust_env,log_level)
```

Code sample 4.4: The APPS class

The parameters of the APPS constructor are defined as follows:

- **settings_file**: path to the settings file, so that a user can authenticate and actually use the API;
- **download_directory**: path to the directory where post-processed files will be downloaded to.

From here, the class provides access to a vast set of methods to interface with the web service. The most useful ones are described here.

4.2.1.1 Check if everything is working

The `ping` method is used to check connectivity to all APPS servers and will refresh the known loads of the APPS processing servers the account is allowed to use. It's defined as:

```
ping()
```

Code sample 4.5: The `ping` method

It doesn't contain any parameters.

It returns a tuple, containing `(True,response)` if the connection could be established and `(False,None)` otherwise. The `response` will be the the response from the portal containing load data.

4.2.1.2 Upload a RINEX file

The `upload_gipsyx` method is used to upload a RINEX file to the APPS server (associated with the user who uploaded it) for it to be processed by GipsyX software suite. It's defined as:

```
upload_gipsyx(file,pressure,attitude,stream,processor,progress_hook,name
    ,email_notify,access,use_case,processing_mode,product,
    troposphere_model,ocean_loading,model_tides,elev_dep_weighting,
    elev_angle_cutoff,solution_period)
```

Code sample 4.6: The `upload_gipsy` method

Its parameters are defined as:

- **file**: the RINEX file to be uploaded or a `tar` containing a RINEX file and any ancillary source files, such as pressure files or attitude quaternion files. Both can be uploaded directly or, first, compressed using `bz2`, `gzip`, `zip`, `lzma` or Unix compression.
- **pressure**: the pressure file to be uploaded, if there is one;
- **attitude**: the attitude quaternion file, if there is one;
- **stream**: `True` if the upload should be streamed, `False` otherwise. If streaming is enabled the return value of this function will be a requests response object, otherwise the json response of the server will be returned;
- **processor**: the label of the processor to use. Leaving this blank will cause APPS to choose the least loaded processor;
- **progress_hook**: a callable, which returns the number of bytes that are being read in real time;
- **name**: the name to use for the submission. If no name is provided the file name will be used by default;

- **email_notify**: `True` if an email should be sent after processing the file and `False` otherwise;
- **access**:
- **use_case**: the processing settings most likely wanted. `defines.GIPSYData.AIRBORNE` if the receiver was in the air, `defines.GIPSYData.GROUNDED_MOBILE` if the receiver was on the ground but moving and `defines.GIPSYData.GROUNDED_STATIONARY` if the receiver was on the ground but stationary;
- **processing_mode**: compute the specific processing mode. `defines.GIPSYData.STATIC` if the receiver was static and `defines.GIPSYData.KINEMATIC` if the receiver was in motion;
- **product**: the GNSS satellite orbit and clock state data product to use to compute the solution. In increasing order of precision, `defines.OrbitClockProduct.REAL_TIM` is available immediately, `defines.OrbitClockProduct.ULTRA` is available within about two hours, `defines.OrbitClockProduct.RAPID` is available the next day and `defines.OrbitClockProduct.FINAL` is available within two weeks. `defines.GIPSYData.BEST` will use the most precise product currently available during the time frame of the measurement.

4.3 Conclusion

Content

Chapter

5

Design

5.1 Introduction

Content

5.2 Corporate Image

Content

5.2.1 Color

Content

5.2.2 Logo

Content

5.3 Interface Design

Content

5.3.1 Wireframes

Content

5.3.2 Mockups

Content

5.4 Conclusion

Content

Chapter

6

Development

6.1 Introduction

Content

6.2 Development

Content

6.3 Conclusion

Content

Chapter

7

Tests

7.1 Introduction

Content

7.2 Functional tests

Content

7.2.1 Unit tests

Content

7.2.2 Feature tests

Content

7.2.3 System tests

Content

7.3 User tests

Content

7.3.1 Work tests

Content

7.3.2 UI tests

Content

7.3.3 UX tests

Content

7.4 Performance and Load tests

Content

7.5 Security tests

Content

7.6 Conclusion

Content

Chapter

8

Conclusion

Content

8.1 Conclusion

Content

8.2 Future work

Content

Bibliography

[1] Ann Edwards. What Is the Oxford Comma (or Serial Comma)? [Online] https://tobi.oetiker.ch/lshort/lshort.pdf. Last accessed at March 14, 2022.

[2] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. ISBN: 978-0132350884. 2008.

[3] MahashrevetaWill Thornton. Unsure of the difference between GNSS and GPS? Keep reading for a brief history and introduction. [Online] https://www.spirent.com/blogs/what-is-the-difference-between-gnss-and-gps. Last accessed at March 19, 2022.

[4] What is GNSS? [Online] https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss#:~:text=Global%20Navigation%20Satellite%20System%20(GNSS,definition%2C%20GNSS%20provides%20global%20coverage.. Last accessed at March 19, 2022.

[5] What are Global Navigation Satellite Systems? [Online] https://novatel.com/tech-talk/an-introduction-to-gnss/what-are-global-navigation-satellite-systems-gnss. Last accessed at March 19, 2022.

[6] 1.2 GLOBAL NAVIGATION SATELLITE SYSTEM (GNSS). [Online] https://www.vectornav.com/resources/inertial-navigation-primer/theory-of-operation/theory-gnss. Last accessed at March 19, 2022.

[7] 1.2 GLOBAL NAVIGATION SATELLITE SYSTEM (GNSS). [Online] https://www.tersus-gnss.com/technology. Last accessed at March 19, 2022.

[8] NovAtel Inc. *An Introduction to GNSS*. ISBN: 978-0-9813754-0-3. NovAtel Inc., 2015.

[9] PHASE 3: DATA CAPTURE. [Online] https://giscommons.org/chapter-2-input/. Last accessed at March 19, 2022.

[10] Sam Souliman. RINEX files and differential corrections in post-processing. [Online] https://support.oxts.com/hc/en-us/articles/115003212489-RINEX-files-and-differential-corrections-in-post-processing#:~:text=Receiver%20INdependent%20EXchange%20format%20files,%2Dprocessing%2C%20improving%20its%20accuracy.. Last accessed at March 20, 2022.

[11] Mahashreveta Choudhary. What are various GNSS systems? [Online] https://www.geospatialworld.net/blogs/what-are-the-various-gnss-systems/#:~:text=The%20four%20global%20GNSS%20systems,IRNSS%20or%20NavIC%20(India).. Last accessed at March 19, 2022.

[12] Scrum. [Online] https://www.scrum.org/. Last accessed at March 22, 2022.

[13] GNSS Data Post-Processing. [Online] https://novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/gnss-data-post-processing#:~:text=Unlike%20RTK%20GNSS%20positioning%2C%20post,one%20or%20more%20GNSS%20receivers.. Last accessed at March 21, 2022.

[14] A. Mohammed Abou-Galala, R. Mostafa, K. Mosbeh, and Z. Zaki. Assessment of the accuracy and convergence period of precise point positioning. *Alexandria Engineering Journal*, 17(5):47–57, 2017.

[15] Free GPS Online Post Processing Services. [Online] https://gisresources.com/free-gps-online-post-processing-services/. Last accessed at March 21, 2022.

[16] GDGPS. [Online] https://www.gdgps.net/index.html. Last accessed at March 21, 2022.

[17] JPL Science. [Online] https://science.jpl.nasa.gov/projects/great/. Last accessed at March 21, 2022.

[18] The Automatic Precise Positioning Service of the Global Differential GPS (GDGPS) System. [Online] https://pppx.gdgps.net/. Last accessed at March 21, 2022.

[19] ITRF2014. [Online] https://itrf.ign.fr/ITRF_solutions/2014/ITRF2014.php. Last accessed at March 21, 2022.

[20] APPS Python Library. [Online] https://pppx.gdgps.net/static/doc/manual/library/index.html. Last accessed at April 12, 2022.

[21] API. [Online] https://pppx.gdgps.net/static/doc/manual/library/api/index.html. Last accessed at April 12, 2022.

[22] Installation. [Online] https://pppx.gdgps.net/static/doc/manual/library/installation.html. Last accessed at April 12, 2022.

[23] APPS. [Online] https://pppx.gdgps.net/static/doc/manual/library/api/apps.html. Last accessed at April 12, 2022.

Appendix

A

Models

This appendix presents several models needed to explain certain functional requirements, defined in section 3.3.1 of chapter 3 in page 20. They aren't meant as a whole representation of the system, but rather as an informative guide to aid in the development.

A.1 Sequence models

The following are sequence diagrams, that represent the interaction of certain features.

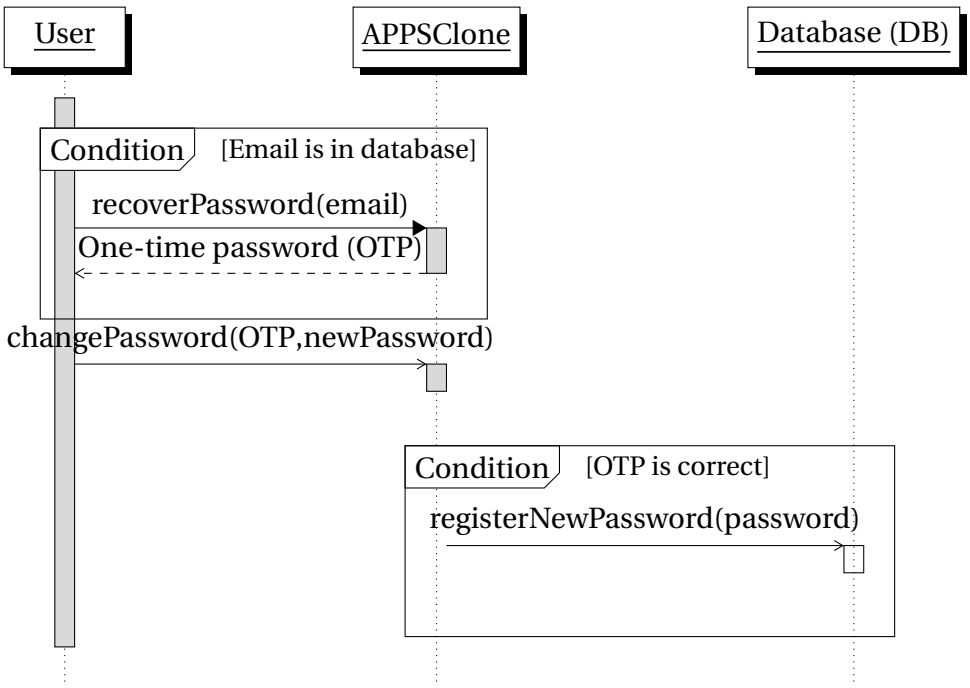

Figure A.1: Sequence diagram for the password recovering system