TÉCNICO LISBOA

Historical Sea Level: an app for the study of global sea level changes

Mihir Sanjay Odhavji

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Doutor João Nuno De Oliveira e Silva
Prof. Doutora Maria Alexandra Oliveira

Examination Committee

Chairperson: Prof. Doutora Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Doutor João Nuno De Oliveira e Silva
Member of the Committee: Prof. Doutor Bruno Emanuel Da Graça Martins

December 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Dedicated to my family

Acknowledgments

I wish to express my deepest gratitude to Professor Doutor João Nuno de Oliveira e Silva and Professora Maria Alexandra Oliveira for providing me with their availability, advice, guidance, support and motivation.

I am indebted to my family and friends for their support and encouragement in completing this stage of my academic life.

Resumo

Existe uma grande quantidade de dados sobre a variação do nível médio das águas do mar proveniente de estudos científicos realizados pelo mundo inteiro. Estes dados encontram-se dispersos, desorganizados, carecem de padronização e, na maioria dos casos, não estão disponíveis online. Mesmo quando os dados estão disponíveis, geralmente são de maneiras pouco práticas e em formatos diferentes, por isso, a análise dos dados recolhidos de várias fontes resultaria num processo ineficiente e demorado. Além disso, para processar com sucesso dados espaciotemporais, o utilizador deve estar equipado com habilidades e ferramentas específicas que são usadas para dados geográficos como o PostGIS, PostgreSQL e GeoAlchemy.

A solução apresentada consiste no desenvolvimento duma aplicação web que resolve alguns dos problemas enfrentados pelos investigadores. A aplicação permite ao utilizador adicionar dados, seja com recurso a formulários num browser ou automatizados com o auxílio de uma API. A aplicação também assiste na consulta, processamento e visualização de dados, com a criação de tabelas, exibição de mapas, desenho de gráficos assim como na comparação de dados de diferentes áreas e publicações.

A aplicação web implementada permite a consulta e armazenamento de dados espaciotemporais sobre a variação do nível médio das águas do mar de forma simplificada, acessível e amiga do utilizador e possibilita também a realização de estudos mais globais.

Palavras-chave: Nível médio das águas do mar, Aplicação Web, Base de Dados Geográfica, Python

Abstract

There is a lot of data about mean sea level variation from studies conducted around the globe. This data is dispersed, lacks organization along with standardization, and in most cases, it is not available online. In some instances, when it is available, it is often in unpractical ways and different formats. Analyzing it would be inefficient and very time-consuming. In addition to all of that, to successfully process spatial-temporal data, the user has to be equipped with particular skills and tools used for geographic data like PostGIS, PostgreSQL and GeoAlchemy.

The presented solution is to develop a web application that solves some of the issues faced by researchers. The web application allows the user to add data, be it through forms in a browser or automated with the help of an API. The application also assists with data querying, processing and visualization by making tables, showing maps and drawing graphs. Comparing data points from different areas and publications is also made possible.

The implemented web application permits the query and storage of spatial-temporal data about mean sea level variation in a simplified, easily accessible and user-friendly manner. It will also allow the realization of more global studies.

Keywords: Mean Sea Level Variation, Web Application, Geographical Database, Python

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter lays out a sketch of the work done, the primary motivation to conduct it, a short description of the problem addressed, a summary of the goals and objectives along with an overview of the solution developed.

1.1 Motivation

Climate change, which can be defined as a modification of climate patterns, has caused worldwide effects owing to changes in weather and climate extremes, a rise in global surface temperature, glacier melting and shrinking along with worldwide mean sea level increase[1]. On the social sphere, these changes are a contributing factor to worsening inequality, economic losses due to a higher number of natural disasters, crop death and failure related to drought or out of season rains, which in turn inflate the food prices[2].

A rise in sea level is especially impacting due to its effects on coastal zones, where big and important cities are mostly located[3]. Nonetheless, sea level integrates the effects of diverse complex geological and climatological phenomena[4].

As an example, locally obtained sea level changes measurements can result from vertical land motion owning to either glacial isostatic adjustments or tectonics (for example, local glacier retreat generates a decreased ice burden - or weight - over landmasses, which in turn generates land uplift), which do not indicate an increase in absolute mean sea level[5].

Understanding the climate response at global, regional and local scales from various lines of evidence is crucial to estimate climate-related risks and adaptation[1]. Even though sea level change represents one of the longest records using human instruments, going back 300 years in Europe, these are not enough to describe the non-anthropogenic related sea level increase of approximately 120m, which has occurred since the last glacial maximum ($\sim$19 000 years)[5, 6]. Therefore, the study of sea level changes is based on geological proxies that show a past position of sea level. Sea-level index indicators include instrumental evidence (for example from tidal gauges), ancient shorelines (such as raised beaches), biological indicators (such as bivalve shells), intertidal deposits (such as coastal marshes),

and archaeological evidence (for example the position of ancient harbor and port facilities)[7].

1.2 Problem

With an increasing number of studies conducted and articles published, the amount of data available about sea level variation is also growing. However, data was saved on physical paper, which is easily corruptible, difficult and slow to share at a global level, can amount to huge stacks very quickly leading places to run out of space. Nonetheless, with computers becoming more mainstream, some of the problems were solved with excel sheets.

However, excel comes with its problems like not being able to create maps and difficulty with collaboration. Every time someone wants to participate, a link has to be shared and they have to be added to the excel sheet and that is not scalable. The file could also be made public to solve the scalability issues, but that is not secure enough. Another argument against excel is the fact that it is not a relational database. If everything is on the same sheet, it can become messy very quickly, if the data is in different sheets then, firstly it can be easy to lose track of the parameters, secondly, it can not have relations between tables, and thirdly queries cannot be made.

Another option can be a desktop app. It can resolve excel's data organization and query problems, but the data sharing among researchers can be complicated, so the creation of dynamic maps and data exploration.

A further problem is the fact that even assuming such a database exists, it is not online. By having it public the database can increase in size in a manner that a private one could never. Also, more people visualizing the Database means more people looking for errors and easily correcting them if needed.

The problem is that there isn't a single database that combines all the data from all the studies conducted.

1.3 Objectives

The objective is to develop a web tool that aggregates the maximum number of data points taken from local researchers about the variation of the average sea levels and helps the study of changes in mean sea level by using data from the holes dug.

A tool that lets researchers collaborate without creating unnecessary friction (such as sharing links and files). An instrument that shows and creates dynamic maps and graphs, allows for some data query and exploration along with letting users compare the variation of the average sea level from different parts of the world.

An additional goal is to facilitate the data entries made by the researchers, help them collaborate more smoothly (without worrying about links and files shared) and do some of the data standardization required for a better comprehension of the data available.

1.4 Results

The expected results from this thesis are the creation of a web application that uses a relational database, a georeferenced database and allows for the creation of dynamic maps and graphs.

The relational database saves observations that are represented by their geographic coordinates in a georeferenced database, the age, the area, the title of the paper on which they were published and the sea level. Using these data points, the application creates maps and graphs where it is possible to compare different places for the variation of the mean sea level.

1.5 Outline

The remaining document is structured as follows:

- In chapter 2, a background is given about concepts of sea level variation and vertical land movement. An introduction is also provided to the libraries and databases used.
- In chapter 3, the implementation is discussed, the requirements are enumerated, the data model, architecture and support system explained and the endpoints listed.
- In chapter 4, results are shown and a case study is presented.
- In chapter 5, the present work is concluded and some ideas for future work are given.

Chapter 2

Background

This chapter presents the definitions of some geographical concepts such as vertical land movement and mean sea level variation. It also gives an introduction to the libraries used for mapping and plotting along with an introduction to geographical databases and ORM.

2.1 Building and Interpreting Sea level Curves

Sea level index points represent the position of past sea levels and mainly comprise biological, sedimentological, morphological, or archaeological evidence of past sea levels, further detailed in table 2.1 and shown in figure 2.1.

Each type of sea level indicator is associated with a precision (assumed as an error interval) in its relationship with the mean sea level. For example, specific foraminiferal assemblages (foraminifera are single cell organisms with particular external shells) are nowadays found in high marsh environments, living only between the highest astronomical tide and the mean high water-level [8].

So, to a great degree, by finding these shells at lower depths when studying sediment cores, researchers find evidence of shallow water conditions, within a specific height range relative to the existing mean sea level.

The next step is to estimate the age of the sediment and/or shells with the appropriate age estimation methods, such as radiocarbon dating, and a local sea level index point can be established.

Several ages estimation methods are often used in the establishment of sea level index points, largely depending on the indicator and time frame under analysis (figure 2.2). Each method has its precision, mainly related to the statistical uncertainty obtained from the physical or chemical analysis used in the age determination [9, 10]. Due to this reason, age estimation of either sediment, biological, or rock samples, is always accompanied by an error.

Sea level indicators	Example	Time Span	Accuracy
Morphological	Erosional terraces - Depositional shorelines Erosional notches	Thousands of years	Meters(m)
Biological	Organisms which can indicate evidence of sub-, mid- and supralittoral positions, such as corals, molluscs or salt marsh microfossils	Thousands of years to interannual variation	From centimetres(cm) in the case of corals to milimetres(mm) in the case of salt marsh microfossils
Sedimentary	Beach rocks, palaeo-beaches Mangrove sediments Cave deposits (submerged caves and speleothemes)	Thousands of year to decades	Centimetres (cm)
Archaelogical	Terrestrial structures or artefacts (such as settlements or dwellings) Coastal structures (such as posts or fish tanks)	Thousands of years	Centimetres (cm)

Table 2.1: Summary of main sea level indicators. Extracted from [11].

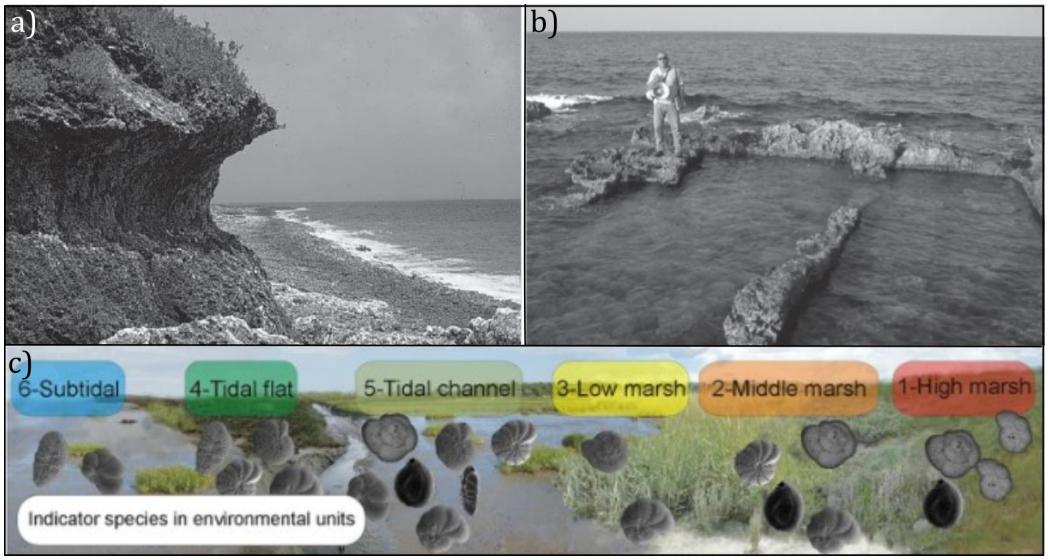


Figure 2.1: a) Erosional notch (morphological sea level indicator) at 6.4m elevation above present sea level (photograph by David Stoddart). Modified after [12]; b) Partially submerged Roman ruins in southern Italy (photograph by Colin Murray-Wallace). Modified after [12]; c) image summarizing different indicator foraminifera species assemblages of different environments in the English Channel. Modified after [13].

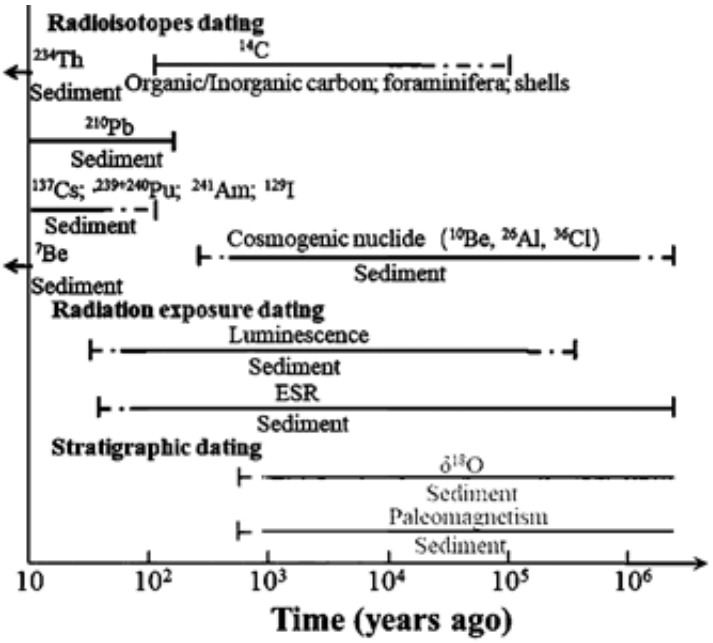


Figure 2.2: Age estimation methodologies and ranges for the past 106 years in marine sediment research, with dashed lines representing uncertainty in their use for those time ranges. Modified after [10].

Sea level curves are normally represented by scatter plots showing the relative sea level plotted against age, also named sea level index points, at a local scale.

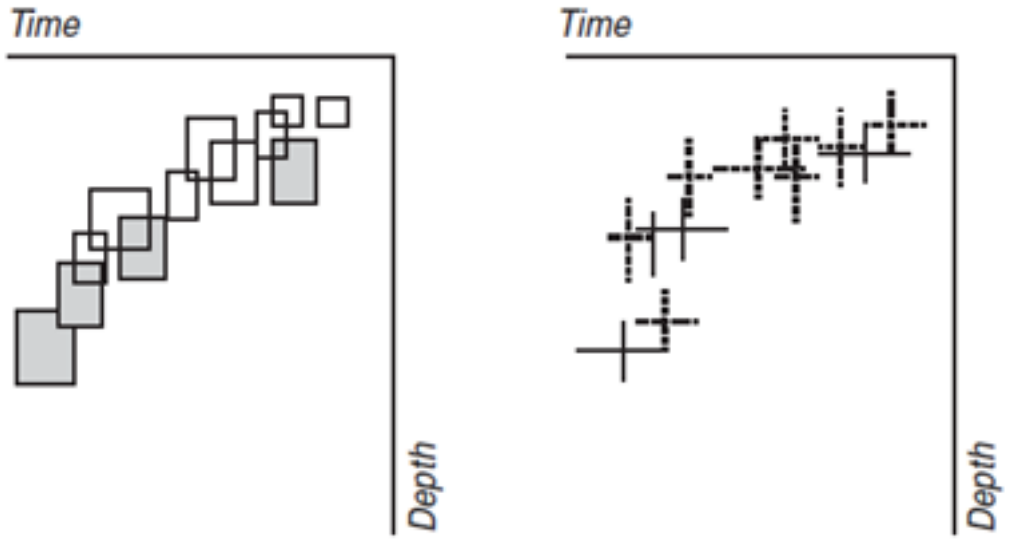


Figure 2.3: Schematic illustration of various approaches to representing a time-depth plot of sea level index data from more than one site, using polygons (image to the left) or error bars (image to the right), that incorporate precisions from the sea level indicator, including different local tidal ranges, (represented in the depth axis) and from the age estimation techniques applied (represented in the time axis). Modified after [9].

The consideration of precision/errors associated with the selected sea level indicator, and region

(due to different tidal ranges, for example), combined with age estimation uncertainties, which depend on the technique used, are extremely important when comparing sea level index points from different locations or ages, and are commonly represented by either polygons or error bars (figure 2.3) [9, 14].

Sea level curves can differ considerably in different locations (figure 2.4), as they represent the local relative sea level (height of the sea surface relative to the earth surface), mostly due to also local vertical land movement [8]. This land movement is related to several regional processes, including present and past changes and land ice mass, with major contributions from surface loading response to past glacial-isostatic adjustments, plate tectonics, as well as local processes, such as sediment compaction and past tidal range changes [8].

In fact, Holocene sea level databases are frequently used to constrain parameters in Glacial isostatic adjustment models, with important applications in the current understanding of sea level rise and its geographical variability [5, 8]. Quantification of relative and absolute sea level changes, the latter also named global mean sea level rise, is in continuous development due to the increasing amount of sea level data, and vertical land movements measured using geophysics and geodesy equipment [15].

Taking all these complex inter-related processes, the development of standardized Holocene sea level and vertical land-movement databases is pivotal to establish the global mean sea level rise and to have a better understanding of its changes and relationship with anthropogenic forcing.

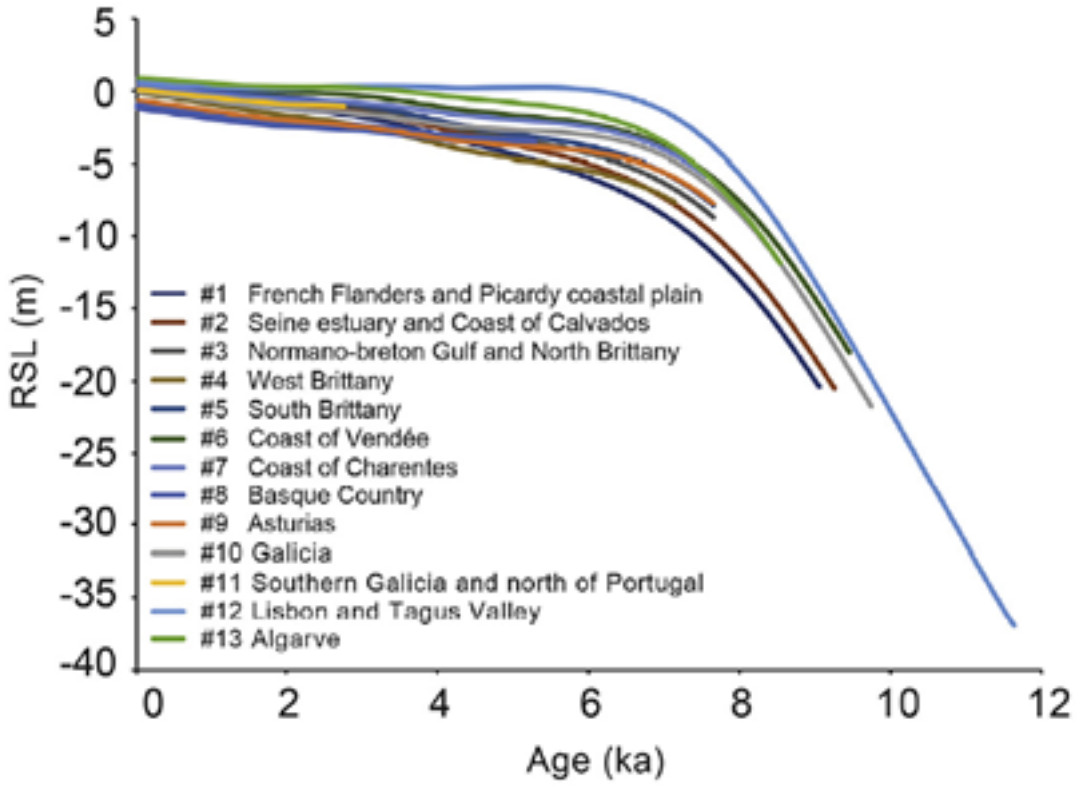


Figure 2.4: Relative sea level-curves for the Atlantic coast of Europe. Extracted from [8].

2.2 Applications for sea level variation

Currently, there are some web applications and databases related to mean sea level variation, however all of them are about future projections and showing the negative impact of the rise in mean sea level. The data that these applications show is also very recent. Some example of applications found are from :

- **NASA** - Sea Level Projection Tool[16, 17]. It is an interactive map that shows projections of global and local sea level rise. The data comes from two distinct sources: records from the 20th century and observations by satellites.

- **NOAA** - Sea Level Rise and Coastal Flooding Impacts[18]. It is an interactive map that shows the negative impact of the rise of the mean sea level. It lets the user choose a height corresponding to the rise in mean sea level, making the map change the areas which get flooded in such a scenario.

As seen in figure 2.5 the data that these applications provide only dates back to 1900, which is not old enough in geological terms.

Total Sea Level

PERIOD	ESTIMATE (mm/yr)	UNCERTAINTY (+- mm/yr)	REFERENCE	MEASUREMENT / FORECAST MODEL
1993 - 2020	3.34	0.40	Beckley et al. (2017)	Satellite altimetry
1993 - 2018	3.35	0.47	Frederikse et al. (2020)	Tide Gauges
1900 - 2018	1.56	0.30	Frederikse et al. (2020)	Tide Gauges

Steric Sea Level

PERIOD	ESTIMATE (mm/yr)	UNCERTAINTY (+- mm/yr)	REFERENCE	MEASUREMENT / FORECAST MODEL
2005 - 2019	1.10	0.20	Roemmich and Gilson (2009)	Argo Floats
1900 - 2018	0.52	0.18	Frederikse et al. (2020)	In Situ Observations

Ocean Mass

PERIOD	ESTIMATE (mm/yr)	UNCERTAINTY (+- mm/yr)	REFERENCE	MEASUREMENT / FORECAST MODEL
2002 - 2020	2.10	0.30	Watkins et al. (2015)	GRACE/GRACE-FO
1900 - 2018	1.00	0.30	Frederikse et al. (2020)	Synthesis of Satellite and In Situ Observations

Greenland Mass Loss

PERIOD	ESTIMATE (mm/yr)	UNCERTAINTY (+- mm/yr)	REFERENCE	MEASUREMENT / FORECAST MODEL
2002 - 2020	0.78	0.06	Watkins et al. (2015)	GRACE/GRACE-FO
1992 - 2018	0.42	0.04	IMBIE (2019)	Data Synthesis
1900 - 2020	0.44	0.09	Frederikse et al. (2020)	Synthesis of Satellite and In Situ Observations

Figure 2.5: Dates of the data from NASA's site [16].

2.3 Data Representation Libraries

As seen in figure 2.6 the web application that is being planned to be developed has a server side and a client side(browser).

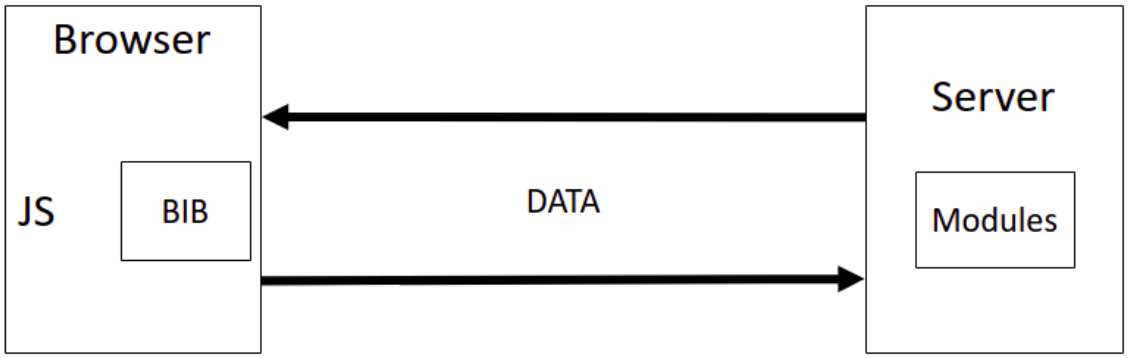


Figure 2.6: Simplified diagram of the relationship between Browser and Server.

For the client side, HTML[19] is being considered, as it is easy to understand, it has a lot of documentation, forums and tutorials online, it can be embedded with other languages such as JavaScript[20] and with some basic knowledge, is fairly easy to create forms and tables for data collection and display. Other than forms and tables, the application should show personalized maps and plot graphs. However is not possible with HTML and JavaScript alone to show personalized maps and plot graphs, so alternatives were contemplated.

For the server side, Python's [21] Flask framework[22] is being considered.

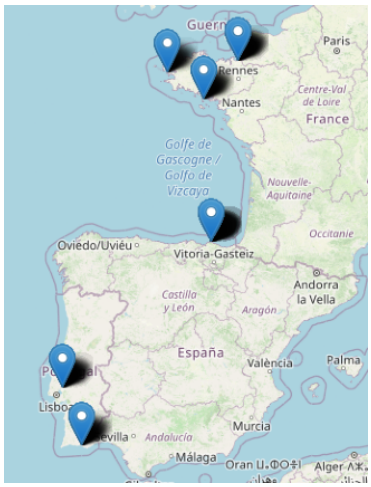
2.3.1 Mapping Libraries

One of the requirements for the application is to show maps from a specific area or publication. These maps shown should have a pin for each of the observations on that specific area or publication. The coordinates of every observation are sent to the browser via a REST/AJAX[23, 24] request. Considering all the prerequisites, two alternatives were shortlisted: Google Maps[25] and Leaflet[26].

Beginning with Google Maps, it is known to everyone, frequently used for all sorts of tasks and has an API. It can do what the conditions require, show maps and pin the coordinates of some locations. Nonetheless, the API is not free.

The other option was Leaflet, a leading open-source JavaScript library for interactive maps[26]. Being a JavaScript library it is effortlessly embedded into HTML, it is also well known and well documented and is capable of fulfilling the requirements, showing maps and pin some locations.

In figure 2.7, an example of a map shown to the user is provided, along with the JSON[27] received by the browser. The library, in this case, should use the data obtained to pin the points on the map.



(a) How coordinates are shown on a map.


```
[
    {
        'CoordinatesLat': 48.602179, 'CoordinatesLong': -1.728949
    },
    {
        'CoordinatesLat': 48.272317, 'CoordinatesLong': -4.605812
    },
    {
        'CoordinatesLat': 47.532450, 'CoordinatesLong': -3.118968
    },
    {
        'CoordinatesLat': 43.449233, 'CoordinatesLong': -2.815113
    },
    {
        'CoordinatesLat': 38.946335, 'CoordinatesLong': -8.923418
    },
    {
        'CoordinatesLat': 37.075348, 'CoordinatesLong': -8.129479
    },
    {
        'CoordinatesLat': 48.272317, 'CoordinatesLong': -4.605812
    },
]
```


(b) Example of a JSON received by the browser.

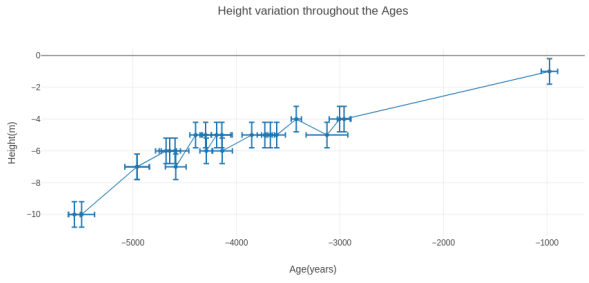
Figure 2.7: Example of a map with coordinates of observations points and the code to do it.

2.3.2 Plot Libraries

Another requirement for the application was to plot graphs for the sea level variation for a specific area and have the option to compare two or more areas. With these conditions in mind, two alternatives were finalized: Chart.js[28] and Plotly[29].

Both the options have a lot in common, such as both of them are open-source, both are easily embedded into JavaScript and HTML and both of them satisfy the plotting requirements. In the end, both the alternatives were tried and tested.

In figure 2.8, an example of a graph shown to the user is provided, along with the JSON[27] received by the browser. The library, in this case, utilizes the data received to draw the graphs and the respective vertical and horizontal error bars.



(a) How graphs are shown.


```
[
    {
        'Height'    : {'value' : -4, 'error' : 0.5},
        'Age'       : {'min' : 1500, 'max' : 1750},
        'Name'      : "Minho, Portugal"
    },
    {
        'Height'    : {'value' : -6, 'error' : 1.5},
        'Age'       : {'min' : 1000, 'max' : 1200},
        'Name'      : "Minho, Portugal"
    }
]
```


(b) Example of a JSON received by the browser.

Figure 2.8: Example of a graph plotted with error bars and the code to do it.

2.4 Geographical Databases

Being a project closely related to geology and geography, it was decided to look for tools that combine these fields with servers and Databases. ArcGIS[30], GeoServer[31], MySQL[32] and PostgreSQL[33] were found.

ArcGIS is a platform to generate, organize, distribute and inspect spatial data. It can be run locally or on cloud services. However, it is not easy to integrate on a client-server architecture.

GeoServer is a combination of a Web server with a Web Gis software and a database. It is open-source and allows users to share, manage and organize geospatial data. Nonetheless, GeoServer is very complex due to supporting various data formats such as vectors and rasters.

MySQL is a relational database management system. MySQL spatial extensions permit the users with the creation, storage and inquiry of spatial data through spatial data types and functions[34].

PostgreSQL is an open source object-relational database system that is more closely related to SQL, however, to use geographical libraries an extension named PostGIS[35] has to be added to the database.

A further advantage of PostGIS and MySQL is their capacity for doing spatial queries. Using these Database Management System, you can create various geographical data types like Points, Line, Circles, Polygons, etc[36]. These data types can interact with each other to make queries such as the meeting point of two lines or the intersection of a line with a shape[37]. This functionality is not only limited to geometry entities, it can be done with geographical bodies as well, namely lakes, rivers mountains, etc.

An example can be given about a spatial query that is searching for points that are within 10 units of distance from a point with coordinates (10, 20). The SQL query for such a code is presented in figure 2.9 a.

```
SELECT point
FROM geom_table
WHERE ST_DWithin( point, 'SRID=312;POINT(10 20)', 10);
```

(a) Example of a SQL spatial query.

(b) An illustration of the spatial query.

Figure 2.9: Example of a SQL spatial query and its visual illustration.

In figure 2.9 b, an illustration of the result from the query is provided. The blue point represents the point with coordinates (10, 20), which is at the center of a circle with a radius equal to 10 units of distance. The green points are the ones that fulfill the requirement of being within 10 units of distance from the blue point, while the red points fail to meet this rule. The data returned from the query would have been the coordinates of the green points.

2.5 ORM

The browser can create queries for a database that is connected to the server. A system is essential to translate the server's request to the SQL queries. That is where ORM enters.

Object-relational mapping (ORM)[38] is a technology that lets the user add, query, change and delete data from a database employing an object-oriented paradigm.

The ORM library is an entirely normal library written in the same language as the rest of the code. The big advantage of using ORM is that it encapsulates the code required to manipulate the data, so SQL is not used by the programmer. Direct interaction is done with the objects that use the same language as you.

For the server, Python is used. Having that in mind, options for ORM were searched, and two of them were shortlisted: SQLAlchemy[39] and Django[40]. SQLAlchemy allows the use of various database management systems with just a configuration change, however it need needs help from GEOAlchemy[41] to work with spatial databases.

The main difference between using SQL and ORM is that, while using the former generic table rows are received, but using the latter a list with objects is returned, which can be immediately used for further development.

Using SQLAlchemy and GEOAlchemy, the SQL query performed in section 2.4 would be :

```
result =  session.query(
geom_table).filter(func.ST_DWithin(geom_table.coordinates, POINT(10 20), 10))
```

Another example can be given about the creation of tables. With SQLAlchemy / GEOAlchemy, creating a table is as simple as creating a Python class, while with SQL, you have to use a specific syntax.

```
CREATE TABLE "Observation" (
        id SERIAL NOT NULL,
        coordinates geometry(POINT,-1),
        height FLOAT,
        "Error" FLOAT,
        area_id INTEGER,
        indicator_id INTEGER,
        publication_id INTEGER,
        PRIMARY KEY (id),
        FOREIGN KEY(area_id) REFERENCES "Area" (id),
        FOREIGN KEY(indicator_id) REFERENCES "Indicator" (id),
        FOREIGN KEY(publication_id) REFERENCES "Publication" (id)
)
```

(a) Example of a SQL query to create a table.

```
class Observation(Base):
    __tablename__ = 'Observation'
    id = Column(Integer, primary_key=True, autoincrement=True)


    coordinates = Column(Geometry(geometry_type='POINT'))
    height = Column(Float)
    Error = Column(Float)


    Area = relationship('Area', back_populates='observations')
    area_id = Column(Integer, ForeignKey('Area.id'))

    Indicator = relationship('Indicator', back_populates='observations')
    indicator_id = Column(Integer, ForeignKey('Indicator.id'))

    publication_id = Column(Integer, ForeignKey('Publication.id'))
    Publication = relationship('Publication', back_populates='observations')
```

(b) Example of a SQLAlchemy query to create a similar table.

Figure 2.10: Comparison of the create table query by SQL and SQLAlchemy.

Chapter 3

Historical Sea Level Application

In this chapter, the Historical Sea Level Application is described, the requirements and endpoints are listed, an Entity - Relationship diagram is provided and explained, the program modules, architecture, support system and software organization are explained.

3.1 Application problems

As seen previously, the amount of data available about sea level variation is increasing and, there isn't a single database that combines all the data from all the studies conducted. Other problems like the programs used not having support for plotting maps and drawing graphs complicate collaboration between researchers.

The solution reached is a web application that combines the data points taken from research papers about the variation of the mean sea level and stores them in a way that allows them to be used posteriorly for various purposes such as viewing maps, showing graphs, doing calculations and making comparisons.

3.2 Requirements

Before starting the implementation, it is important to outline the requirements of the program that will be developed. Such an obligation gives the developer/programmer a road map of the project that is about to start and helps in the creation of a proposal to conclude the development in time and successfully meet all the conditions.

The suggested requirements are :

- The application should be remotely accessible.
- The application should store the data generated by the researcher about proxies for the variation of sea levels (proxy data).
- The proxy data about sea level variation should be organized in areas.

- The application should store the publications where the data was published.
- The application should store information about the indicator used to estimate the age of the sample.
- Each data point should include the following information: coordinates, estimated age and height.
- The application should present the data in maps (global with all the data and particular to a publication).
- The application should create plots with the representation of the data for a specific area.
- The plots should contain error bars for the age and height estimations.
- The user should be allowed to compare different areas in the same plot.
- The user should be able to download the data in tabular format.
- The application should contain local information about the vertical land movement.
- The application should provide forms for the introduction of data to the database.

3.3 Data Model

In order to solve the problems described in 1.2, meet the goals in 1.3 and fulfill the requirements enlisted on 3.2, it was decided to build a relational database. The final result is illustrated in figure 3.1.

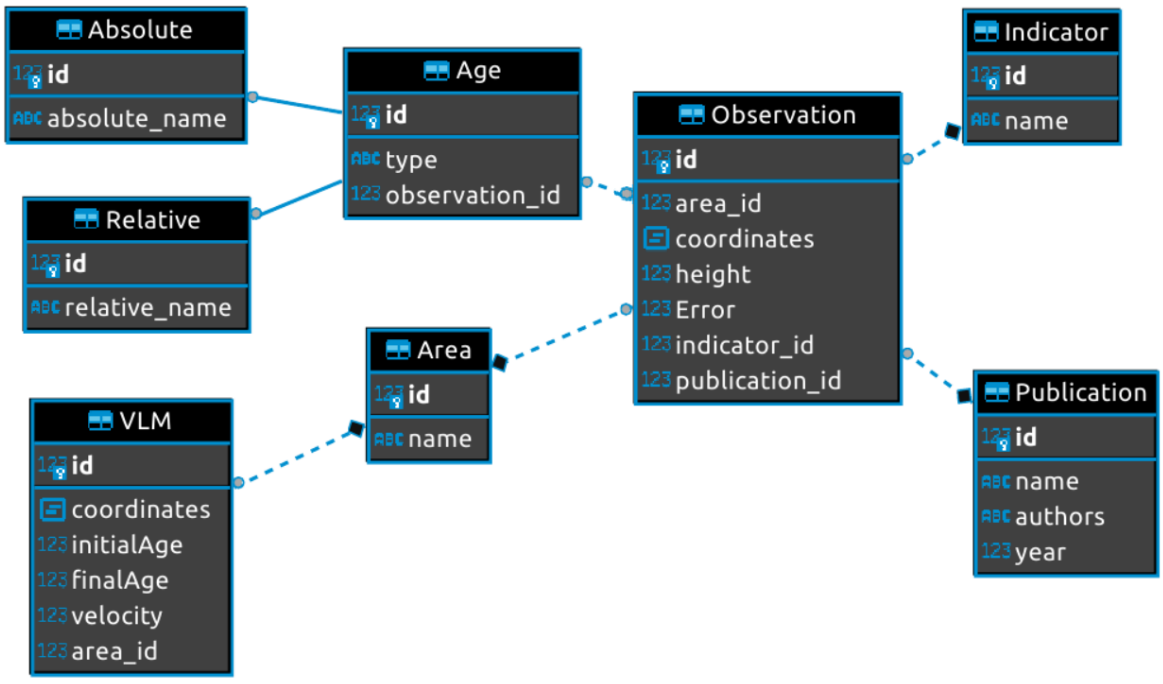


Figure 3.1: Unified Modelling Language of the data structures.

As seen in figure 3.1, there are many tables that save different data proxies. Every table has a primary key and its own ID.

The tables are:

- **Area** - a table that saves the area name and an ID number that is unique to each area.
- **Indicator** - a table that saves the indicator name and an ID number that is unique to each Indicator. An indicator on this text means the method used to determine the age of the sample. Because the same indicator can have different error values, the database does not save that value on this table, however that value is saved on the observation table.
- **Publication** - a table that saves the publication name, its authors and the year on which it was published. It also has a unique ID number for each row.
- **Vertical Land Movement or VLM** - a table that saves the coordinates of the point, the initial and final ages, the velocity in mm/yr, the ID of the area that is related to and a unique ID for each of its own entries.
- **Observation** - a table that saves the coordinates and height of the sample, the error (in meters) associated with the Indicator, the ID of the area and publication and a unique ID for each observation. The point coordinates are saved in hexadecimal in a WKBElement [42].
- **Age** - a table that saves the ID of the observation that it is related to, the type of age (Absolute or Relative) and a unique ID for each age. The Absolute and Relative ages table inherit the ID from its parent table age[43]. The Absolute type represents the case where the age is exact, e.g, 1000BC and Relative represents the instances where a time interval is provided, e.g, 1000BC-8000BC.

While plotting the graphs, the error bars for both axes need to be shown. For the graphs, the X axis represents the age of the observation and the Y axis represents its height.

The error on the X axis is related to age. If the observation's age is absolute, the error is considered zero. However, if the observation's age is relative, the middle point of the interval is considered the observation age and the values provided for the relative age are the extremes of the horizontal error bar. For example, if an observation has relative age with values: 500 - 600, while plotting the map the observation will be shown as having an age of 550 and error of $\pm$ 50. By that, the error bars go from 500 to 600, as initially stated.

Regarding the error on the Y axis, it is related to the height of the observation. When adding an observation to the database, a text box is provided to input the error in meters. These values are often provided in the publications from where the observations are taken. While plotting the graphs, the height point is the middle point of the error bar, while the lowest point is the middle subtracted by the error, and the highest is the middle point is summed by the error. As an example, if an observation has height -5m and error -1m, then the error bars will be from -6m to -4m.

After finishing the tables and defining their fields, it was essential to establish relationships between them.

Starting from the Observation table, it can be said that observation has only one area, but an area can have multiple observations. A similar logic can be applied to the relationship between Observation and Publication, Observation and Indicator and VLM and Area. So all these relationships are One-to-Many[44]. The relationship between Age and Observation is One-to-One. There is a legitimate argument that it could be a One-to-Many relationship like the one mentions previously, but given the fact that there are so many ”ages” since the beginning of time till today, it has a very low probability that two observations can have the Absolute or Relative time. If the relationship between Age and Observation was a One-to-Many kind, while adding observations, the UI would have an endless list of ”ages” to choose from. Having that in mind, it was decided it's better to go ahead with a relationship type of One to One[44].

3.4 Architecture

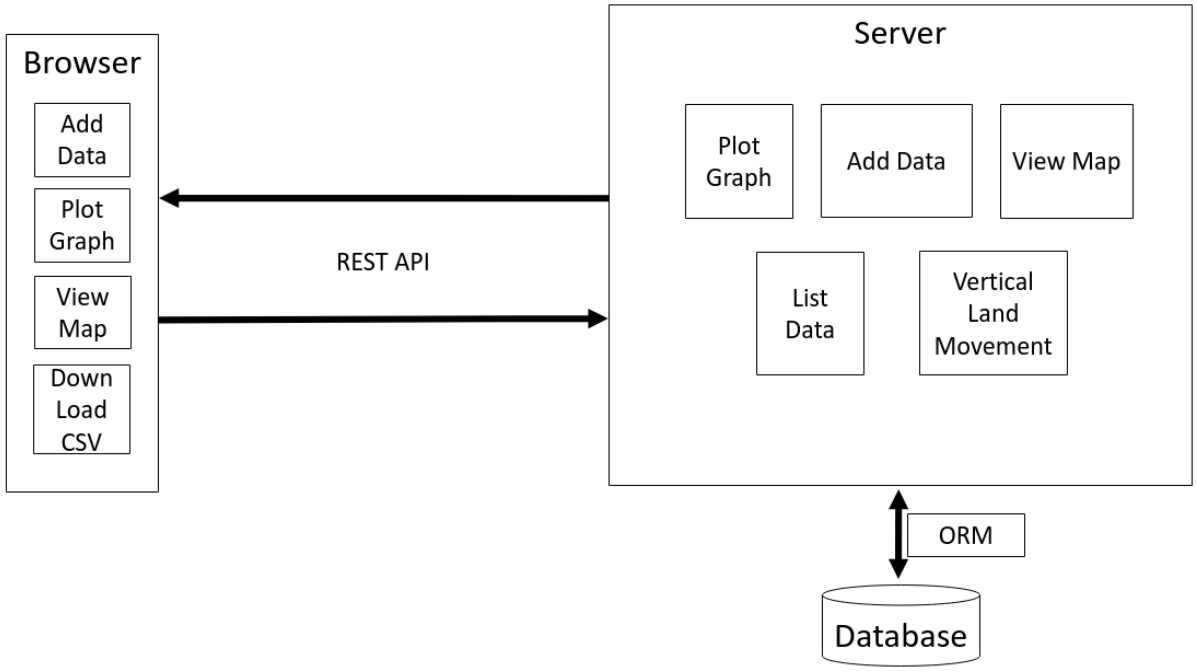


Figure 3.2: Program Module.

As seen in figure 3.2 the three main components of the program developed are:

- The browser is used as the client of the server. It is where the HTML[19] presents forms to introduce data along with showing tables, graphs and maps. The browser sends requests to the server with the help of HTTP[45]. The browser has components of the succeeding modules:
  - **Add Data** - It is a module for adding data to the database. The browser creates forms that are filled by the user to successfully add data to the database. Each data type has a different form, based on the parameters stored in the database.

  - **View Map** - With the coordinates sent by the server, the browser pins them on a flat 2D map.
  - **Plot Graph** - With the information received from the server, the browser plots it on the graph. Other tasks such as hiding and showing graph lines, zoom settings and converting the graph to an image are done by the browser.
  - **Download CSV** - It is a module that transforms the data listed on the web pages to a CSV type file[46] and allows it to be saved to your device. The module is utilized for exporting data and saving it in your own device for posterior use.

- The server receives requests sent by the client(browser) and responds to them. To successfully respond to them, it needs to convert the requests to queries for the database. To do that translation, it uses ORM[38]. After receiving the result from the database, it sends the output to the browser to show it to the user. The server interacts with the database through different modules, dependent on the requests sent by the browser, which will be discussed ahead in the document. The server is comprised of the following modules :
  - **Add Data** - It is a module for adding data to the database.
  - **List Data** - It is a module to list data from the database
  - **Vertical Land Movement** - It is a module that stores the data related to vertical land movement and can be used for future calculations.
  - **View Map** - It is a module that plots the observations' coordinates on a flat 2D map.
  - **Plot Graph** - It is a module that draws the graphs for sea level variation throughout the ages.

- The database receives queries from the server and it responds to them. It has to correctly save geographical data such as the point coordinates. It has various tables that are related to each other.

3.5 Support System

To successfully implement this web application, it was important to plan in which language to code the three main modules of the program. The browser is mainly coded in HTML and JavaScript[20], the server in Python[21], using the Flask framework[22], and the Database in PostgreSQL[33].

Starting from the browser, HTML is used to create the web pages, to receive input from the users and to show them the output, while CSS[47] is used to style it. JavaScript is utilized to create requests that are sent to the Flask server. The requests are read by the HTML from the user, built by JavaScript and sent to the server with help of REST and AJAX[23, 24]. Afterward, the JavaScript receives the response and displays it using to the user using HTML.

To produce the maps, JavaScript's Leaflet[26] library is used. Both the options presented in section 2.3.1 were considered, but the pricing was a determining factor, so it was decided to go ahead with Leaflet. Even if Google Maps[25] was free, Leaflet would be used because it is coding friendly and well documented.

For plotting the graphs it is used the Plotly[29] library. After trying both the options presented in section 2.3.2, Plotly's easiness and lack of friction in processes such as adding data to the plot was the key factor, so it was decided to go ahead with Plotly. Additional advantages of using Plotly are the fact that the user can personalize the zoom settings as well as download the graph plotted to the computer directly, without having to take screenshots.

With regard to the server, it was coded in python using the Flask framework, mainly due to its simplicity and speed in programming. The server has the responsibility to receive requests sent by the browser and give them to the ORM to query the database.

After trying Django[40] and SQLAlchemy[39], SQLAlchemy was elected to be the bridge between server and database because the author of this thesis has some experience with it from a previous project. As the database has geographical elements, the GEOAlchemy[41] library is used to add support for spatial data types. Using SQLAlchemy and GeoAlchemy libraries, the first is used to translate the queries received by the server to SQL[48] and the latter to assist the first regarding spatial queries. With GeoAlchemy it is also possible to add certain geographical data types such as 'POINT' to represent the latitude and longitude of an observation.

At last, the Database selected is PostgreSQL. It is an advanced form of SQL that allows the use of the normal SQL functions like query, primary keys, triggers, etc plus the use of Geospatial tools like GeoAlchemy and its Geometry library to enable the creation of POINT data type. PostGIS is also installed as an extender for PostgreSQL. Other options like MySQL[32] were rejected because GeoAlchemy only supports PostGIS, while GeoServer[31] and ArcGIS[30] were very complex to start programming.

3.6 End Points REST

For a secure and trustworthy connection between browser and server, it was decided to implement it using REST endpoints.

The REST Endpoints Implemented are the following :

1. **"/API/Area/", 'GET'** - Used to get the list of all areas. It receives back a python dictionary with the area ID and area name.

2. **"/API/Pub/", 'GET'** - Utilized to get the list of all publications. It is given back a python dictionary with the publication ID, its' name, the authors' name and the year on which it was published.

3. **"/API/Obs/", 'GET'** - It has the objective to get the list of all observations. It accepts the delivery of a python dictionary with the observation ID, the names of the area, publication and indicator associated with that observation, the coordinates(Latitude and Longitude), the height, the age and the error associated with the indicator.

4. **"/API/Indicator/", 'GET'** - Implemented to get the list of all indicators. It welcomes back a python dictionary with the indicator ID and indicator name.

5. **"/API/VLM/", 'GET'** - Exercised to get the list of all vertical land movements. It receives back a python dictionary with the VLM ID, the coordinates(Latitude and Longitude), the area name, the initial age, final age and velocity(in mm/yr).

6. **"/API/GetName/', 'POST'** - It has the goal to get a name of area or publication. The data sent has to be a dictionary with the "AreaID" and "PubID" as keys and values are either: the ID of the area or publication you want the name of and the other value has to be 0. The data received back is a python dictionary with the name of the area or publication requested.

7. **"/API/GetObservations/", 'POST'** - Employed to get a list of observations based on ID. It could be area ID or publication ID. The format for the request is equal to the one described for the sixth point with the exact same rules. It receives back an identical python dictionary as received by the API request on the third point.

8. **"/API/AddArea/", 'POST'** - Request used to create a new area. It sends a python dictionary with 'Area' key and the value is the name of the area that you want to create. On success, it gets returned the ID of the new area created.

9. **"/API/AddPub/", 'POST'** - Request utilized to create a new publication. It sends a python dictionary with keys: 'Name', 'Aut' for authors and 'Year'. The values are the name of publication, the name of the authors and the year of publication are given to their respective keys. On success, it gets receives back the ID of the new publication created.

10. **"/API/AddObs/", 'POST'** - Request sent to create a new observation. It sends a python dictionary with keys : 'Lat', 'Long', 'Alt', 'Area', 'Pub', 'AgeID', 'AbsoluteAge', 'BP', 'UpperAge', 'LowerAge', 'Ind' and 'Error'. The keys represent the Latitude, Longitude, Altitude, area name, publication name. The 'AgeID' works like a flag that indicates with the age of this observation is an absolute one or a relative one. If it is an absolute age then 'AbsoluteAge' is filled, if it is a relative age, then 'UpperAge' and 'LowerAge' are filled. The indicator name and the error associated with it are filled in the last two keys. On success, it gets receives back the ID of the new observation created.

11. **"/API/AddInd/", 'POST'** - Request used to create a new indicator. It sends a python dictionary with 'Ind' key and the value is the name of the indicator that you want to create. On success, the ID of the new indicator created is returned.

12. **"/API/AddVLM/", 'POST'** - Request used to create a new vertical land movement. It sends a python dictionary with keys : 'Lat', 'Long', 'InitialAge', 'FinalAge', 'Velocity' and 'Area' and their corresponding values. On success, it receives the ID of the new VLM created.

13. **"/API/GetChart/", 'POST'** - Request created to plot a graph. It sends a python dictionary with key 'ID ' and the value is the ID of the area you want to make the graph for. If you send 0 as an ID it sends back information to plot the graphs for all the areas available on the database. On success, it receives back information needed by the library to create the graphs.

The returned values are then used to create the output shown to the user on the browser such as tables, maps and Graphs.

3.7 Software Organization

The figure 3.2 illustrates the main components of the program developed: browser, server and database. With the browser and database already discussed, the focus of this section is on the server. Analyzing figure 3.2 can be noted that is made of smaller modules that have a specific task at hand. The modules illustrated in figure 3.2 are:

- **Add Data** - It is a module to add data to the database. It can be an area, publication, observation, indicator or vertical land movement. To help with the standardization while adding data, an algorithm was implemented that converts ages from the Before Present Scale to the AD/BC scale. It does so by taking 1950 and subtracting the age introduced in the Before Present Scale. It utilizes the following REST endpoints already described in section 3.6:
  - **"/API/AddArea/", 'POST'** - Request used to create a new area.
  - **"/API/AddPub/", 'POST'** - Request utilized to create a new publication.
  - **"/API/AddObs/", 'POST'** - Request sent to create a new observation.
  - **"/API/AddInd/", 'POST'** - Request used to create a new indicator.
  - **"/API/AddVLM/", 'POST'** - Request used to create a new vertical land movement.
- **List Data** -It is a module to list data from the database. It can be an area, publication, observation, indicator or vertical land movement. There is also the possibility to list observations for a particular area or publication. As the point coordinates are saved in hexadecimal, a function was created to convert them back to decimal, so that the coordinates are readable for the user. The module makes use of the upcoming REST endpoints already explained in section 3.6:
  - **"/API/Area/", 'GET'** - Used to get the list of all areas.
  - **"/API/Pub/", 'GET'** - Utilized to get the list of all publications.
  - **"/API/Obs/", 'GET'** - It has the objective to get the list of all observations.
  - **"/API/Indicator/", 'GET'** - Implemented to get the list of all indicators.
  - **"/API/VLM/", 'GET'** - Exercised to get the list of all vertical land movements.
  - **"/API/GetName/', 'POST'** - It has the goal to get a name of area or publication.
  - **"/API/GetObservations/", 'POST'** - Employed to get a list of observations based on ID.
- **View Map** - It is a module that plots the observations' coordinates on a flat 2D map. You can see all the points on the Database or filter for a particular area or publication. It uses the ensuing REST endpoints described in section 3.6:
  - **"/API/Obs/", 'GET'** - It has the objective to get the list of all observations.

  - **”/API/GetObservations/”, ’POST’** - Employed to get a list of observations based on ID.

- **Plot Graph** - It is a module that draws the graphs for sea level variation throughout the ages. You can see it for each area individually, see the graphs for all the areas simultaneously and able or disable certain areas. The module also draws the horizontal and vertical error bars for the graphs. It makes use of the following REST endpoint described in section 3.6:

  - **”/API/GetChart/”, ’POST’** - Request created to plot a graph.

- **Vertical Land Movement** - It is a module that stores data related to the vertical land movement and later can be used for rectifying the data introduced on the database by applying a correcting factor related to tectonic movement and geographic location. It does not use any of the REST endpoints explained in section 3.6.

Chapter 4

Results

This chapter demonstrates how the previously listed requirements were fulfilled along with the presentation of a case study.

4.1 Demonstration

This section presents how the requirements were fulfilled in the project.

4.1.1 The application should be remotely accessible

The creation of a web application satisfies this requirement. The historical sea level application runs on a browser, it is connected to a server that communicates to a database using an ORM. An API is also provided to send requests to the server and received information. It is also possible to develop other web applications that connect to the server and show data differently.

Other options were studied and considered as the development of a desktop application. However, with such a solution, it would be difficult to share data between researchers.

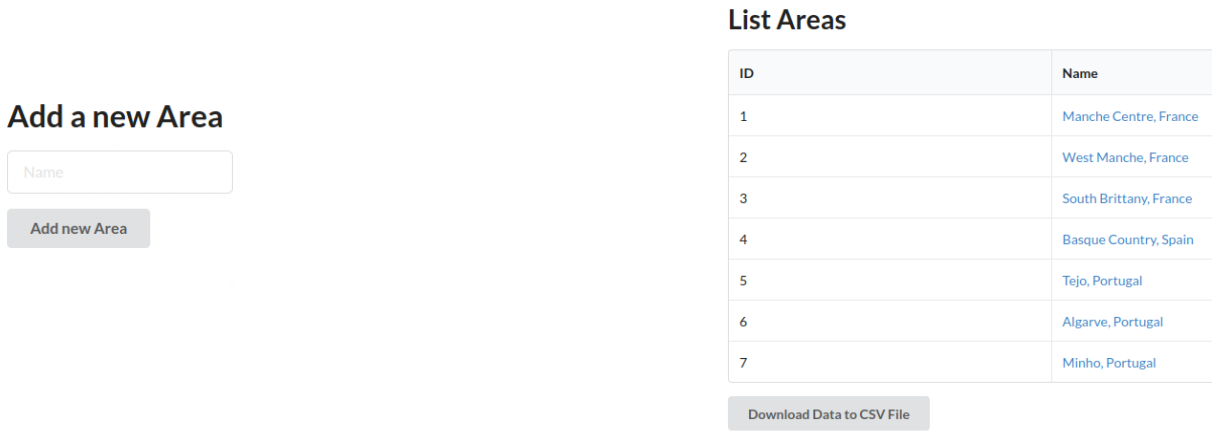
4.1.2 The application should store the data generated by the researcher about proxies for the variation of sea levels (proxy data)

The creation of a database with the model given in figure 3.1, utilized to store the different data types related to an observation meets this requirement. The data is not only stored on a georeferenced database but there is also the possibility of doing spatial queries as seen in section 2.4. This can be a foundation, where future works could be built.

4.1.3 The proxy data about sea level variation should be organized in areas

According to the UML provided in figure 3.1, a table with the name Area is created, that saves the name of the area and a unique ID to identify it. This table is related to the Observation table as well as

the Vertical Land Movement table. Therefore, the proxy data is organized in areas. The papers, from where the data points are taken, organize them in areas as well. Those areas are the ones that later are plotted in the graphs. While adding an observation, the user has to choose to which area the observation is related to (figure 4.10). If the area is not in the database, the user can add it by filling a form similar to the one present in figure 4.1 a. The user can also check a list with all the areas in the database(figure 4.1 b). If any of the areas in figure 4.1 is clicked on, then the user is taken to a page with the list of all observations of that area.



(a) Form to create a new area.

(b) List of all the areas in the database.

Figure 4.1: How the application adds and lists areas.

4.1.4 The application should store the publications where the data was published

As stated by the UML given in figure 3.1, a table with the name Publication is created, that saves the name of the publication, the name of its' authors, the year it was published and possesses a unique ID to identify it. As seen in figure 4.2, the application stores the publications where the data was published. If any of the publications in figure 4.2 is picked, then a page with the list of all observations taken from that publication shows up.

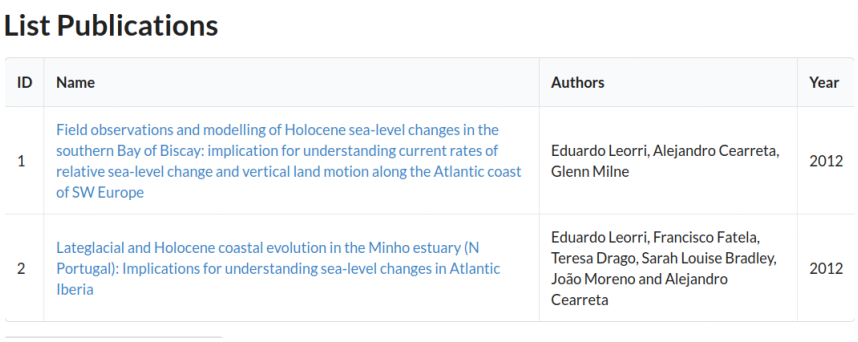


Figure 4.2: The list of publications stored in the database.

4.1.5 The application should store information about the indicator used to estimate the age of the sample

According to the UML given in figure 3.1, a table with the name Indicator is created, that saves the name of the indicator along with a unique ID to identify it. This table is related to the Observation table as well. As seen in figure 4.3, the application contains local information about the Indicator.

List Indicators

ID	Name
1	Read peat
2	Amorphous peat
3	Peaty gyttja + reed remains
4	Oxidized reed peat
5	Reed peat + leaf remains
6	Reed peat + reed remains
7	Humic clay + vegetation remains

Figure 4.3: The list of indicators stored in the database.

4.1.6 Each data point should include the following information: coordinates, estimated age and height

As stated by the UML provided in figure 3.1, a table with the name Observation is created, that saves the coordinates and the height they were found. It is related to the Age table, which saves the Age type(Absolute or Relative) as well as the value. As seen in figure 4.4, it can be concluded that this requirement was fulfilled.

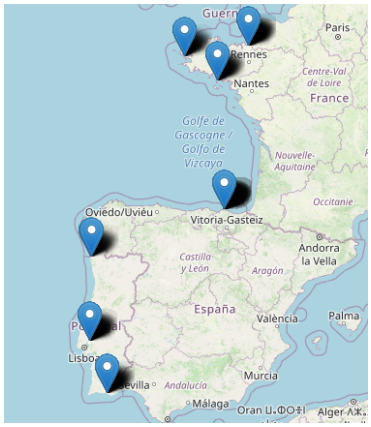
List Observations

ID	Area Name	Publication Name	Coordinates(Lat Long)	Height(m)	Age	Indicator	Error(m)
1	Manche Centre, France	Field observations and modelling of Holocene sea-level changes in the southern Bay of Biscay: implication for understanding current rates of relative sea-level change and vertical land motion along the Atlantic coast of SW Europe	48.602179 -1.728949	-10	5622BC - 5507BC	Read peat	0.8
2	Manche Centre, France	Field observations and modelling of Holocene sea-level changes in the southern Bay of Biscay: implication for understanding current rates of relative sea-level change and vertical land motion along the Atlantic coast of SW Europe	48.602179 -1.728949	-6	4746BC - 4541BC	Amorphous peat	0.8
3	Manche Centre, France	Field observations and modelling of Holocene sea-level changes in the southern Bay of Biscay: implication for understanding current rates of relative sea-level change and vertical land motion along the Atlantic coast of SW Europe	48.602179 -1.728949	-5	4352BC - 4243BC	Read peat	0.8

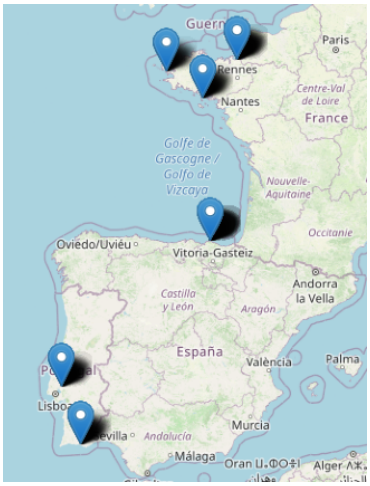
Figure 4.4: The list of observations stored in the database.

4.1.7 The application should present the data in maps (global with all the data and particular to a publication)

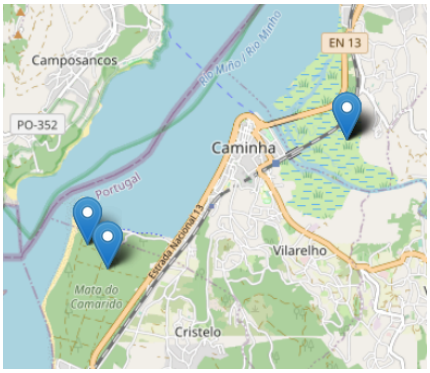
The application can show a list of all the publications in the database. By selecting one, it is possible to see a map with the coordinates points of observations taken from that publication. An example is given in figure 4.5 b where the coordinates were taken from [49]. The application also has a tab called Map to see the location of all observations on the database. The respective example is provided in figure 4.5 a. A third option is also available, by selecting an area from a list of areas(figure 4.1 b), the application shows a map with the coordinates of all the observations from that area(figure 4.5 c).



(a) Map with all the coordinates on the database.



(b) Map with all the coordinates related to the same publication.



(c) Map with all the coordinates from the Minho region in Portugal.

Figure 4.5: The three types of maps present on the web application.

4.1.8 The application should create plots with the representation of the data for a specific area

The application gives the possibility to see a list of all areas in the database. By selecting one, it is feasible to spot a graph with the data point about that area. An example is given in figure 4.6.

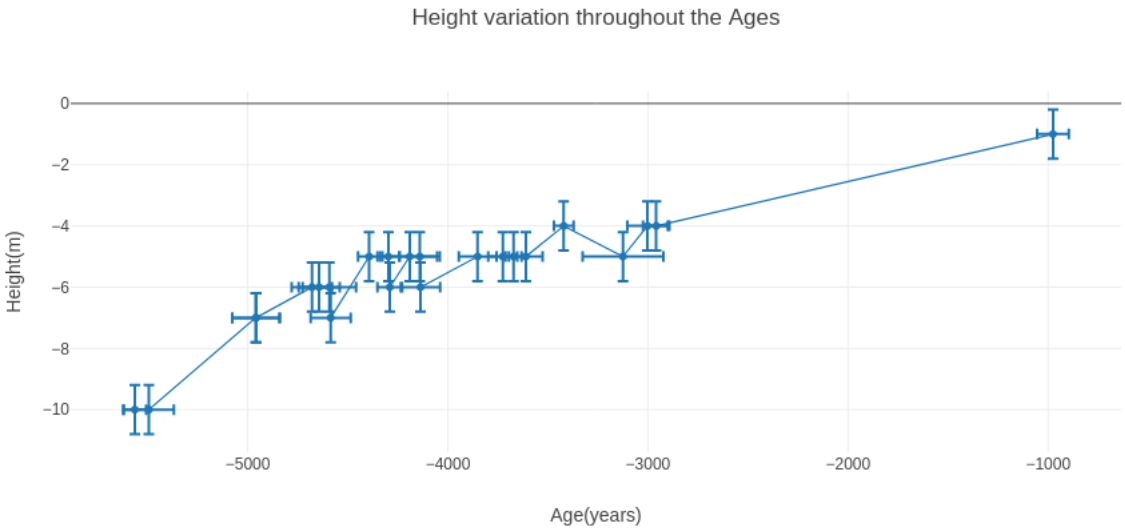


Figure 4.6: Graph plotted with data from only one area.

4.1.9 The plots should contain error bars for the age and height estimations

As seen in figure 4.6, the plots have error bars. The vertical error bars are related to the height estimations, while the horizontal bars are related to the age estimation. Compared with the graph in figure 4.7 [49], it can be concluded that the graphs plotted by the application are similar to the graphs drawn on the research papers.

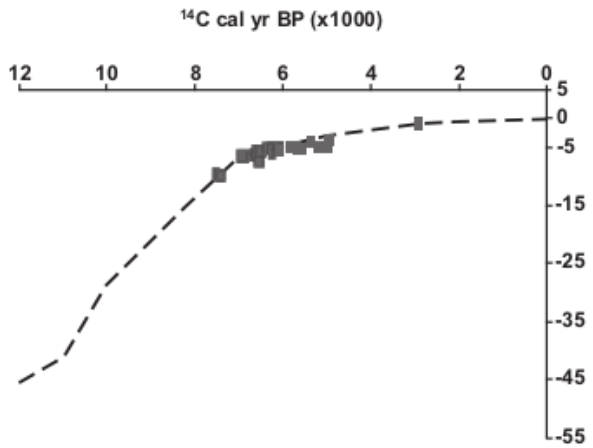
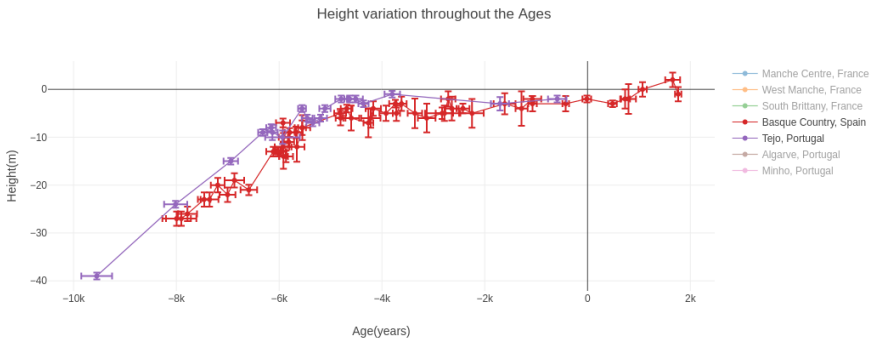


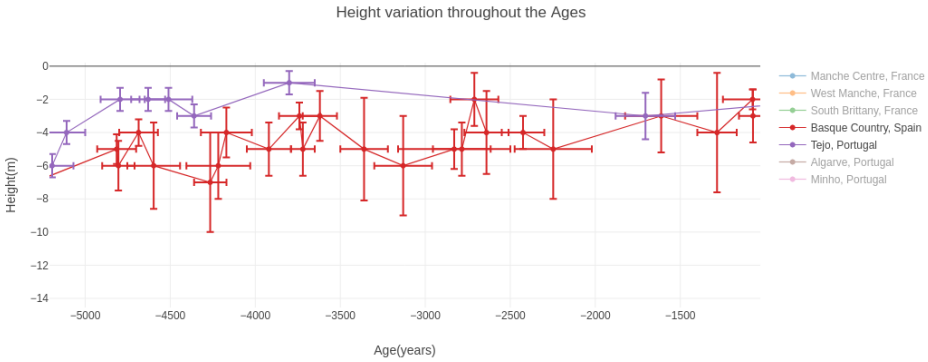
Figure 4.7: Graph plotted with data from only one area. Graph taken from [49].

4.1.10 The user should be allowed to compare different areas in the same plot

The application has the option to see the graphs of all areas in the same plot. On that plot, there is a choice to hide and unhide certain areas.



(a) Graph plotted to compare data from two areas.



(b) Graph zoomed in.

Figure 4.8: The three types of maps present on the web application.

As seen in the figure 4.8, the selected areas appear brighter than the hidden ones, which appear a bit more shadowy.

4.1.11 The user should be able to download the data in tabular format

To download the data from the database to your device, at the end of every list on the web application a download button is placed similar to the one in the figure 4.9. By clicking on it, the data from the table that the user is consulting gets written to a CSV file[46] and downloaded.

Figure 4.9: Button used to download data to a CSV file.

As an example, if the table with the list of observations for a particular area was to be transformed and downloaded, the resulting CSV file would have the following columns :

- **ID** - a unique identifier of the observation.
- **Coordinates** - Latitude and Longitude of the local of the observation.
- **Height** - The height at which the sample was found.

- **Age** - The estimated age in the AD/BC scale.
- **Indicator** - The indicator used to estimated the age.
- **Error** - The error associated with the indicator.

The data in the downloaded CSV file can be processed and used with other tools or applications.

4.1.12 The application should contain local information about the vertical land movement

As claimed by the UML provided in figure 3.1, a table with the name Vertical Land Movement is created, that saves the coordinates of the point, the initial and final Ages, the velocity and has a unique ID to identify it. This table is related to the area table as well. Thus, the application contains local information about vertical land movement. Besides the fact that data about vertical land movement is only stored and not processed, its' existence in the database allows for the further development and implementation of a simpler model. This tool will be useful for future work about the relation between vertical land movement and mean sea level variation because it is possible to realize studies at a small and local scale along with at a larger and global scale.

4.1.13 The application should provide forms for the introduction of data to the database

The application provides forms for the user to introduce data into the database. The form seen in figure 4.10 is for adding observation to the database. Similar forms are available to introduce the different data types: Area, Publication, Indicator and Vertical Land Movement.

The input numbers in the figure 4.10 are:

1 - To add the Latitude coordinate. It has to be between -90 and 90 degrees.

2 - To put the Longitude coordinate. It has to be between -180 and 180 degrees.

3 - To introduce the altitude value.

4 - To write the error value(in meters).

5 - To choose the area related to the observation.

6 - To select the indicator used to estimate the observation's age.

7 - To pick the publication on which the observation was published.

8 - For choosing the age type - absolute or relative. If absolute is selected, the text box on number eleven is hidden, and only the text box on number ten is used to input the age value.

9 - If the Before Present option is selected, it automatically converts the Age values to the AD/BC scale.

10 - To input the Upper Limit in case the Age type is relative, in case the age type is absolute, this is the only text box used to input the value.

11 - To input the Lower Limit in case the Age type is relative, if it is absolute, then this text box is hidden.

12 - Button used to submit data into the database.

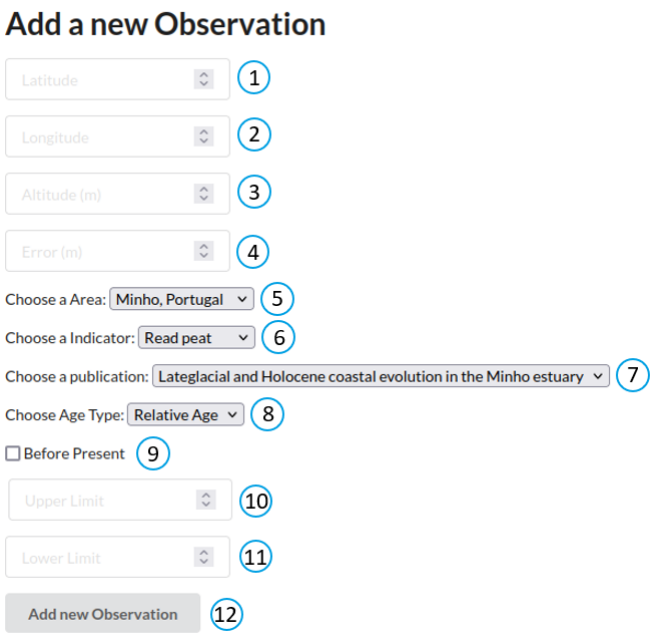


Figure 4.10: Form utilized to introduce an observation into the database.

To ease the navigation between pages of the web application, a menu (figure 4.11) was created that is visible on all the pages. The menu has seven options :

- **Home** - Brings the user back to the homepage.
- **Area** - To add or list all areas.
- **Publication** - To add or list all publications.
- **Observation** - To add or list all observations.
- **Indicator** - To add or list all indicators.
- **Map** - To see a map with all the different coordinates of the observations stored in the database.
- **Graphs** - To see a graph with the mean sea level variation of all the areas stored in the database.

Figure 4.11: Main menu of the web application developed.

While hovering above some of the options, a drop down menu (figure 4.12) appears with two options: add or list all.

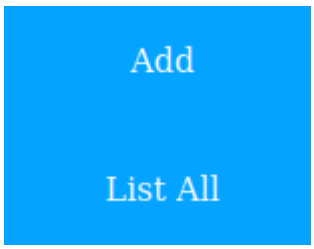


Figure 4.12: Drop down menu that appears when the user hovers above certain options from the main menu.

If the option to list data is clicked on, a list similar to figure 4.1 b shows up. If any area is selected, then a page with a list of observations related to that area emerges.

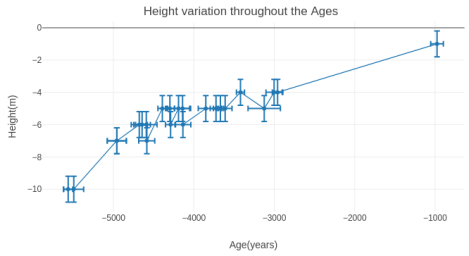
Something similar happens with the option to list publications where a list similar to figure 4.2 shows up. If any publication is picked, then a page with a list of observations related to that publication emerges.

4.2 Case Study

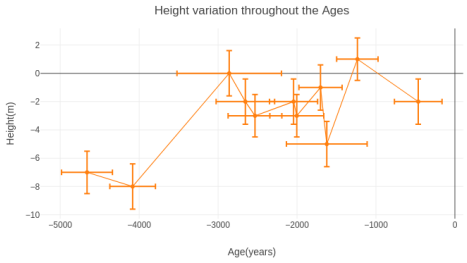
To do this case study, observations from seven different areas published on papers [49] and [50] were introduced to the database. These areas are:

- Manche Centre, France [49]
- West Manche, France [49]
- South Brittany, France [49]
- Basque Country, Spain [49]
- Tejo, Portugal [49]
- Algarve, Portugal [49]
- Minho, Portugal [50]

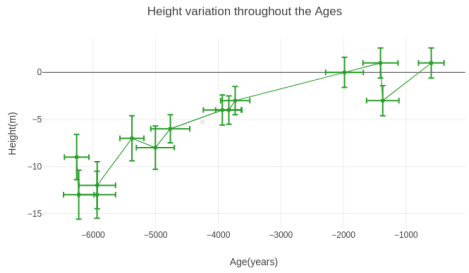
For each of the previously mentioned areas, a graph was plotted.



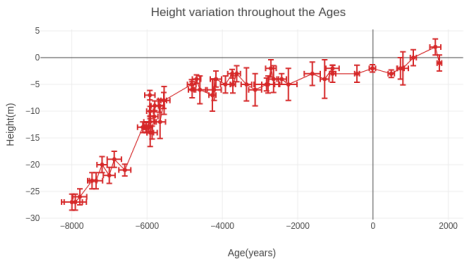
(a) Graph plotted for Manche Centre, France.



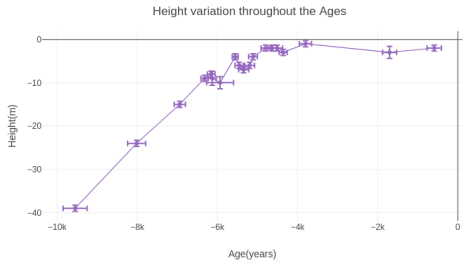
(b) Graph plotted for West Manche, France.



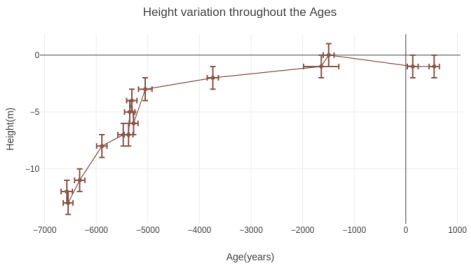
(c) Graph plotted for South Brittany, France.



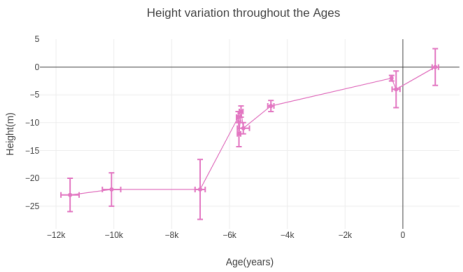
(d) Graph plotted for Basque Country, Spain.



(e) Graph plotted for Tejo, Portugal.



(f) Graph plotted for Algarve, Portugal.



(g) Graph plotted for Minho, Portugal.

Figure 4.13: Graph plotted for each of the areas mentioned in papers [49] and [50].

As seen before in section 4.1.10, the web application can plot graphs with data from two different areas. Using this feature, two different plots were created: one with graphs that look similar, and another plot with graphs that look a bit different.

The graphs in figure 4.14, show two lines that appear similar. In this case, not much can be concluded

by the graph alone, but it can be said that there weren't events that changed the direction of the two lines.

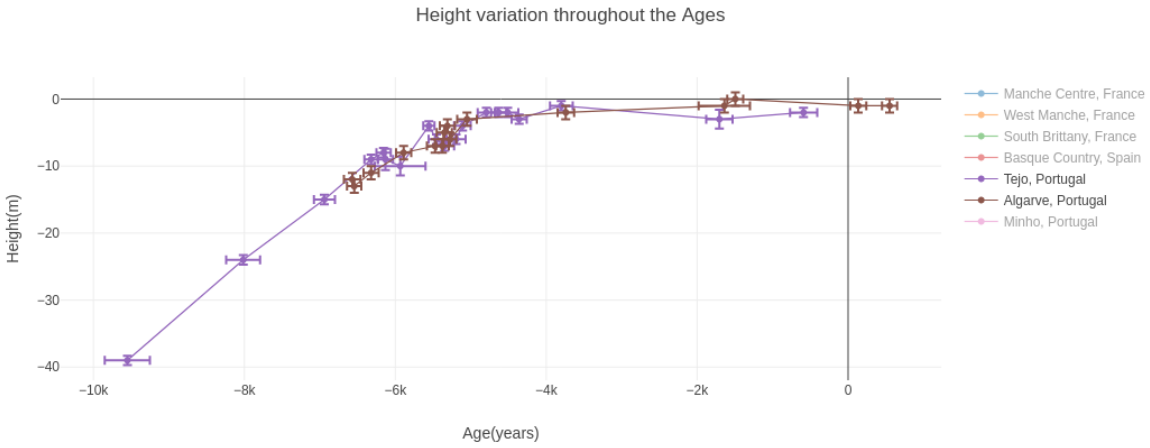


Figure 4.14: Graph plotted for two similar areas, Tejo and Algarve both in Portugal[49].

The comparison between the Tejo and Minho data, represented in figure 4.15, shows significant differences between sea level index points older than -7k. This could be related to data correction by different authors, which highlights the importance of customized correction and error attribution of data. In this case, critical data comparison is necessary to understand these differences and to adapt data collection/processing. These differences were detected only due to the use of this system.

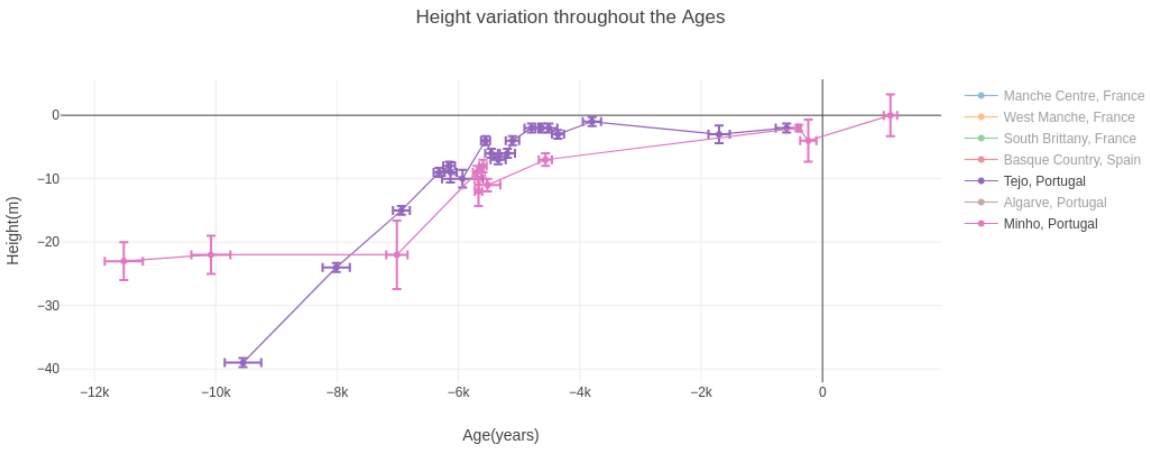


Figure 4.15: Graph plotted for two different areas, Tejo and Minho both in Portugal[49] [50].

Chapter 5

Conclusions

Until now, there was not a platform that aggregated spatial information and allowed at a global scale the processing of data related to mean sea level variation and vertical land movement.

With this thesis, a web application was created that aggregates data from numerous publications and makes them available in an organized and researcher friendly manner. The application can be accessed by anyone from a browser. A REST API was also developed, so requests can also be automated.

The user can also add data to the database in a very simple and frictionless way. The data added is standardized, especially the age value can be converted from Before Present to the AD/BC scale. Data about the vertical land movement is also stored on the database and can be used for further work.

The developed application lists the data in a clean and organized manner. That data can be viewed online or downloaded to a CSV file for other works. The developed application also has the possibility of viewing maps and plotting graphs. Additionally, there is the possibility of data comparison. The plots about the sea level variation throughout the ages can have graphs from more than one one area enabling, this way, the comparison between two or more areas.

This project, for the first time, allows the creation of interactive graphs (with features like zoom in/out and selecting which areas to hide or show) about the mean sea level variation. This new way of data presentation not only permits, easy comparison between distinct areas but also for the availability of this content on the web, to increase the visibility of such problems.

The graphs plotted in figures 4.14 and 4.15 are a tool used by specialists, who try to understand what kind of events occurred in those areas. However, these types of graphs help identify the time intervals on which the events might have happened. These types of graphs can be used to be shown to the general public to create awareness about the rising global sea level problem and help them understand this phenomenon.

The current version only stores the values related to the vertical land movement. It is also the first time, that it was possible to aggregate on one platform these types of data at a global level. This will allow scientists and data modelers, the development of new theories and/or models to explain the differences between the various areas. Using the spatial queries abilities of PostGIS discussed section in 2.4, combined with the data available on the database about the vertical land movement, mainly the

geographic coordinates, a model can be developed that automatically suggests corrections based on the location of the observations.

At the moment, the web application does not offer an authentication mechanism. Such a module could be easily implemented (for example, an integration with Google using OAuth protocol[51]. Such an addition would empower a more secure and trustworthy collaboration between researchers.

Another feature could be the creation of a webpage, where the user chooses areas from a list to plot a graph that later can be integrated into an iframe to be embedded into some other site. To create such a new API endpoint would be needed.

Bibliography

[1] P Zhai, A Pirani, SL Connors, C Péan, S Berger, N Caud, Y Chen, L Goldfarb, et al. Climate change 2021: The physical science basis. *Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*, 2021. https://www.ipcc.ch/report/ar6/wg1/downloads/report/IPCC_AR6_WGI_SPM.pdf.

[2] Stephane Hallegatte. *Shock waves: managing the impacts of climate change on poverty*. World Bank Publications, 2016. https://openknowledge.worldbank.org/handle/10986/22787.

[3] Christopher Small and Robert J Nicholls. A global analysis of human settlement in coastal zones. *Journal of coastal research*, pages 584–599, 2003. https://www.jstor.org/stable/4299200.

[4] Vivien Gornitz. Sea level change, post-glacial. *Encyclopedia of Paleoclimatology and Ancient Environments*, pages 887–893, 2009. https://doi.org/10.1007/978-1-4020-4411-3_207.

[5] Nicole S Khan, Benjamin P Horton, Simon Engelhart, Alessio Rovere, Matteo Vacchi, Erica L Ashe, Torbjörn E Törnqvist, Andrea Dutton, Marc P Hijma, and Ian Shennan. Inception of a global atlas of sea levels since the last glacial maximum. *Quaternary Science Reviews*, 220:359–371, 2019. https://doi.org/10.1016/j.quascirev.2019.07.016.

[6] Mark Siddall, Eelco J Rohling, A Almogi-Labin, Ch Hemleben, D Meischner, Ina Schmelzer, and DA Smeed. Sea-level fluctuations during the last glacial cycle. *Nature*, 423(6942):853–858, 2003. https://doi.org/10.1038/nature01690.

[7] Michael S Kearney. Sea level indicators. *Encyclopedia of Paleoclimatology and Ancient Environments*, pages 899–902, 2009. https://doi.org/10.1007/978-1-4020-4411-3_209.

[8] Ane García-Artola, Pierre Stéphan, Alejandro Cearreta, Robert E. Kopp, Nicole S. Khan, and Benjamin P. Horton. Holocene sea-level database from the atlantic coast of europe. *Quaternary Science Reviews*, 196:177–192, 2018. https://doi.org/10.1016/j.quascirev.2018.07.031.

[9] Mike Walker. *Quaternary dating methods*. John Wiley and Sons, 2005.

[10] Wenpeng Li, Xinxin Li, Xi Mei, Fan Zhang, Jingping Xu, Chunru Liu, Chuanyi Wei, and Qingsong Liu. A review of current and emerging approaches for quaternary marine sediment dating. *Science of The Total Environment*, 780:146522, 2021. https://doi.org/10.1016/j.scitotenv.2021.146522.

[11] E.S.M. Zugabi. *Environmental and economic impacts of sea-level rise on the Basque Coast.* PhD thesis, Universidad del País Vasco, 2016.

[12] Colin V Murray-Wallace and Colin D Woodroffe. *Quaternary sea-level changes: a global perspective*. Cambridge University Press, 2014.

[13] E. Armynot du Châtelet, F. Francescangeli, V.M.P. Bouchet, and F. Frontalini. Benthic foraminifera in transitional environments in the english channel and the southern north sea: A proxy for regional-scale environmental and paleo-environmental characterisations. *Marine Environmental Research*, 137:37–48, 2018. https://doi.org/10.1016/j.marenvres.2018.02.021.

[14] Kurt Lambeck, Colin D Woodroffe, Fabrizio Antonioli, Marco Anzidei, W Roland Gehrels, Jacques Laborel, and Alex J Wright. Paleoenvironmental records, geophysical modeling, and reconstruction of sea-level trends and variability on centennial and longer timescales. *Understanding Sea-Level Rise and Variability*, pages 61–121, 2010. https://doi.org/10.1002/9781444323276.ch4.

[15] Giorgio Spada. Glacial isostatic adjustment and contemporary sea level rise: An overview. *Surveys in Geophysics*, 38(1):153–185, 2017. https://doi.org/10.1007/s10712-016-9379-x.

[16] NASA. Understanding Sea Level. https://sealevel.nasa.gov/understanding-sea-level/by-the-numbers/. (accessed: 30.10.2021).

[17] NASA. Sea Level Projection Tool. https://sealevel.nasa.gov/ipcc-ar6-sea-level-projection-tool/. (accessed: 30.10.2021).

[18] National Oceanic and Atmospheric Administration. . https://coast.noaa.gov/slr/. (accessed: 30.10.2021).

[19] WHATWG. HTML. https://html.spec.whatwg.org/, 2021. (accessed: 30.10.2021).

[20] ECMA International. JavaScript. https://www.ecma-international.org/publications-and-standards/standards/ecma-262/, 2021. (accessed: 30.10.2021).

[21] Python Software Foundation. Python. https://www.python.org/, 2021. (accessed: 30.10.2021).

[22] Pallets. Flask. https://flask.palletsprojects.com/en/2.0.x/, 2010. (accessed: 30.10.2021).

[23] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, 2000.

[24] OpenJS Foundation and jQuery contributors. AJAX. https://api.jquery.com/, 2021. (accessed: 30.10.2021).

[25] Google. Google Maps. https://developers.google.com/maps, 2021. (accessed: 30.10.2021).

[26] Vladimir Agafonkin. Leaflet. https://leafletjs.com/, 2021. (accessed: 30.10.2021).

[27] JSON.org. JSON. https://www.json.org, 2002. (accessed: 30.10.2021).

[28] Chart.js contributors. Chart.js. https://www.chartjs.org/, 2021. (accessed: 30.10.2021).

[29] Plotly. Plotly.js. https://plotly.com/javascript/, 2021. (accessed: 30.10.2021).

[30] ESRI. ArcGIS. https://www.esri.com/en-us/arcgis/products/arcgis-desktop/resources, 2021. (accessed: 30.10.2021).

[31] Open Source Geospatial Foundation. GeoServer. http://geoserver.org/, 2021. (accessed: 30.10.2021).

[32] Oracle Corporation and/or its affiliates. MySQL. https://www.mysql.com/, 2021. (accessed: 30.10.2021).

[33] The PostgreSQL Global Development Group. PostgreSQL. https://www.postgresql.org/, 2021. (accessed: 30.10.2021).

[34] Oracle Corporation and/or its affiliates. MySQL Spatial data types. https://dev.mysql.com/doc/refman/8.0/en/spatial-types.html, 2021. (accessed: 30.10.2021).

[35] PostGIS Project Steering Committee. PostGIS. https://postgis.net/, 2021. (accessed: 30.10.2021).

[36] PostGIS: Data Management and Queries. https://postgis.net/docs/manual-1.4/ch04.html.

[37] PostGIS Project Steering Committee. PostGIS: Building Applications. https://postgis.net/docs/manual-1.4/ch05.html, 2021. (accessed: 30.10.2021).

[38] Object–relational mapping(ORM). https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping. (accessed: 30.10.2021).

[39] SQLAlchemy authors and contributors. SQLAlchemy. https://www.sqlalchemy.org/, 2021. (accessed: 30.10.2021).

[40] Django Software Foundation. Django. https://www.djangoproject.com/, 2021. (accessed: 30.10.2021).

[41] Eric Lemoine. GeoALchemy. https://geoalchemy-2.readthedocs.io/en/latest/, 2012. (accessed: 30.10.2021).

[42] Eric Lemoine. WKBElement. https://geoalchemy-2.readthedocs.io/en/0.2.6/elements.html, 2012. (accessed: 30.10.2021).

[43] SQLAlchemy authors and contributors. SQL Inheritance Hierarchies. https://docs.sqlalchemy.org/en/14/orm/inheritance.html, 2021. (accessed: 30.10.2021).

[44] SQLAlchemy authors and contributors. SQL Relationship Patterns. https://docs.sqlalchemy.org/en/14/orm/basic_relationships.html, 2021. (accessed: 30.10.2021).

[45] CERN, IETF, W3C. HTTP. https://datatracker.ietf.org/doc/html/rfc8740, 2020. (accessed: 30.10.2021).

[46] Y. Shafranovich. Comma-separated values. https://datatracker.ietf.org/doc/html/rfc4180, 2005. (accessed: 30.10.2021).

[47] W3C. CSS. https://www.w3.org/TR/CSS/#css, 2016. (accessed: 30.10.2021).

[48] ISO/IEC. SQL. https://www.iso.org/standard/63555.html/, 2016. (accessed: 30.10.2021).

[49] Eduardo Leorri, Alejandro Cearreta, and Glenn Milne. Field observations and modelling of holocene sea-level changes in the southern bay of biscay: implication for understanding current rates of relative sea-level change and vertical land motion along the atlantic coast of sw europe. *Quaternary Science Reviews*, 42:59–73, 2012. https://doi.org/10.1016/j.quascirev.2012.03.014.

[50] Eduardo Leorri, Francisco Fatela, Teresa Drago, Sarah Louise Bradley, João Moreno, and Alejandro Cearreta. Lateglacial and holocene coastal evolution in the minho estuary (n portugal): Implications for understanding sea-level changes in atlantic iberia. *The Holocene*, 23(3):353–363, 2013. https://doi.org/10.1177/0959683612460786.

[51] IETF OAuth Working Group. OAuth 2.0 Protocol. https://oauth.net/2/, 2010. (accessed: 30.10.2021).