

POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering



**Politecnico
di Torino**

Master's Degree Thesis

**DEVELOPMENT OF A WEB APPLICATION
FOR INTERACTIVE ANALYSIS OF THE
ITALIAN OIL ENERGY SECTOR DATA**

Supervisors:

Prof. Maurizio Morisio

Prof. Ettore Bombard

Prof. Tao Huang

Candidate

Esteban Caliva

July 2021

TABLE OF CONTENTS

1. Introduction.....	1
1.1. Summary of Oil Risk Model.....	2
1.1.1. Indicators and formulas:.....	3
1.2. Data Storytelling.....	5
1.2.1. Techniques used in visual Storytelling:.....	8
1.2.2. Tools for Data storytelling.....	10
2. Requirements.....	12
2.1. Functional Requirements.....	12
2.2. Non-Functional Requirements.....	12
2.3. Use Case Diagram.....	13
2.4. Use Case Modelling.....	14
3. Design.....	16
3.1. Theoretical Background.....	16
3.1.1. Web applications.....	16
3.1.1.1. Django Framework.....	17
3.1.1.2. Backend Technologies.....	18
3.1.1.3. Frontend Technologies.....	19
3.1.2. Database Management.....	21
3.2. Application Architecture.....	22
3.3. Web Application.....	23
3.4. Database.....	24
3.5. Indicator Calculation:.....	25
3.6. Hardware.....	26
4. Implementation.....	27
4.1. Data Collection.....	27
4.1.1. Scripts.....	28
4.2. Risk Index Calculations.....	33
4.2.1. Scripts.....	34
4.3. Database.....	35
4.3.1. Scripts.....	35
4.4. Wrapper.....	36
4.5. Web Application.....	37
4.5.1. Model.....	37
4.5.2. Views.....	38

4.5.2.1. Utils.....	39
4.5.3. Templates	39
5. Validation.....	45
5.1. Data Collection Validation	45
5.2. Risk Index Calculation.....	46
5.3. Web Application	52
5.3.1. User Interface	52
5.3.2. User Authentication.....	53
5.3.3. Models and Queries.....	55
6. Conclusions.....	56
References	57

LIST OF FIGURES

Figure 1 Conceptual map of captive and sea corridor failure assessment. [1].....	5
Figure 2 Combination of data storytelling [3].....	6
Figure 3 Example of a good Storytelling	7
Figure 4 Example of bad Storytelling.	7
Figure 5 Example of different techniques applies in a plot.....	9
Figure 6 Use case Diagram.	13
Figure 7 Client-server HTTP Communication [13]	17
Figure 8 Django Architecture Pattern. [15].....	18
Figure 9 Differences between a classic web application model and an Ajax web application model. [22]	21
Figure 10 Application Architecture.	22
Figure 11 Database table structure.....	25
Figure 12 parser_alphatanker.py input file.....	28
Figure 13 parser_alphatanker.py output.	28
Figure 14 parser_piracy.py input file.....	29
Figure 15 parser_piracy.py output.	29
Figure 16 parser_wgi.py input file.	30
Figure 17 parser_wgi.py output.	30
Figure 18 parser_strait.py custom input file.	31
Figure 19 parser_strait.py output.	31
Figure 20 parser_lvh.py output.	32
Figure 21 parser_pipeline.py output.	33
Figure 22 output of the java_input_OSM.py script.	34
Figure 23 output of the calculation_route_eez_intersection.py.....	35
Figure 24 Wrapper configuration flags.....	36
Figure 25 Code Snippet of the Model	38
Figure 26 Extend Django User Class Code Snippet	38
Figure 27 Home page without user login.	40
Figure 28 Home page after user logs in.....	40
Figure 29 Login page.....	40
Figure 30 Sign Up page.	41
Figure 31 Password reset page.....	42
Figure 32 Password reset after submit form.	42
Figure 33 After clicking URL receive in the mail.	42
Figure 34 Password change successfully.....	42
Figure 35 Contact page.	43
Figure 36 Example of a Story #1.	44
Figure 37 Example of a Story #2.	44
Figure 38 Java Input Log File	47
Figure 39 Output for the corridor Sidi Kerir-Trieste.....	48
Figure 40 Output for the corridor Corpus Christi-Trieste.....	49
Figure 41 Sample of the debug file of the intersection script.	50
Figure 42 Probability of Failure of Egypt corridors calculated by the script.....	51
Figure 43 Probability of Failure of Ceyhan corridor calculated by the script.	52
Figure 44 Password mismatch error	54

Figure 45 Email or Password Log In fail55

LIST OF TABLES

Table 1 Data Analysis Tools [7] 11
Table 2 Functional requirements of the application. 12
Table 3 Non-Functional requirements of the application..... 13
Table 4 Egypt Corridor provided by EST- Lab.48
Table 5 Corpus Christi-Trieste provided by EST- Lab.49
Table 6 Probability of Failure of main Egypt Corridors provided by EST-Lab.51
Table 7 Probability of Failure of main Egypt Corridors provided by EST-Lab.52
Table 8 Database table for Users54

ABSTRACT

Nowadays common people have access to many highly detailed information even though it is difficult for them to understand the meaning behind the numerical values without a proper elaboration and explanation from an expert. In this sense, the narrative offers an optimal way to organize data by adding layers of context, facilitating understanding of project data, and closing communications gaps between social, economic, and political dimensions. Hence, data storytelling, based on the simple and synthetic narrative, is becoming an innovative way through which a complex system, such as energy systems, can be represented efficiently.

Energy statistics usually go into detail about all the variables which are taken into consideration making reports very accurate and reliable for experts on one hand, but also very long and dispersive on the other. Today, energy actors demand a scientific tool capable of providing information and key indicators that summarize the current situation in the field of energy security.

The thesis project consists of a web application development using the Django Framework, that brings compiling and interactive narratives around complex data by the means of storytelling. The core of the thesis is the application of data storytelling to the EST Oil Risk Model which aims to quantify the national energy risk related to Italian oil suppliers. This model relies on the collection and processing of many data from several sources and datasets and identifies the oil suppliers as well as the oil pathways through geo-referencing mapping of both captive and open-sea corridors.

The implemented web application, called Oil-IST (Oil Interactive Story Telling), can be accessible to all energy actors who are interested and/or involved in crude and oil products supply systems. Moreover, since Italy is strongly dependent on external oil suppliers, Oil-IST is designed to provide a science-based tool able to support and accelerate political decision-making.

1. Introduction

The world is changing towards new energies, like in every field, it is important to have accurate information to make decisions, in the context of “energy analysis”, decision-makers and stakeholders must have all the data available in a clear, synthetic, and concise way. Energy analysis considers many aspects from different fields that range from political, geographical, economic to environmental, and because of this having a holistic view is difficult.

The reports given to the stakeholders can be lengthy and, difficult to understand if there is no close connection to the dataset, making decisions slow and demanding. Within this framework, nowadays is more demanded by the energy stakeholders a scientific tool capable of providing key information and key indicators that summarize the current situation in the energy security field.

Narratives are considered the basis of human communication. Since the early years' narratives always had help humans to experience and comprehend life, it can be view as a comprehensive means of delivering information when the narrator can create a context around a sequence of events it helps the narratee to have a better conceptual understanding of the subject. In engineering sometimes narratives are overlooked as baseless and manipulative, but researchers suggest that narratives are easier to comprehend, and audiences find them more engaging than traditional logical-scientific communication. [1]

The way narratives help to have a better understanding of the topic as a whole, the approach chosen to deliver complex data in an easy and comprehensible way to energy stakeholders is data storytelling, which is a methodology for communicating information, tailored to a specific audience, with a compelling narrative by giving context and interpretation to data.

The energy sector that is cover in this work is the oil energy supply in Italy. The first step for proper political decision-making, that aims to reduce the level of risk due to the supply of energy commodities is to build a risk model, for example, in Italy, a critical problem of oil imports is due to high dependence on countries with a weak geopolitical situation and stability. For this reason, EST lab developed a new risk model specific to an oil supply that brings together numerical and georeferenced data.

The scope of the project is to take this model and automatize all the data and calculations and delivering compiling stories to the energy stakeholders by the means of a web application.

1.1. Summary of Oil Risk Model

The aim of the model is the risk assessment of the Italian oil supply, the consequence is the loss of commodity, and the hazardous event is the failure of oil supply. In particular, the supply failure may involve the maritime or the captive corridors. The risk analysis results in the quantitative estimation of risk related to each corridor and each commodity. This final value, expressed in terms of loss of energy, quantifies the security of the Italian energy supplies. [1]

The model needs input parameters to be able to properly calculate the indicators, the main goal is to calculate the probability of failure for each corridor; a corridor is understood as a group of routes that transport petroleum products between two countries. To obtain this parameter there are some complex calculations around it, for example, tracing the routes from where the commodity is extracted to the Italian port of arrival, in consequence, it involves a land route (captive) and the maritime route (sea).

The captive route is basically where the commodity is extracted and carried to the load port, the main consideration is the pipeline specifically the length and traversing countries. The maritime route is from the load port to an Italian port. The maritime route considers the length that crosses the exclusive economic zone of each country the route passes through as well if there are any chokepoints. Both routes take into consideration the geopolitical indexes and for the maritime route, the piracy index must be considered to model the risk.

The input data is taken from different data sources, the geopolitical data is taken from the “Worldwide Governance Indicator” [2] since the probability of failure depends on the geopolitical stability of the corresponding sovereign country crossing a specific EEZ. The piracy index is taken from “One Earth Foundation”, this index is important to track the criminal activities in the sea that can lead to vessel disruption and consequently to failure of supply. The presence of a chokepoint also affects the probability of failure of maritime routes. One of the most important sources of data is “Alphatanker” which is a private resource that provides the amount of petroleum product that is imported to Italian ports as well as their port of provenience.

1.1.1. Indicators and formulas:

The indicators and formulas used by the model are carefully explained in [2], in this section is presented a summary of the indicators and how their formulas.

The Worldwide Governance Indicators (WGI) were considered to calculate the reference geopolitical risk index φ_k . Indeed, φ_k is obtained by the arithmetic mean of six minor WGI indexes which range between 0 and 100:

$$\varphi_k = \sum_j^6 \frac{WGI_j}{6}$$

Criminal activities in the sea can lead to vessel disruption and consequently to failure of supply that led to the next indicator, Piracy:

$$\varphi'_k = \frac{\sum_j^6 WGI_j + \eta_k}{7}$$

The presence of a chokepoint affects the probability of failure of maritime routes. Especially in the case of strait crossing, many ships are forced to pass through a narrow channel. Thus, chokepoint disruption, defined as the ships inability to cross it, may be caused mainly by two factors:

1. Geopolitical risk of the coastal countries closes to the chokepoint (φ_k)
2. Piracy activity (η_k)

Each coastal country is characterized by a new aggregated parameter which summarizes in one single value both geopolitical risk index and piracy index.

$$\bar{\varphi}' = \frac{\sum \varphi'_k}{K}$$

Where:

- K total amount of basin countries.
- φ'_k maritime geopolitical risk of the K country.

The probability of chokepoint failure was defined as the product between two factors: L_{cp} directly connected to the political stability of coastal countries close to the chokepoint and α' which reflects the intrinsic vulnerability of a chokepoint due to its physical characteristic.

$$\xi_{cp} = L_{cp} * \alpha'$$

Where:

- ξ_{cp} is the probability of chokepoint failure.
- L_{cp} is the likelihood of failure.
- α' is the vulnerability index.

Weighting factor γ_k , so the contribution to the overall probability of failure of the single branch is proportional to its length.

$$\gamma_k = \frac{l_i}{L_{tot}}$$

Once defined the probability of failure related to both the cross a chokepoint and cross EEZ or international water, then, the overall formulation of the open-sea probability of failure is obtained:

$$\xi_{route} = 100 * \left[1 - \prod \left(1 - \frac{\gamma_k * \varphi'_k}{100} \right) \right]$$

$$\xi_{OpenSea} = 1 - [(1 - \xi_{cp}) * (1 - \xi_{route})]$$

Like the maritime route, also the captive corridor probability of failure depends on the geopolitical stability of the crossed country and the weighting factor.

$$\xi_{captive} = 100 * \left[1 - \prod \left(1 - \frac{\gamma_k * \varphi_k}{100} \right) \right]$$

A conceptual map of how the corridors are modeled by their sea and captive branches are presented in figure 1.

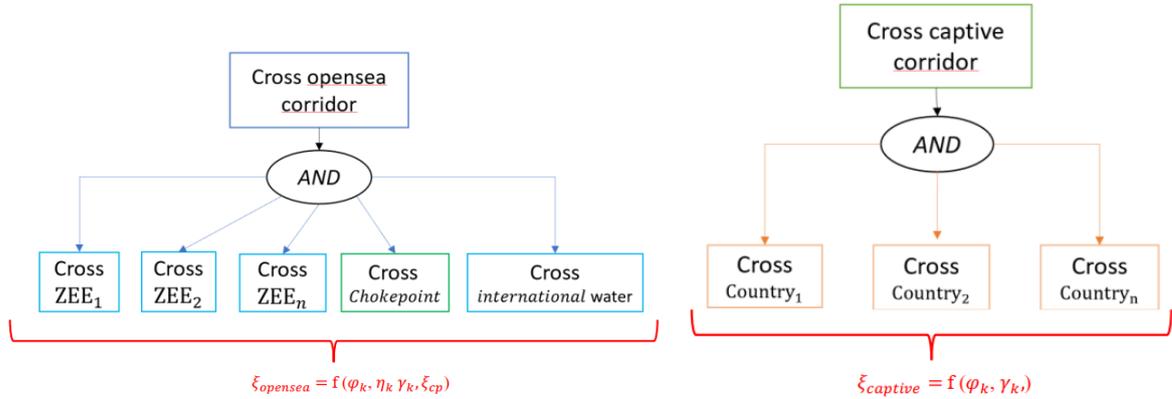


Figure 1 Conceptual map of captive and sea corridor failure assessment. [1]

After modeling each corridor, it is possible to calculate the energy risk level quantification for each corridor or by providing country. The overall probability of failure of the entire corridor is calculated with the following function which combines $\xi_{opensea}$ and $\xi_{Captive}$:

$$\xi_i = 1 - [(1 - \xi_{OpenSea}) * (1 - \xi_{Captive})]$$

Finally, the corridor risk can be calculated by multiplying the probability of failure with the corresponding energy transported through the corridor. The total provisioning risk is obtained by the summation of all the single corridor risks.

$$R_{ext_i} = \xi_i * E_{i*}$$

$$R_{ext} = \sum_{i \in I} R_{ext_i}$$

1.2. Data Storytelling

Engineering always involves a large amount of experimentation and subjective decision-making. Thus, it is important to document the story behind the computation of results, reporting alternative solutions, and explaining their limitations.

Storytelling through data is often overlooked as a trivial task, but stories provoke thought and bring inside that could not have been understood or explained before, in other words, is the process of transforming data-driven analyses into a variety of media such as textual explanations, graphs, forms, interactive visualizations, code segments to influence a business decision, strategy, or action by utilizing analytical information that, ultimately, turn into actionable insights.

As the result of the lacking context in the data sets problematic, is born the necessity of having a story that guides the user through the data set, for example, where this data come from, how and the time it was taken, the reason, importance, and value. This creates better comprehension therefore is possible to give pure raw data a meaning especially for a non-expert in the field of where the data is taken.

The three aspects of data storytelling are narrative, visuals, and data. These aspects create a good data story and present the data in a form that can create audience engagement and accurate problem-solving, so it represents the complex datasets and separates the trends and patterns which are not seen in the spreadsheets.

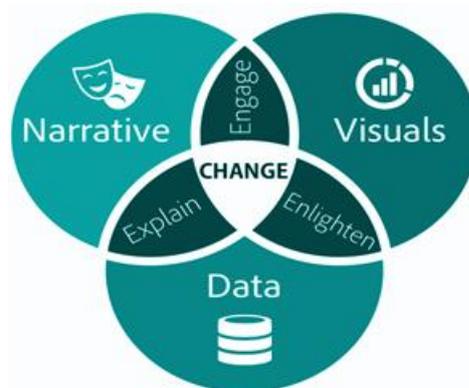


Figure 2 Combination of data storytelling [3]

The importance of data storytelling lays in the creation of a compelling story that can influence the audience and provide critical information to make the best decision possible.

To get a better understanding of the importance of a good data representation one example of how a good narrative is shown in the visualization of the plot and how a bad example looks like according to [4]:

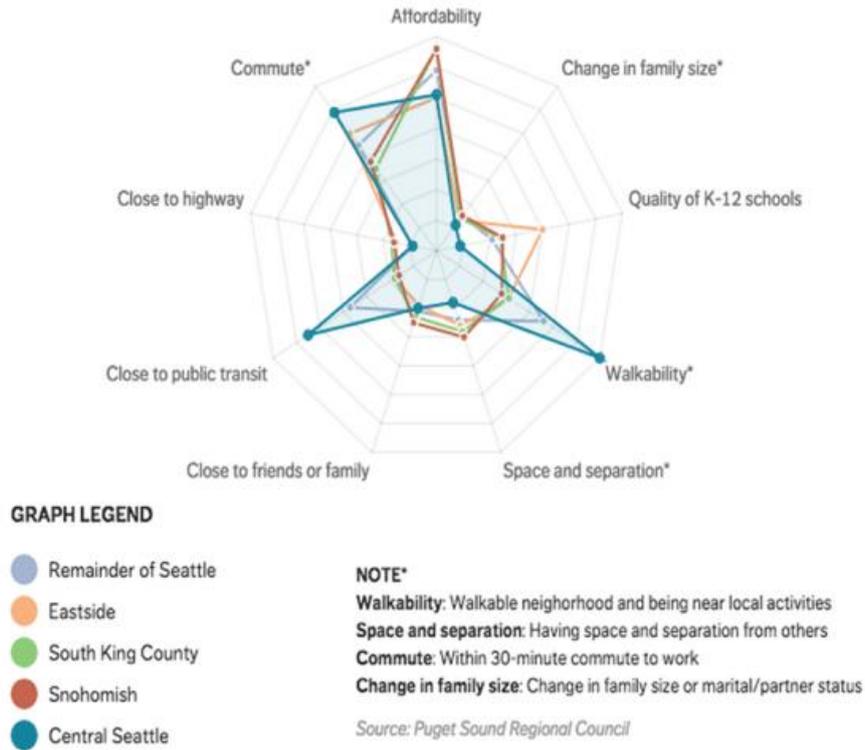


Figure 3 Example of a good Storytelling

This graph was taken from the “The Seattle Times” it represents data in a very concise way, it is possible to infer that most people have nearly the same universal concerns, also it is possible to see with easy correlations for example how people that care about Walkability are far more interested in proximity of the public transport.

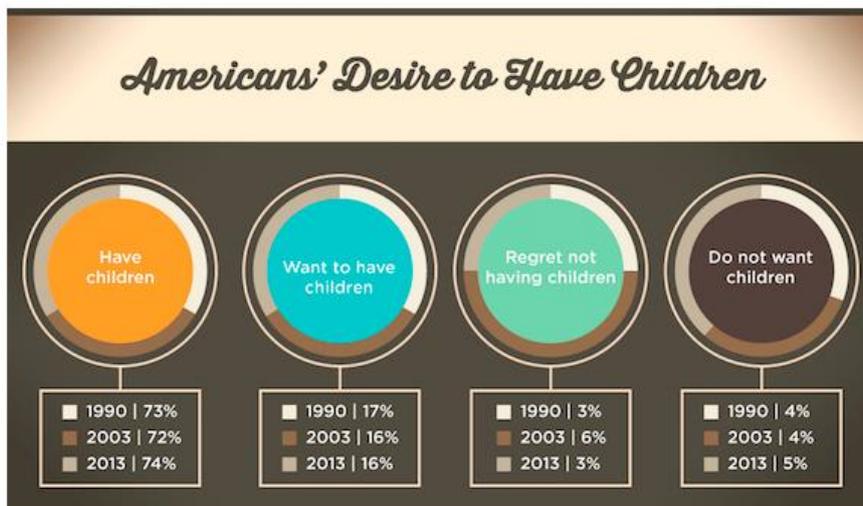


Figure 4 Example of bad Storytelling.

This graph was made by “Sparefoot” it is intended to show how American society is changing over time concerning household living arrangements. The main questions are: Is it easy to read? Does this pie chart a better representation? Can the reader understand it clearly?

For representing a change over time, it is better to use a line or bar chart, but the major problem is that the three main data points are not part as a whole but they have been presented as if they are although it does not combine a 100% of something, leading to miscommunicating the message.

1.2.1. Techniques used in visual Storytelling:

In this section is presented techniques for how the data should be visualized and presented according to [5]:

a) Communicating Narrative and Explaining Data:

One of the simplest techniques uses long textual narratives to express the key points to help to explain and communicate the author's message. The communication of the narrative and explanation of the data not necessarily are confined to text, multiple forms of media can be used such as video, audio narration.

b) Linking Separated Story Elements:

Data-driven stories often contain multiple story elements in different text forms or charts, making a connection between elements is crucial to explain the different aspects of the story. There are different techniques for properly linking the elements:

- Linking elements through interactions: Story collection where the interaction of one visualization is mapped to change another.
- Linking elements through colors: usually accomplish this through a consistent color mapping between attributes that appear in multiple visualizations.
- Linking elements through animations

c) Enhancing Structure and Navigation:

This technique is closely tied to the author specifying ordering where the author imposes the order in which the data will be displayed, often provide a navigation aid.

- Next/Previous buttons: Uses buttons to order the storyline.
- Scrolling: Scrolling triggers changes in the visualization as the story unfolds in front of the user.
- Breadcrumbs: a common breadcrumb style uses dots to represent each slide or scene and each dot acts as a shortcut, taking the reader to the associated slide.
- Section header buttons: Use titles in each of the scenes, and these titles are placed at the top of the story to help control movement through the narrative.
- Geographic map: Interactive map to help the user to jump in any state.

d) Providing Controlled Exploration:

Enables the reader to explore the story as it significantly changes the data and visualization displayed. One of the risks is that the data could not be consistent with the surrounding narrative.

Dynamic queries: Gives the possibility of interaction by changing different subsets of the data.

Embedded exploratory visualization: Used when the authors have included a more exploratory visualization, often defined by a large amount of user interaction.

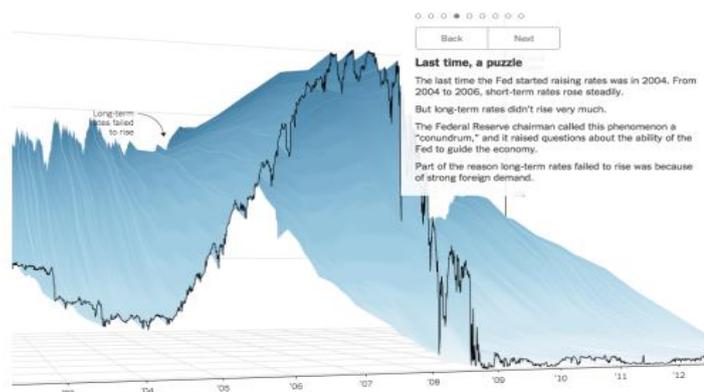


Figure 5 Example of different techniques applies in a plot.

This plot was taken from “The New York Times”, it applies different techniques, such as textual narrative, dot style breadcrumbs besides this it present annotation in each point of the chart the user points to as well as a 3D rotation.

Other good practices ANALYTICS VIDHYA [6] consider are important to deliver good data-driven storytelling, that sometimes can be obvious, are:

- Always label your axes and give the heading of your plot.
- Use legends where necessary.
- Use colors that are lighter on the eye and in proportion.
- Avoid adding unnecessary detail to your visualization like backgrounds or themes that do not allow good readability.
- Only a point can be used to simultaneously encode two quantitative values based on a horizontal and vertical location.
- Never use points for visualization if you are doing time series encoding.

1.2.2. Tools for Data storytelling

A myriad of analytical techniques is employed in generating insights from a large amount of data for storytelling. (Manyika, 2011) identified several of these techniques including association rule mining, classification, cluster analysis, data fusion and integration, machine learning, genetic algorithms, natural language processing, network analysis, pattern recognition, predictive modeling, regression analysis, sentiments analysis, spatial analysis, and time series analysis. Some of the tools implementing these analytical techniques include R, Python, Microsoft Excel, IBM-SPSS, Tableau, NodeXL, Google Fussion Tables, RapidMiner, and Knime Analytic.

Purpose	Description	Tools
Data Analysis	Tools and techniques used for exploring and analyzing data.	Spreadsheet software SPSS SAS Stata Python (Panda) R (Object Graph)

Data Visualization	These are tools and techniques used for user-facing data and information visualization, in both static and interactive forms.	Tableau Public R (when used to data visualization) Google fusion tables Brackets Highcharts Linkurious JavaScript libraries (D3.js and Raphael)
Map Visualization	Visualization tools are specifically concerned with geospatial data, and maps.	ArcGIS, MapBox Mapper Open Street Maps API Google maps
Databases	Database management tools, used for storage and retrieval of data of different types.	MySQL SQL Server SQL Base PostgreSQL MongoDB Neo4j
Data Preparation and Wrangling	Tools that are used for data preparation, data cleaning, and data transformation or wrangling, used for pre-processing data for suitable for data analysis.	OpenRefine Mr Dataconverter Wigle Nitro PDF Tabula
Data Scraping	Tools that are used for scraping or collecting data from webpages, PDFs, or scanned documents.	Imacros HTTrack Omnipage Nokogiri

Table 1 Data Analysis Tools [7]

2. Requirements

“The software requirement specification is the basis for software development. It describes the functional and non-functional requirements and includes a set of use cases that describe user interactions that the software must provide” [8].

The main requirements for this platform can be divided into three parts, the automatic calculation of the risk indexes after the input files are given to the system, storage and distribution of the risk model data, and the creation of compiling narratives to deliver the processed data to the end-users.

2.1. Functional Requirements

The functional requirements, summarize what operations should the user of the application will be able to perform define what the system does or must not do. The functional requirements contain all the technicalities that surround the project, basically how it behaves in matters of manipulation, processing, integration, and migration of the data, plus security requirements and performance.

Name	Description
FR01	Allow the admin to parse the new data once it is provided.
FR02	Allow the admin to calculate the risk indexes once the data is provided.
FR03	Allow the admin to insert new data into the database.
FR04	Allow the user to log in to their account.
FR05	Allow the user to sign up on the platform.
FR06	Allow the user to recover its password.
FR07	Allow the user to select stories.
FR08	Allow the user to fill a contact form for new requests and enhancements.
FR09	Allow the user to interact with the stories.

Table 2 Functional requirements of the application.

2.2. Non-Functional Requirements

Non-functional requirements specify the quality attributes of the system. These requisites do not affect the basic functionality of the system.

Even if the non-functional requirements are not met, the system will still perform as it is supposed to.

Name	Description
NFR01	The application will be developed in Django
NFR02	Data and user information will be stored in PostgreSQL.
NFR03	The application will be scalable, add new stories.
NFR04	The application must be available to all users.
NFR05	The database paradigm must be suitable.
NFR06	The stories must load in less than 100 seconds.

Table 3 Non-Functional requirements of the application.

2.3. Use Case Diagram

A use case diagram is a behavior diagram that models the functionality of the system in the form of actors interacting with a system. The use cases are the actions and/or services the system needs to perform. The system is something to develop and operate on, as a website. The actors are users or entities operating defined by roles within the system. [9]

The actors for this project are the Users, the system administrator, and the storyteller in figure 6 it is possible to observe the interactions of the actors with the system.

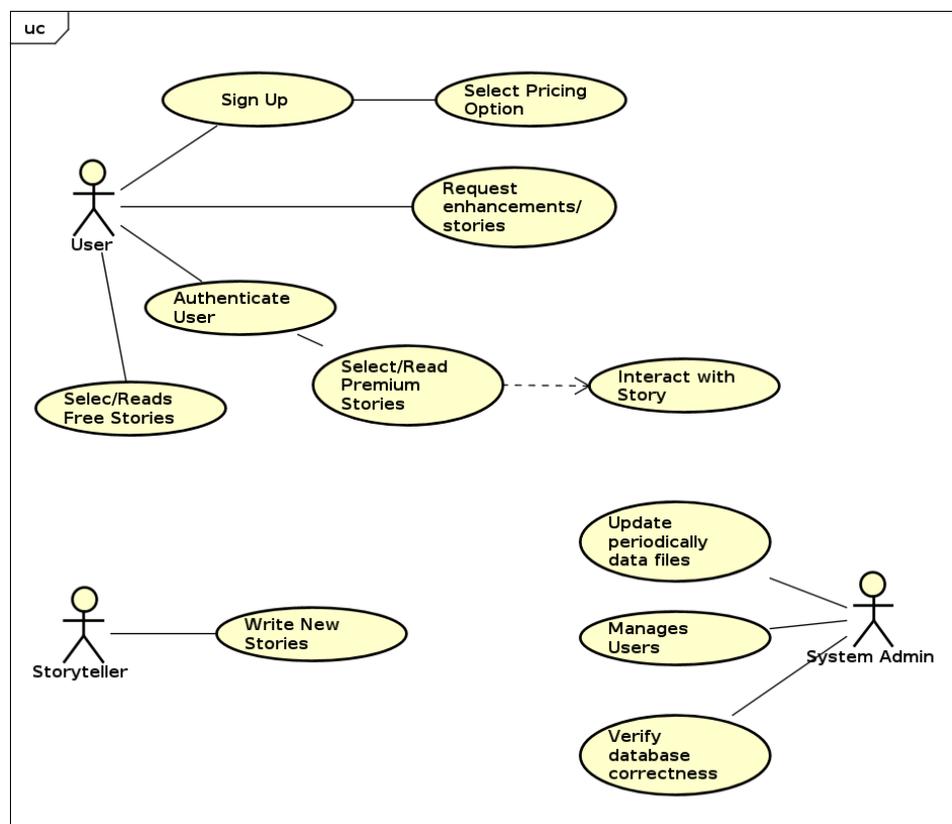


Figure 6 Use case Diagram.

2.4. Use Case Modelling

“A use case is a written description of the list of actions or event steps of how the user will perform tasks, typically the interactions between a role and a system, to achieve a goal. It outlines a user’s point of view and a system’s behavior as it responds to a request.” [10]

Scenario #1: User Sign Up – Create Account

Pre-Condition	Account does not exist.
Nominal Scenario	The user goes to the Sign-Up page. The user inserts the requested data. The system verifies the correctness. The system sends a very email to the user. The user clicks on the link inside the mail.
Post-Condition	The account was added to the system.

Scenario #2: User Select a Story and Interact

Pre-Condition	-
Nominal Scenario	The user selects a story of interest. The user interacts with the story.
Post-Condition	-

Scenario #3: User Request a Story or Enhancement

Pre-Condition	Users must be Log in
Nominal Scenario	User populates all the fields required fields.
Post-Condition	Email is sent to the web admins.

Scenario #4: System Admin Update data files

Pre-Condition	Have administrator rights
Nominal Scenario	Admin downloads new real-world data. Configure the wrapper script depending on the flow. Run the wrapper script. Check for any issues.
Post-Condition	Files updated on the system.

Scenario #5: Storyteller creates a story.

Pre-Condition	Have access to request mail inbox
Nominal Scenario	Read user suggestions. Create new stories. Add stories to the system.
Post-Condition	A new story was added to the system.

3. Design

This section consists of the theoretical background of the components of the application as well as the technologies that were used, the architecture of the platform itself, and the database.

3.1. Theoretical Background

In today's diverse world of software, there are countless ways to design and implement a piece of software or application. There are many possibilities and options to choose from, starting from the choice of the application type, followed up by choice of environment, programming languages, technologies, and frameworks.

In this project, the focus is to deliver compiling and interactive narratives to the energy stakeholders of the oil field, independently of the platform or system they are using, in this way it is possible to expand the usability of the platform. For this matter the web application route was chosen; it must be acknowledged that recent surveys show that smartphone users are becoming less and less motivated to download extra applications, as not to bloat their own devices [11]. The same can be said with desktop applications.

3.1.1. Web applications

"A Web application can be defined as a software system that is accessible over the web. It uses technologies for the web and strives to use standard technologies where feasible." [12] Web applications are computer programs that allow website visitors to submit and retrieve data to/from a database over the internet using a web browser.

Nowadays web applications are becoming more and more popular because of the accessibility of an internet connection, helping to share applications easily with the customers.

The typical flow of a web application according to [13], starts when the user triggers an HTTP request to the web server over the internet, the browser sends the request, then it reaches the dedicated hardware and then the HTTP server sends a response containing the information that was requested. When HTTP protocol is used a certain set of rules is always followed:

- The server can only respond to the request that was sent by the client. It cannot send requests to the browser.

- The server must send a response to every incoming request, at least a response that contains an error message.
- Every HTTP request must contain a URL address that indicates the server and path that this request should be sent to.

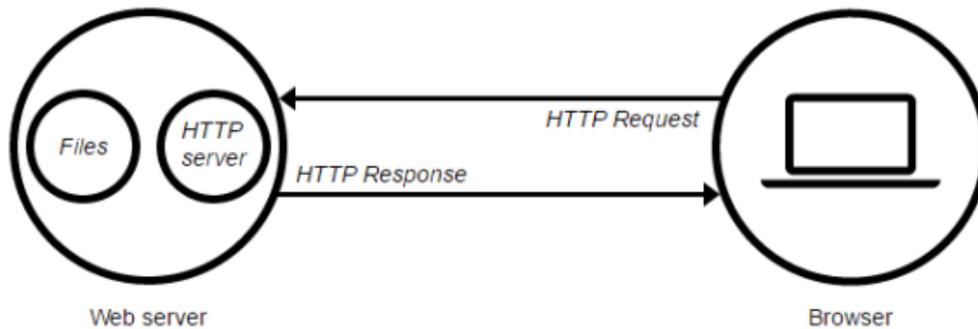


Figure 7 Client-server HTTP Communication [13]

The main programming language used for this project is Python, used for all the data collection and analysis with strong usage of Pandas and NumPy libraries. The web application was built using the Django framework in combination with HTML, JavaScript, jQuery, CSS, and Bootstrap for structure, interactivity, and styling.

Most of the modern applications are coded using a framework, there is a huge variety of frameworks built with different programming languages, a framework or software framework, is a “platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform and a standard way to build and deploy applications”. [14]

3.1.1.1. Django Framework

Is a high-level Python web application framework that enables the rapid development of web applications. It achieves so with a pragmatic, much cleaner design and is also easy to use. Thus, is very popular among web developers.

It was originally developed to avoid writing new Web applications entirely from scratch.

Features:

- Stability
- Excellent Documentation
- High Scalable

- Utilizes SEO (Search Engine Optimization)

Architecture:

Django uses a slightly modified version of the MVC (Model-View-Controller). This kind of architecture helps to separate the input, processing, and output of the application, called MVT (Model-View Template) being the main difference that Django takes care of the controller part leaving the developer the template.

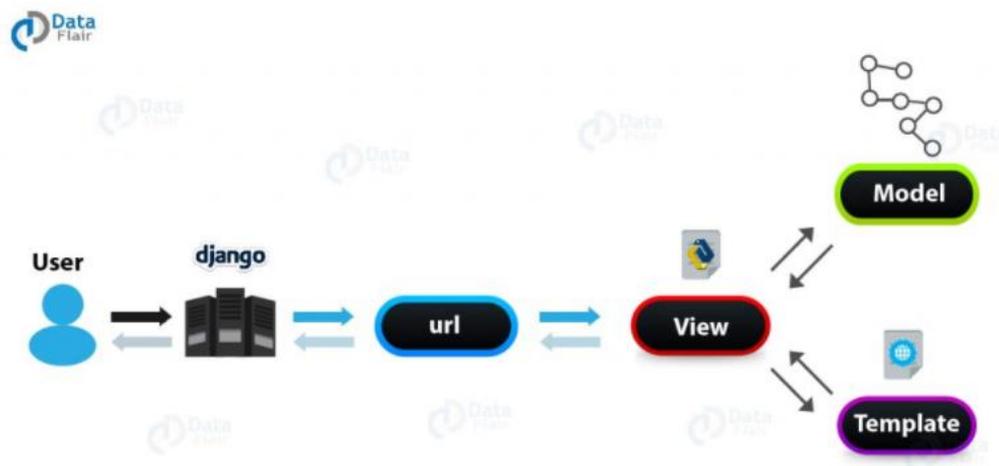


Figure 8 Django Architecture Pattern. [15]

Model: Acts as the link between the website interface and the database, it is the object which the logic for the application's data domain.

View: Communicate with the database via the model, handle the user interaction and select the view according to the model.

Template: Contains the User Interface logic, it contains everything the browser renders.

URLs: is a URL mapper used to redirect HTTP requests to the appropriate view based on the request URL.

3.1.1.2. Backend Technologies

All the backend is coded in python language.

Python: is a popular high-level dynamic programming language. It is powerful and fast. This language has some interesting features such as:

- Intuitive object orientation.
- Strong introspection capabilities.
- Readable and very clean syntax.
- Full modularity, supporting hierarchical packages.
- Extensive standard libraries and third-party modules for every task.
- Very high-level dynamic data types.

Python is also used in a huge range of domains including Web Applications and has good documentation and community.

The main libraries used for this application are:

Pandas: “is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures, and data analysis tools for the Python programming language” [16].

NumPy: “is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects, and an assortment of routines for fast operations on arrays” [17] .

3.1.1.3. Frontend Technologies

Frontend technologies are responsible for building the part of a web application that is in direct contact with the end-user.

The application is built on core web technologies. The use of HTML for creating what is contented of pages, Bootstrap CSS framework for styling, and JavaScript supported by JavaScript library jQuery for enabling the interactivity of the web pages.

HTML and CSS are not programming languages. HTML is a markup language; its main function is to create and construct documents viewed inside a web browser. Every single website displays its markup using HTML. CSS is a style sheet language and used for the styling content and document layout.

Bootstrap: “is a free and open-source front-end development framework for the creation of websites and web applications. The Bootstrap framework is built on HTML, CSS, and JavaScript to facilitate the development of responsive, mobile-first sites and apps” [18].

JavaScript (JS): is a scripting language that was originally developed by Netscape Communication Corporation for Web use in 1995. JS should not be confused with Oracles’ Java programming language. These languages are unrelated and have different schematics. JS has a syntax that is influenced by

the C programming language, and it also copies many names and naming conventions from Java. JavaScript is a lightweight object-oriented, prototype-based, multi-paradigm scripting language that is type-safe and dynamic. JS uses an interpreter and supports also object-oriented, imperative, and functional programming styles. JavaScript has a standard called ECMAScript. The modern Internet browsers are updated and fully support the ECMAScript v5.1, as of 2012 [19].

jQuery: “fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility” [20].

AJAX (Asynchronous JavaScript and XML): “is a technique that helps creating fast and dynamic web pages. Ajax allows web pages to be updated asynchronously by exchanging data with the server. Ajax makes it possible to update parts of a web page without reloading the full page” [21].

Ajax opened new ways of providing content on web applications. Earlier it was difficult to provide much information at once because an application could not react to user inputs effectively. With Ajax, a single web application page can provide the same information that earlier needed page reloading and multiple pages.

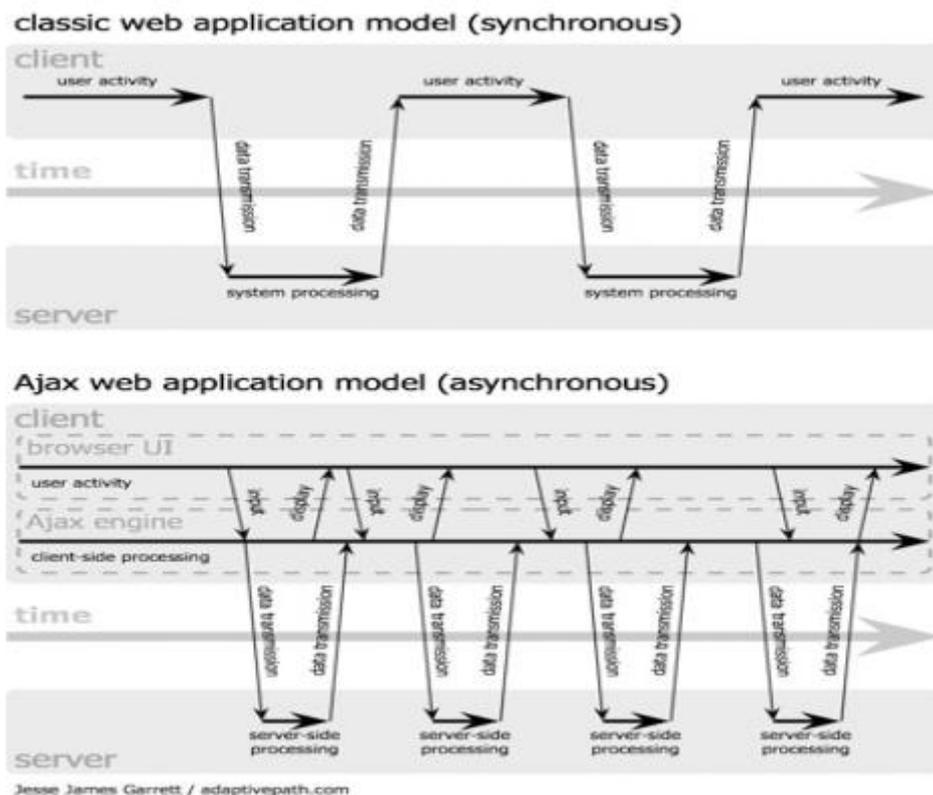


Figure 9 Differences between a classic web application model and an Ajax web application model. [22]

3.1.2. Database Management

Nowadays in every modern project, a database is indispensable, a database contains the collection of data organized and structure for easy store and access of the data. The database is designed to manage and substantial quantities of information. Management of data involves both defining structures for storage and providing mechanisms for the manipulation of information.

In addition, the database system must guarantee the safety of the information stored, despite system failures, crashes, or persons trying to access it without permission. Databases also have disadvantages, such as slow data searching retrieval, difficulty in maintaining, which make it crucial to have a database management system.

“A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data” [23]. This is a collection of related data with an implicit meaning and hence is a database. A database in essence is nothing more than a collection of information that exists over a period of time. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both

convenient and efficient. By data, we mean known facts that can be recorded and that have implicit meaning.

PostgreSQL: Powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. [24]

3.2. Application Architecture

As it was stated before the solution consists of a web application as an easy and efficient way to share data stories to the energy stakeholders.

The first problem was to automatically calculate the risk indexes from the data taken from the internet as it was needed to track all the parts of the route with their economic exclusive zones, commodity, and intake for every corridor.

The second challenge was to develop a database architecture capable to easily retrieve all the data needed for the calculation and display.

After having the database ready with all the information needed to create appealing data stories, the web application was developed to connect bring all the data to the users, in other words, the Django back-end will take the data from the database transform it, and deliver it to the users in form of stories by the front-end.

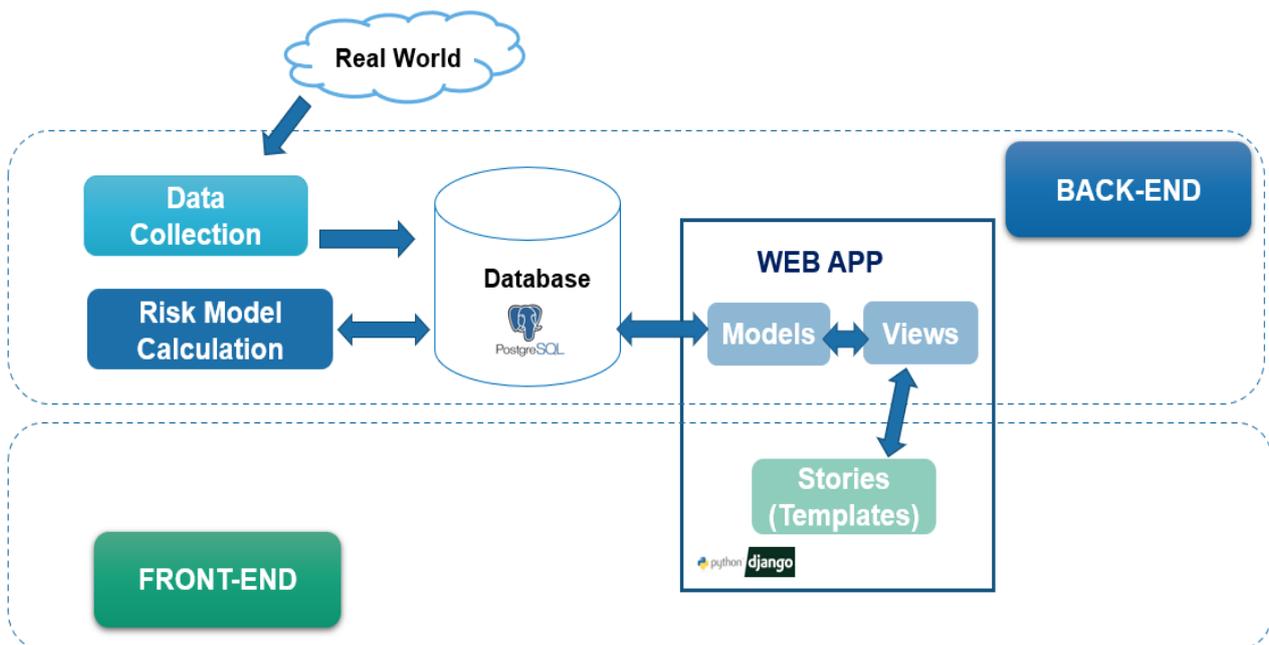


Figure 10 Application Architecture.

In figure 10 it is possible to observe the different parts of the applications, having the input coming from real-world to the “Data Collection” module that will transform and store in the database information needed for further use, the “Risk Model Calculation” oversees retrieving the data needed to calculate each risk index and save it in the database.

The database is used as a link with the risk indexes calculated with changing real-world data and the Django Framework. Django is divided into three modules by architecture design, so the “Models” role is to map the database tables and make them visible to the “Views” which contains the logic and behavior of the application, and the “Templates” that contains the presentation layer that means it contains the user interface logic.

3.3. Web Application

Functional:

- **Interactivity:**
The platform must provide the users a certain level of freedom in which they can change locations, time ranges, and units of measurements. Also, it will be possible to interact with the plots provided in the stories, for example, download the plot, zoom in, zoom out high and turn on and off parts of the plot.
- **Client Authentication:**
The platform must provide a robust client authentication where the clients can sign up and log in, with email verification as well as recover the password in the case they forgot it.
- **Request a Story:**
The users can also request new stories as they consider necessary for their interest and enhancements (different data visualization or new data) for the pre-existing ones.

Non-Functional:

- **Accessibility:**
The platform must be accessible to all users.
- **Scalability:**
The application must be expandable to handle new kinds of content and new stories can be written.
- **Usage:**
The UI must provide an enable the users to surf through the stories and click the one of their interest.

3.4. Database

The database is needed for data management, and it must be fast and reliable and plays a crucial role in the applications as it serves as a link between data collection and risk calculations with the web application.

The table structure was planned to have synergy with the previous modules making easily retrieve data without passing from more than three tables. In figure 11, it is possible to observe that the corridor is the basis of the structure, and the other tables are the characterization of each corridor as they hold the sea branch, captive branch information, corridor intake has the information that came from Alphatanker with the intake and dates per corridor, another important table is the corridor failure as it holds the calculation needed for getting the risk this is characterized by the corridor, year, pipeline and depends on the commodity has two types of failures.

Functional:

- Access:
Write new data to the database it is only possible by the system administration or an authorized person with their credentials.

Non-Functional:

- Scalability:
Content stored in a database must be scalable. The structure of the database schema and tables must be easily expandable.
- Data Integrity:
The database must avoid storing duplicate records. This keeps the size of the database as small as possible.
- Performance:
Database queries must have a reasonable response time.
- Maintenance:
The database must be maintainable. Should be kept as simple as possible.
- Suitability:
The selected database paradigm must work with the selected web application technique(s). Needed queries should be easily implemented into code.

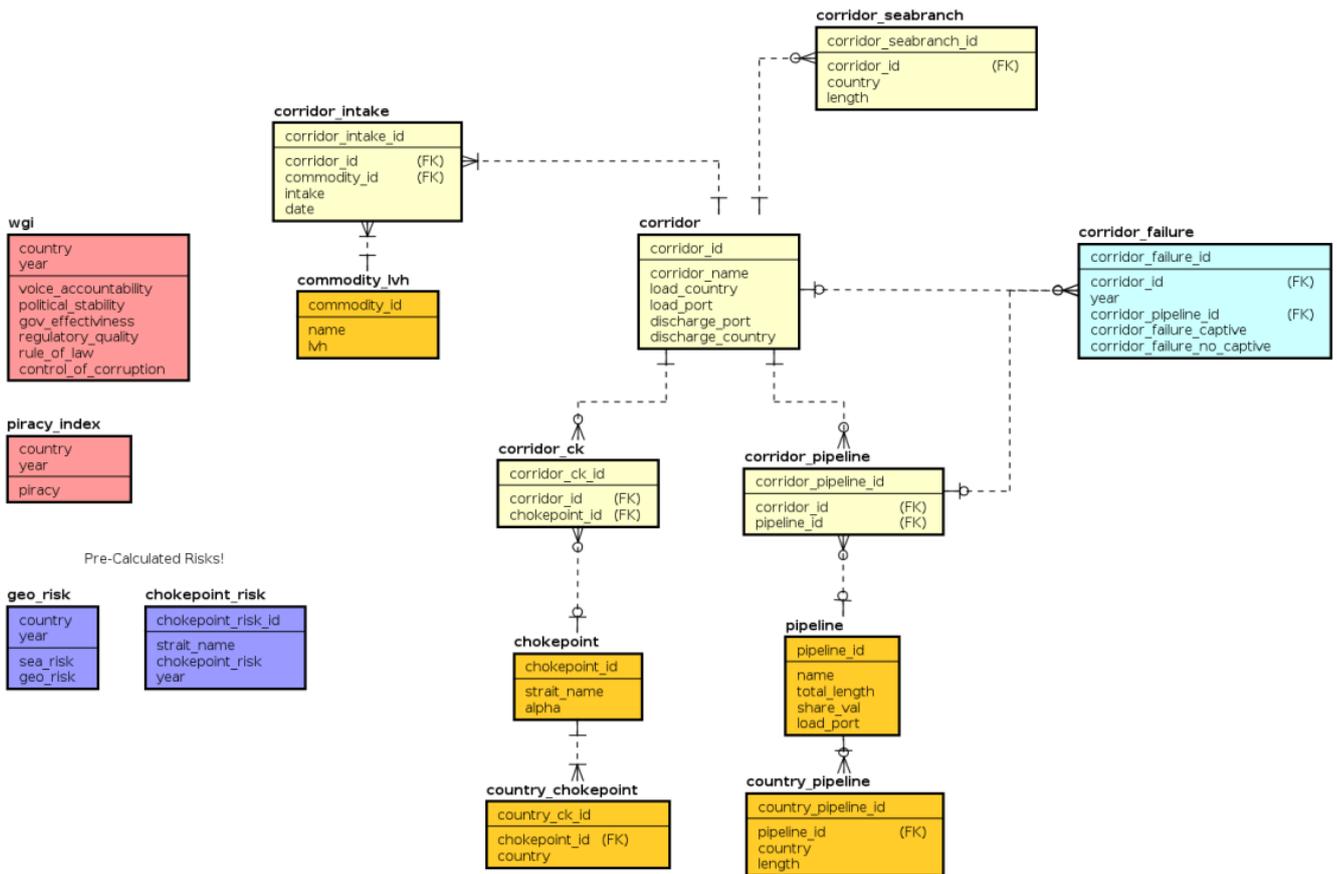


Figure 11 Database table structure.

3.5. Indicator Calculation:

This is where the data from real life is taken and transform to feed the risk model. This part is only managed by the system admin and no user has access to this.

Functional:

- **Route Tracing:**
The script must provide a way to calculate the maritime sea route from the load port to the discharge port and the length of economic zones of each country it crosses as well as if it passes through chokepoints.
- **Parse:**
It must provide a way to read and transform the data coming from Alphatanker, World Bank, and One Earth Foundation.
- **Flow Control:**
The script must provide an easy way to manipulate the flow of the program when new data is input and store it in the database.

3.6. Hardware

The platform will be hosted on a Linux server with Ubuntu OS, the CPU is a Skylake Server with 32 cores and 16GB of RAM.

4. Implementation

In this section, specific parts of the application implementation will be shown and discussed. It is divided into four major parts, data collection, calculation of the risk model, database, and web application.

4.1. Data Collection

Data Collection is the first part of the project is in charge of getting all the information needed for calculation the oil risk indexes with data from the real world. Most of these data have to be transformed to be correctly stored in the database, for this purpose six scripts were written. Most of the scripts shared the same characteristics as all of them are written in Python and uses the Pandas library.

The main sources and input files are:

- **Alphatanker:**
A private source that provides information about dates, intake in tons, commodity, and the port of origin and destination. This information is also used to track which are the routes needed to be calculated.
- **One Earth Foundation:**
Open-source which provides the piracy index by year.
- **World Bank:**
Open-source which provides the “World Governance Indicators” that consists of six indicators about the geopolitical situation of each country.
- **Strait Information:**
Custom file which provides information about the straits of interest with their basin countries as well with the “alpha indicator” that varies depending on the navigable length and width of the canal.
- **Commodity:**
Custom file which provides the information about the low-heat value of each petroleum derivate. The commodities of interest were taken from the Alphatanker database.
- **Pipelines:**
Custom file, because of the difficulty of tracking the land route of the commodity at the point of the development, the EST lab already have some pipelines and routes pre-calculated for the most important corridors, for the information missing a rough estimation with the surface of the country in square meters was done.

4.1.1. Scripts

- parser_alphatanker.py:

This script receives as input an excel file generated from the database of “Alphatanker” with the required information (Load Port, Discharge Port, Commodity, and Date). The file downloaded by alphatanker is shown in figure 12. It has a multi-header to differentiate between the different discharge ports, commodities, and dates, for this matter the file was parse with the “read_excel” method that allows using a multi-header format and transform it into a DataFrame. The DataFrame needs to be cleaned from the Non-a-Number fields that come from the empty spaces, and drop the “Total” column since there is no use to it, then the frame is flatter to produced output with date granularity as it can be seen in figure 13.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1																		
2		Augusta																
3	Load Port	Fuel Oil					Total (Fuel Oil)	Gas Oil	Total (Gas Oil)	Jet Fuel		Total (Jet Fuel)	Naphtha			Total (Naphtha)		
4		06/09/2020	09/09/2020	28/09/2020	11/10/2020	18/10/2020		18/09/2020		14/10/2020	31/10/2020		03/09/2020	11/10/2020	08/11/2020		06/09/2020	*3/09/20
5	ABOT - Al Basra Oil Terminal (ex Al Bakr)																	
6	Agio Theodoroi																	
7	Alexandria																	
8	Alaga						34172	34172			28193	28193						
9	Antwerp																	
10	Arzew																	
11	Bejaia																	
12	CPC Terminal																	103200
13	Cardagena (Esp)																	
14	Casibac																	
15	Corpus Christi																	
16	Eleusis																	
17	Marsa el Brega																	
18	Marsa el Hamig																	
19	Moogstadi																	
20	Moskossytsk		41090	46803			87893	30989	30989									
21	Port de Bouc																	
22	Rajmorsk																	
23	Sidi Keer																	
24	Stussnaes					31534		31534										
25	TEN Field																	
26	Taragona																	
27	Tuapse																	
28	Vysotsk	91316						91316										
29	Yanbu																	
30	Zueitina																	
31	Grand total	91316	41090	46803	31534	34172	244915	30989	30989	32064	28193	60257	31368	37512	30990	99870	103200	90

Figure 12 parser_alphatanker.py input file.

	Load Port	Discharge Port	Commodity	Date	Intake
0	ABOT - Al Basra Oil Terminal (ex Al Bakr)	Milazzo	Non Heat Crude	29/09/2020	131610.0
1	ABOT - Al Basra Oil Terminal (ex Al Bakr)	Milazzo	Non Heat Crude	11/11/2020	135361.0
2	Agio Theodoroi	Gaeta	Ultra Low Sulfur Diesel	04/10/2020	30596.0
3	Agio Theodoroi	Marghera (Venice)	Ultra Low Sulfur Diesel	20/10/2020	30294.0
4	Alexandria	Augusta	Jet Fuel	31/10/2020	28193.0
..
72	Tuapse	Sarroch (Porto Foxi)	Fuel Oil	19/09/2020	33717.0
73	Vysotsk	Augusta	Fuel Oil	06/09/2020	91316.0
74	Yanbu	Fiumicino	Gas Oil	03/09/2020	94860.0
75	Zueitina	Sarroch (Porto Foxi)	Non Heat Crude	08/11/2020	76182.0
76	Zueitina	Sarroch (Porto Foxi)	Non Heat Crude	15/11/2020	82599.0

Figure 13 parser_alphatanker.py output.

- parser_piracy.py:

A simple script that parses the excel input file shown in figure 13 and gets the “Piracy and Armed Robbery” to create a DataFrame with the Country, Year, and Piracy Index as shown in figure 14.

	A	B	C	D	E	F	G	H	I	J	K
1	Country	Overview	Rule of Law	International Cooperation	Maritime Enforcement	Blue Economy	Fisheries	Coastal Welfare	Piracy and Armed Robbery	Illicit Trades	Maritime Mixed Migration
2	Algeria	65	43		81	66	39	62	68	100	72
3	Angola	55	35		58	48	46	62	56	71	64
4	Bahrain	71	44		77	75	58	40	93	100	79
5	Bangladesh	52	40		88	69	36	53	50	19	65
6	Benin	50	66		71	47	31	37	63	12	48
7	British Indian Ocean Territories	80	89		75	90	41	61	90	100	91
8	Brunei	65	54		56	55	61	37	93	70	88
9	Cabo Verde	69	73		94	39	60	60	79	91	62
10	Cambodia	57	38		73	47	47	26	77	90	70
11	Cameroon	44	32		75	60	37	40	37	19	43
12	Comoros	49	45		50	22	42	42	71	92	32
13	Congo REP	55	33		77	42	49	46	72	47	75
14	Cote d'Ivoire	52	51		83	40	36	45	54	51	53
15	Djibouti	58	47		56	48	40	34	73	99	75
16	DRC	41	23		44	29	21	26	45	63	71
17	East Timor	65	62		75	39	39	47	74	100	87
18	Egypt	61	37		60	74	57	57	49	100	68
19	Equatorial Guinea	49	22		50	50	47	52	60	16	80
20	Eritrea	53	26		40	35	32	47	67	100	89
21	Mayotte	77	86		75	80	39	63	89	92	79
22	Gabon	56	43		58	46	49	57	79	35	76
23	Ghana	57	68		100	57	41	62	64	11	48
24	Guinea	54	41		94	37	24	54	47	78	58
25	Guinea-Bissau	56	36		77	40	27	65	68	88	44
26	India	58	62		88	74	62	56	44	24	48
27	Indonesia	57	63		88	66	66	61	58	0	60
28	Iran	62	39		50	76	52	56	74	99	58
29	Iraq	58	32		77	69	26	34	53	100	79
30	Israel	67	70		21	78	46	49	79	100	82
31	Jordan	66	58		81	51	39	33	82	100	77

Figure 14 parser_piracy.py input file.

```

Country Year Piracy and Armed Robbery
0 Algeria 2019 100
1 Angola 2019 56
2 Bahrain 2019 100
3 Bangladesh 2019 19
4 Benin 2019 12
5 British Indian Ocean Territories 2019 100
6 Brunei 2019 70
7 Cabo Verde 2019 91
8 Cambodia 2019 90
9 Cameroon 2019 19
10 Comoros 2019 92
11 Congo REP 2019 47
12 Cote d'Ivoire 2019 51
13 Djibouti 2019 99
14 DRC 2019 63
15 East Timor 2019 100
16 Egypt 2019 100
17 Equatorial Guinea 2019 16
18 Eritrea 2019 100
19 Mayotte 2019 92
20 Gabon 2019 35
21 Ghana 2019 11
22 Guinea 2019 78
23 Guinea-Bissau 2019 88
24 India 2019 24

```

Figure 15 parser_piracy.py output.

- parse_wgi.py:

The script receives as input an excel file that divides the indicators by tab, hence, for each governance indicator. In each tab, it is possible to find a

column with the countries and a multi-header for the year and different metrics in which “rank” is the only value to consider. The script has two modes of operation which it parses all the governance indicators from the first year to the last or it is possible to supply a year of interest. Figure 16 showed the input file, and figure 17 the transformation and output.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	
1	Voice and Accountability																											
2	Reflects perceptions of the extent to which a country's citizens are able to participate in selecting their government, as well as freedom of expression, freedom of association, and a free media.																											
3																												
4	Legend																											
5	Estimate Estimate of governance (ranges from approximately -2.5 (weak) to 2.5 (strong) governance performance)																											
6	StdErr Standard error reflects variability around the point estimate of governance.																											
7	NumSrc Number of data sources on which estimate is based																											
8	Rank Percentile rank among all countries (ranges from 0 (lowest) to 100 (highest) rank)																											
9	Lower Lower bound of 90% confidence interval for governance, in percentile rank terms																											
10	Upper Upper bound of 90% confidence interval for governance, in percentile rank terms																											
11																												
12	The Worldwide Governance Indicators (WGI) are a research dataset summarizing the views on the quality of governance provided by a large number of enterprise, citizen and expert survey respondents in industrial and developing countries. These data are gathered from a number of survey institutes, think tanks, non-governmental organizations, international organizations, and private sector firms. The WGI do not reflect the official views of the World Bank, its Executive Directors, or the countries they represent. The WGI are not used by the World Bank Group to allocate resources.																											
13																												
14																												
15	Country/Territory	Code	1996 Estimate	1996 StdErr	1996 NumSrc	1996 Rank	1996 Lower	1996 Upper	1998 Estimate	1998 StdErr	1998 NumSrc	1998 Rank	1998 Lower	1998 Upper	2000 Estimate	2000 StdErr	2000 NumSrc	2000 Rank	2000 Lower	2000 Upper	2002 Estimate	2002 StdErr	2002 NumSrc	2002 Rank	2002 Lower	2002 Upper	2003 Estimate	
16	Aruba	ABW	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
17	Andorra	ADO	1.56	0.29	3.00	98.50	81.50	100.00	1.53	0.29	3.00	97.51	78.61	100.00	1.54	0.29	3.00	97.01	80.60	100.00	1.44	0.27	3.00	95.52	74.63	100.00	1.42	
18	Afghanistan	AFG	-1.91	0.26	4.00	1.00	0.00	9.50	-2.04	0.26	4.00	0.50	0.00	4.98	-2.03	0.25	4.00	1.00	0.00	5.47	-1.43	0.19	6.00	9.45	2.99	14.93	-1.18	
19	Angola	AGO	-1.58	0.21	6.00	6.00	0.50	13.00	-1.41	0.21	6.00	9.45	2.49	16.92	-1.46	0.20	7.00	8.96	1.99	12.94	-1.24	0.16	9.00	12.94	7.96	20.40	-1.27	
20	Anguilla	AIA	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
21	Albania	ALB	-0.65	0.23	5.00	29.50	16.50	40.50	-0.39	0.23	5.00	38.81	27.36	48.76	-0.29	0.20	6.00	41.29	30.85	51.74	-0.01	0.17	7.00	48.26	41.29	56.72	0.07	
22	Netherlands Antilles (former)	ANT	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
23	United Arab Emirates	ARE	-0.41	0.22	5.00	37.00	24.00	50.00	-0.52	0.23	5.00	33.83	21.39	46.27	-0.52	0.23	5.00	31.34	22.39	45.77	-0.63	0.21	6.00	30.35	20.40	41.29	-0.91	
24	Argentina	ARG	0.39	0.20	8.00	62.50	53.00	69.00	0.31	0.20	8.00	57.71	47.76	67.16	0.42	0.19	9.00	63.19	52.74	69.65	0.26	0.16	11.00	54.22	48.26	63.18	0.35	
25	Armenia	ARM	-0.57	0.26	4.00	33.00	16.50	45.00	-0.34	0.23	5.00	40.80	28.36	49.75	-0.37	0.20	6.00	36.32	28.36	50.25	-0.44	0.17	7.00	36.82	26.87	43.28	-0.47	
26	American Samoa	ASM	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
27	Antigua and Barbuda	ATG	0.33	0.29	3.00	61.50	45.00	71.50	0.34	0.29	3.00	59.20	46.27	71.14	0.39	0.29	3.00	60.20	49.25	73.63	0.34	0.27	3.00	58.21	45.27	70.15	0.32	
28	Australia	AUS	1.44	0.21	7.00	93.00	82.50	100.00	1.41	0.21	7.00	93.03	79.10	100.00	1.47	0.20	8.00	94.03	85.07	100.00	1.41	0.19	9.00	93.53	82.09	100.00	1.44	
29	Austria	AUT	1.45	0.21	7.00	93.50	83.50	100.00	1.35	0.21	7.00	92.04	76.62	100.00	1.32	0.20	8.00	90.05	76.62	99.50	1.31	0.19	9.00	91.54	74.63	100.00	1.34	
30	Azerbaijan	AZE	-1.12	0.24	5.00	15.50	8.00	25.00	-1.02	0.21	6.00	17.91	9.95	30.35	-0.91	0.18	7.00	21.89	12.94	30.83	-0.98	0.16	9.00	20.40	12.94	26.87	-1.04	
31	Burundi	BDI	-1.55	0.26	4.00	7.50	0.00	15.50	-1.44	0.26	4.00	8.46	1.49	17.41	-1.49	0.24	5.00	7.46	1.49	14.43	-1.16	0.18	7.00	14.43	7.96	22.39	-1.18	
32	Belgium	BEL	1.43	0.21	7.00	92.50	81.50	100.00	1.34	0.21	7.00	91.54	76.62	100.00	1.37	0.20	8.00	92.54	78.61	100.00	1.38	0.19	9.00	93.03	79.10	100.00	1.48	
33	Benin	BEN	0.26	0.26	4.00	59.00	44.50	68.50	0.43	0.26	4.00	62.69	48.76	72.14	0.41	0.24	5.00	60.70	51.74	71.64	0.05	0.18	7.00	51.24	41.79	58.21	0.19	
34	Burkina Faso	BFA	-0.53	0.23	5.00	20.50	10.50	44.50	-0.40	0.23	5.00	37.81	25.87	48.26	-0.32	0.21	6.00	44.28	30.85	53.23	-0.39	0.17	8.00	39.30	28.36	45.27	-0.31	
35	Bangladesh	BGD	-0.06	0.21	6.00	50.00	37.00	59.00	-0.09	0.21	6.00	47.26	34.83	56.72	-0.23	0.20	7.00	44.78	30.85	52.74	-0.43	0.16	9.00	37.31	26.87	44.28	-0.55	
36	Bulgaria	BGR	0.43	0.21	6.00	63.50	53.50	71.00	0.44	0.21	6.00	63.18	51.24	71.14	0.46	0.18	8.00	62.19	53.73	70.65	0.52	0.15	10.00	63.68	56.72	70.15	0.55	
37	Bahrain	BHR	0.72	0.22	5.00	25.50	16.00	37.50	-1.07	0.23	5.00	16.42	7.96	28.85	-1.06	0.23	5.00	17.41	8.45	28.36	-0.57	0.17	8.00	31.34	22.39	41.28	-0.59	
38	Bahamas, The	BHS	1.10	0.24	4.00	83.00	68.50	94.50	1.16	0.26	4.00	85.07	69.15	99.50	1.20	0.25	4.00	87.06	71.14	99.50	1.19	0.24	4.00	87.56	71.14	100.00	1.03	
39	Bosnia and Herzegovina	BIH	-0.10	0.25	4.00	47.50	34.50	60.00	-0.01	0.25	4.00	48.76	36.32	61.69	-0.11	0.21	5.00	48.76	34.33	56.22	-0.09	0.16	8.00	46.27	40.30	53.73	0.21	
40	Belarus	BLR	-0.82	0.26	4.00	23.50	13.00	37.00	-0.77	0.23	5.00	27.36	13.43	38.31	-1.31	0.20	6.00	11.94	4.98	19.40	-1.45	0.17	8.00	8.46	3.48	13.93	-1.46	

Figure 16 parser_wgi.py input file.

	country	year	voice_accountability	political_stability	gov_effectiveness	regulatory_quality	rule_of_law	control_of_corruption
0	Aruba	2019	92.610840	95.238098	80.288460	75.961540	86.538460	85.096153
1	Andorra	2019	87.192116	98.571426	98.076928	86.057693	90.865387	87.500000
2	Afghanistan	2019	21.674877	0.952381	7.211538	10.096154	4.326923	6.730769
3	Angola	2019	25.615763	35.238094	12.980769	16.346153	13.461538	13.942307
4	Anguilla	2019	NaN	NaN	96.190475	76.442307	63.461540	87.500000
5	Albania	2019	52.216747	52.857143	50.480770	63.942307	38.942307	33.173077
6	Netherlands Antilles (former)	2019	NaN	NaN	NaN	NaN	NaN	NaN
7	United Arab Emirates	2019	17.733990	69.523811	88.942307	78.365387	77.884613	83.653847
8	Argentina	2019	66.502464	43.333332	49.038460	33.653847	37.019230	53.365383
9	Armenia	2019	47.783253	27.619047	50.000000	63.461540	49.038460	50.000000
10	American Samoa	2019	NaN	NaN	91.428574	72.596153	40.865383	87.980766
11	Antigua and Barbuda	2019	72.413795	81.904762	51.923077	68.269234	65.384613	63.461540
12	Australia	2019	93.103447	88.571426	92.788460	98.557693	93.269234	94.230766
13	Austria	2019	93.596062	82.857140	91.826920	91.346153	97.115387	90.865387
14	Azerbaijan	2019	7.389163	21.904762	46.153847	43.750000	30.288462	19.711538
15	Burundi	2019	4.926108	7.142857	10.096154	14.903846	5.769231	4.807693
16	Belgium	2019	95.566505	61.904762	80.769234	87.500000	88.461540	91.346153
17	Benin	2019	50.246304	34.285713	36.057693	37.500000	26.923077	42.788460
18	Burkina Faso	2019	39.901478	11.904762	22.115385	37.019230	37.500000	49.519230
19	Bangladesh	2019	27.093596	15.238095	23.557692	15.384615	27.884615	16.346153
20	Bulgaria	2019	60.591133	66.190475	65.384613	71.153847	54.807693	50.480770
21	Bahrain	2019	9.852217	22.389953	63.942307	67.788460	68.750000	56.730770

Figure 17 parser_wgi.py output.

- parser_strait.py:

The sea route takes into consideration if it passes through one or more straits or canals. The data needed from the straits are the basin countries as well as the alpha, there is no data source the provides this type of information, so the file must be created by hand. The output of the script is a DataFrame with the name of the strait and the alpha index and a dictionary with key as the name of the strait and the basin countries as values as shown in figure 19.

	A	B	C	D	E
1	strait name	alpha	country_1	country_2	country_3
2	BAB-EL-MANDEB STRAIT		5 Djibouti	Eritrea	Yemen
3	DANISH GREAT BELT STRAIT		4 Denmark		
4	DOVER STRAIT		16 France	England	
5	HUDSON STRAIT	14.16	Canada		
6	MALACCA STRAIT	16	Malaysia	Singapore	Indonesia
7	PANAMA CANAL	1014	Panama		
8	SINGAPORE STRAIT	6.43	Malaysia	Singapore	Indonesia
9	STRAIT OF GIBRALTAR	4.3	United Kingdom	Morocco	Spain
10	STRAIT OF HORMUZ	20	Oman	Iran, Islamic Rep.	United Arab Emirates
11	SUEZ STRAIT	633	Egypt, Arab Rep.		
12	TAI-WAN STRAIT	4	Taiwan, China		
13	TURKISH STRAIT	100	Turkey		
14	FLORIDA STRAIT	4	United States	Cuba	Bahamas
15					

Figure 18 parser_strait.py custom input file.

```

    strait name      alpha
0      BAB-EL-MANDEB STRAIT      5.00
1      DANISH GREAT BELT STRAIT   4.00
2              DOVER STRAIT      16.00
3              HUDSON STRAIT     14.16
4              MALACCA STRAIT     16.00
5              PANAMA CANAL     1014.00
6              SINGAPORE STRAIT   6.43
7      STRAIT OF GIBRALTAR       4.30
8      STRAIT OF HORMUZ          20.00
9              SUEZ STRAIT       633.00
10             TAI-WAN STRAIT     4.00
11             TURKISH STRAIT     100.00
12             FLORIDA STRAIT     4.00
BAB-EL-MANDEB STRAIT->['Djibouti', 'Eritrea', 'Yemen']
DANISH GREAT BELT STRAIT->['Denmark', nan, nan]
DOVER STRAIT->['France', 'England', nan]
HUDSON STRAIT->['Canada', nan, nan]
MALACCA STRAIT->['Malaysia', 'Singapore', 'Indonesia']
PANAMA CANAL->['Panama', nan, nan]
SINGAPORE STRAIT->['Malaysia', 'Singapore', 'Indonesia']
STRAIT OF GIBRALTAR->['United Kingdom', 'Morocco', 'Spain']
STRAIT OF HORMUZ->['Oman', 'Iran, Islamic Rep.', 'United Arab Emirates']
SUEZ STRAIT->['Egypt, Arab Rep.', nan, nan]
TAI-WAN STRAIT->['Taiwan, China', nan, nan]
TURKISH STRAIT->['Turkey', nan, nan]
FLORIDA STRAIT->['United States', 'Cuba', 'Bahamas']

```

Figure 19 parser_strait.py output.

- parser_lvh.py:

This script uses a custom input file, as the name of the commodities must match with the name provided by Alphatanker, the low-heat values were provided in MJ/Kg. The input file is created to be easily transformed to a DataFrame with just two columns commodity name and low-heat value, the output of the script is shown in figure 20.

	name	lvh
0	Bitumen Mixture	40.2
1	Asphalt/Bitumen Mixture	40.2
2	Chemicals	NaN
3	Carbon Black Feedstock	NaN
4	Vegetable Oil	NaN
5	Soybean Oil	NaN
6	NaN	NaN
7	LNG	44.2
8	Asphalt	40.2
9	Palm Oil	NaN
10	Unleaded Motor Spirit	44.3
11	LPG	47.3
12	Ethylene	NaN
13	Liquid Fertilizers	NaN
14	Ammonia	NaN
15	Marine Gas Oil	45.0
16	Sunflower Oil	NaN
17	Bio Diesel	27.0
18	Ethanol	NaN
19	Fish Oil	NaN
20	Molasses	NaN
21	Crude Oil	42.3
22	Carbon Dioxide	NaN
23	Fuel Oil	40.4
24	Gas Oil	43.0

Figure 20 parser_lvh.py output.

- parser_pipeline.py:

This script uses two custom files as input, the first one has the pipelines already pre-calculated by the EST-Lab which contains the total length, the shared value, the end port, and the countries the pipeline pass through, also the length of the pipeline per country. In the other file as there is no information about the other pipelines, endpoints, and lengths a rough approximation using the square root of the country surface is used and assumed a shared value of one. The output is shown in figure 21.

	name	total_length	share_val	load_port
0	BTC	2204.16	0.38	Ceyhan
1	ITP	1225.10	0.62	Ceyhan
2	CPC	2149.30	1.00	Novorossiysk
3	NEO	1804.85	0.63	Novorossiysk
4	JSC	223.00	0.38	Novorossiysk
5	BPS_1	3794.00	1.00	Primorsk
6	BPS_2	3794.44	0.67	Ust-Luga
7	UAS	4574.74	0.33	Ust-Luga
8	WREP	995.02	1.00	Supsa Marine Terminal
pipeline	name	country	length	
0	BTC	Azerbaijan	547.04	
1	BTC	Georgia	334.82	
2	BTC	Turkey	1322.30	
3	ITP	Iraq	440.00	
4	ITP	Turkey	784.58	
5	CPC	Kazakhstan	640.00	
6	CPC	Russian Federation	1509.00	
7	NEO	Azerbaijan	279.20	
8	NEO	Russian Federation	1525.65	
9	JSC	Russian Federation	223.00	
10	BPS_1	Russian Federation	3794.00	
11	BPS_2	Russian Federation	3794.44	
12	UAS	Russian Federation	3500.00	
13	UAS	Kazakhstan	1074.74	
14	WREP	Azerbaijan	546.02	
15	WREP	Georgia	449.00	
	name	total_length	share_val	load_port
0	Aruba_Internal	13.416408	1	Aruba
1	Afghanistan_Internal	807.997525	1	Afghanistan
2	Angola_Internal	1116.557209	1	Angola
3	Albania_Internal	169.558250	1	Albania
4	Andorra_Internal	21.679483	1	Andorra
..
259	Kosovo_Internal	0.000000	1	Kosovo
260	Yemen,Rep._Internal	726.615442	1	Yemen, Rep.
261	SouthAfrica_Internal	1104.124087	1	South Africa
262	Zambia_Internal	867.530979	1	Zambia
263	Zimbabwe_Internal	625.107991	1	Zimbabwe

Figure 21 parser_pipeline.py output.

4.2. Risk Index Calculations

The calculation of the index is not an easy task as it needs a lot of the information discussed in the previous section, but it is possible to have some pre-calculation that will help to speed the process. The final goal of this section is to calculate the probability of failure for a single corridor which considers the captive branch (pipelines) and sea branch (maritime route). The maritime route tracing was done with an external script that takes the minimum route between two ports, created by Eurostat. This script uses a geo-package file that contains pre-existing routes all over the world, it had to be modified and cancel certain passages where a petroleum ship cannot pass to get accurate results on the routes. Also, this script needs to input the coordinates of the ports to start tracing the routes, which had to be created. Therefore, three scripts were developed to create the input file for the sea route script and find the probability of failure for each corridor.

4.2.1. Scripts

- java_input_OSM.py:

This script uses as input the output of the parser_alphatanker.py to know which route must trace as it has the information of the load port and discharge port. Also, an auxiliary file is used that maps the ports to their respective countries. It uses the Open Street Map API called “Nominatim” to get the coordinates of each port. Alphatanker does not always give the accurate name of the port so the coordinates of the city in which the port contains are passed otherwise it is not found as happened with some of the terminals there is a file with coordinates for some ports and terminals with only two decimals precision to look up. The script looks in the database if the route has already been calculated, since the route is static there is no point to calculate it again, so it skips it. The output file is shown in figure 22.

	A	B	C	D	E	F	G	H	I
1	RouteName	oPort	oCountry	olon	olat	dport	dcountry	dlon	dlat
2	ABOT - Al Basra Oil Terminal-Milazzo	ABOT - Al Basra Oil Terminal	Iraq	11.35	4.47	Milazzo	Italy	15.2415129	38.2208049
3	Agio Theodoroi-Gaeta	Agio Theodoroi	Greece	25.8631613	40.9351416	Gaeta	Italy	13.56281	41.218312
4	Agio Theodoroi-Marghera	Agio Theodoroi	Greece	25.8631613	40.9351416	Marghera	Italy	12.2247811	45.4758097
5	Alexandria-Augusta	Alexandria	Egypt, Arab Rep.	29.52	31.11	Augusta	Italy	15.2196575	37.2369363
6	Alexandria-Genoa	Alexandria	Egypt, Arab Rep.	29.52	31.11	Genoa	Italy	8.9338624	44.40726
7	CPC Terminal-Augusta	CPC Terminal	Russian Federation	37.6415664	44.6391831	Augusta	Italy	15.2196575	37.2369363
8	CPC Terminal-Santa Panagia Bay	CPC Terminal	Russian Federation	37.6415664	44.6391831	Santa Panagia Bay	Italy	15.2739927	37.0974572
9	CPC Terminal-Sarroch	CPC Terminal	Russian Federation	37.6415664	44.6391831	Sarroch	Italy	9.009445	39.067234
10	CPC Terminal-Trieste	CPC Terminal	Russian Federation	37.6415664	44.6391831	Trieste	Italy	13.7931263	45.6504806
11	Ceyhan-Augusta	Ceyhan	Turkey	35.8712312357115	37.0556361	Augusta	Italy	15.2196575	37.2369363
12	Ceyhan-Milazzo	Ceyhan	Turkey	35.8712312357115	37.0556361	Milazzo	Italy	15.2415129	38.2208049
13	Ceyhan-Sarroch	Ceyhan	Turkey	35.8712312357115	37.0556361	Sarroch	Italy	9.009445	39.067234
14	Ceyhan-Trieste	Ceyhan	Turkey	35.8712312357115	37.0556361	Trieste	Italy	13.7931263	45.6504806
15	Corpus Christi-Trieste	Corpus Christi	USA	-97.104649205622	27.8365527	Trieste	Italy	13.7931263	45.6504806
16	Novorossiysk-Augusta	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Augusta	Italy	15.2196575	37.2369363
17	Novorossiysk-Genoa	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Genoa	Italy	8.9338624	44.40726
18	Novorossiysk-Marghera	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Marghera	Italy	12.2247811	45.4758097
19	Novorossiysk-Milazzo	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Milazzo	Italy	15.2415129	38.2208049
20	Novorossiysk-Santa Panagia Bay	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Santa Panagia Bay	Italy	15.2739927	37.0974572
21	Novorossiysk-Taranto	Novorossiysk	Russian Federation	37.844143479874	44.7102335	Taranto	Italy	17.080580124548	40.54881555
22	Primorsk-Naples	Primorsk	Russian Federation	28.616673	60.366665	Naples	Italy	14.2487826	40.8359336

Figure 22 output of the java_input_OSM.py script.

- calculation_route_eez_intersection.py:

This script oversees finding the intersection with the route with the Exclusive Economic Zone which the routes cross. To get the length of each country the function overlay is used which allows to compare two GeoDataFrames containing polygon or multi-polygon geometries and create a new GeoDataFrame with the new geometries representing the spatial combination *and* merged properties. [GeoPandas]. It takes three input files in the form of shapefiles, one shapefile of the routes generated by the sea route script containing the routes of interest, a strait shapefile, and the shapefile of the EE zones.

One consideration is that overlay is an expensive computational operation and intersecting the route with the whole EEZ frame is not feasible, so first, the route is converted to points and spatial joined with the EEZ to know which countries

crossed and decrease the size of the EEZ GeoDataFrame and in the same operation get which strait is involved in the route. All these operations are done with the Pseudo-Mercator projection.

The output of the script gives the name of the corridor, the country it crosses, and the length as shown in figure 23.

	A	B	C	D	E	F
1	corridor_name	country	length	geometry		
2	Alexandria-Augusta	Egypt	490.642285597897	LINESTRING (2919089.618414131 3932791.261201667, 2920800.799433912 3931563.176434615, 3008965.836142185 3868687.973		
3	Alexandria-Augusta	Greece	1000.45133807216	LINESTRING (2038137.391359207 4399604.713110087, 2135413.962014669 4350166.880971999, 2141202.575535919 4347229.875		
4	Alexandria-Genoa	Egypt	490.642285597897	LINESTRING (2919089.618414131 3932791.261201667, 2920800.799433912 3931563.176434615, 3008965.836142185 3868687.973		
5	Alexandria-Genoa	Greece	1054.94036204719	LINESTRING (2035907.542487077 4480661.350005741, 2059466.239420958 4474815.246379535, 2113372.702837601 4459407.062		
6	Arzew-Trieste	Tunisia	293.13672208807	LINESTRING (917012.263260683 4560114.488196473, 927541.8271622538 4561817.780088364, 958989.5833113536 4566966.5378		
7	Arzew-Trieste	Croatia	609.178030801369	MULTILINESTRING ((1844198.592088626 5201825.330973577, 1842699.360973756 5204331.411302072, 1838775.348923293 5211		
8	Arzew-Trieste	Slovenia	21.7894954952224	MULTILINESTRING ((1501232.774791989 5704781.476167714, 1500920.694365708 5709838.877694652, 1504474.046683118 5716		
9	Arzew-Trieste	Algeria	1055.7276041913	LINESTRING (-53517.64446996203 4281501.428120323, -67793.5698931036 4281615.021845065, -41800.46879287423 4309465.67		
10	Ceyhan-Augusta	Greece	1175.36290413811	MULTILINESTRING ((2038279.05637764 4404692.089868512, 2285611.784967493 4363549.044390174, 2333646.145244791 43555		
11	Ceyhan-Augusta	Turkey	876.750530578761	MULTILINESTRING ((3111928.045791141 4366942.477181171, 3115526.418704045 4367632.713494365, 3122318.472236606 4366		
12	Ceyhan-Trieste	Greece	1230.40013434658	MULTILINESTRING ((3303841.744934675 4314215.999162053, 3299303.016310641 4315943.159029788), (3298581.205418359 431		
13	Ceyhan-Trieste	Turkey	913.004211758022	MULTILINESTRING ((4003798.648789134 4425822.79438707, 4014598.28609592 4404490.72993346, 3999291.856111845 4392340		
14	Ceyhan-Trieste	Croatia	609.178030801369	MULTILINESTRING ((1844198.592088626 5201825.330973577, 1842699.360973756 5204331.411302072, 1838775.348923293 5211		
15	Ceyhan-Trieste	Slovenia	21.7894954952224	MULTILINESTRING ((1501232.774791989 5704781.476167714, 1500920.694365708 5709838.877694652, 1504474.046683118 5716		
16	Sidi Kerir-Augusta	Egypt	507.808472649904	LINESTRING (2919089.618414131 3932791.261201667, 2920800.799433912 3931563.176434615, 3008965.836142185 3868687.973		
17	Sidi Kerir-Augusta	Greece	1000.45133807216	LINESTRING (2038137.391359207 4399604.713110087, 2135413.962014669 4350166.880971999, 2141202.575535919 4347229.875		
18	Sidi Kerir-Leghorn	Egypt	507.808472649904	LINESTRING (2919089.618414131 3932791.261201667, 2920800.799433912 3931563.176434615, 3008965.836142185 3868687.973		
19	Sidi Kerir-Leghorn	Greece	1054.94036204719	LINESTRING (2035907.542487077 4480661.350005741, 2059466.239420958 4474815.246379535, 2113372.702837601 4459407.062		

Figure 23 output of the calculation_route_eez_intersection.py.

- calculation_risk_model.py:

This script is the final one from the calculation module chain, it contains all the functions needed to calculate the indicators of section 1.1.1. At this point, all the information should have been stored in the database, so this script just retrieves the data and start the calculations. It saves in the database the geopolitical and sea risk; alpha normalizes for the straits and the corridor failure.

4.3. Database

In section 3.5 the tables and the architecture of the database was defined, the way to communicate with the PostgreSQL database is to use the python library pycpg2, which is a database adapter for the Python programming language.

4.3.1. Scripts

- Database_conn.py:

This is a simple script the contains the function that gets the credentials of the database as well as the function to insert into and get the values from the database, the script is made to work with pandas DataFrames so all the input files are DataFrames as well as the returned values. This script is not used in the Django Framework as Django has its methods to retrieve information of the database that is proven to be secure.

4.4. Wrapper

The wrapper script contains all the imports of all the scripts explained before, it is in charge of putting everything together and manages the order of the flow and how the functions should be executed.

The script is very simple and uses a flag system, if the flag is put to “1” the functions of the specific task will be executed otherwise it will be skipped. Some functions need more parameters an input file or a year, all the parameters must be provided to have a successful execution. It is important to mention that there are some functions that there is no need to run very often or just needed to run once for example to add the strait information to the database as this geographical data rarely sees changes. Meanwhile, some functions have to be run periodically to update the intake of the petroleum products coming from the different routes, in this case, the administrator decides how often this data should be updated.

```
1. #####CONFIGURATION FLAGS#####
2. #####
3. flag_insert_lvh = 0
4. lvh_file = "../input_data_spreadsheet/commodity_AT_GDP.xlsx"
5.
6. flag_insert_straits = 0
7. strait_file = '../input_data_spreadsheet/straits_country.xlsx'
8.
9. flag_insert_pipelines = 0
10. pipeline_file = '../input_data_spreadsheet/pipeline_info.xlsx'
11.
12. flag_insert_approx_pipelines = 0
13. approx_pipeline_file = '../input_data_spreadsheet/surface_area_country.xls'
14.
15. flag_insert_wgi = 0
16. wgi_file = "../input_data_spreadsheet/wgidataset.xlsx"
17. wgi_year = 2018
18.
19. flag_insert_piracy = 0
20. piracy_file = '../input_data_spreadsheet/2019 Stable Seas Index Data(OEF).xlsx'
21. piracy_year = 2019
22.
23. flag_geo_risk = 0
24. geo_risk_year = 2019
25.
26. flag_ck_risk = 0
27. ck_risk_year = 2019
28.
29. flag_intake = 0
30. alphantanker_file = '../input_data_spreadsheet/alphantanker_files/2021/01_04-30_04
    alphantanker.xlsx'
31. flag_corridor_failure = 1
32. corridor_failure_year = 2019
```

Figure 24 Wrapper configuration flags.

4.5. Web Application

The web application was developed using the Django Framework, as mentioned before this framework uses a Model-View-Template architecture, in this section is explained all the work that was made by each part as well as the user interface where the user can interact with.

The first step is to create an app inside Django in this case the name is stories, which is going to contain all the parts and code.

Django needs some parameters to be configured to work as it is intended, the important settings to consider are:

- **INSTALLED_APPS:**
It is important to add the app already created to enable it and make it visible to the framework, in the case of this project the stories app had been added.
- **DATABASES:**
This variable contains all the parameters needed to connect to the database in the form of a python dictionary. It is needed to specify the IP address of the database, port, name, password, and engine.
- **STATIC_URL:**
Use when referring to static files located, in this case, it is specified the folder where the CSS styling file and the images are stored.
- **EMAIL_BACKEND:**
There are some variables to configure the email services, in this application, only a debug console backend is enabled to see the emails in the terminal, but it is possible to configure a proper email service.
- **DEBUG:** For testing, the value should be true, for production must be changed to false.

4.5.1. Model

The model is the part of the framework that serves as the link between the View and the Database and allows the programmer to interact with the data using a database-abstraction API called ORM(Object Relational Mapper). For this project in particular the only model that the Django framework is going to manage is the users, the other tables are managed externally using PostgreSQL database-manager with the same structure explained in section 4.3.

The models were exported from the database using a Django method 'inspectdb' which allows the use of a legacy database. By default, "inspectdb" creates unmanaged models. That is, managed = False in the model's Meta class tells Django

not to manage each table's creation, modification, and deletion [25]. Django will not manage any table of the risk model only the User tables for authentication.

```
33. class Corridor(models.Model):
34.     corridor_id = models.IntegerField(primary_key=True)
35.     corridor_name = models.CharField(unique=True, max_length=100, blank=True, null=True)
36.     load_country = models.CharField(max_length=20, blank=True, null=True)
37.     load_port = models.CharField(max_length=20, blank=True, null=True)
38.     discharge_port = models.CharField(max_length=20, blank=True, null=True)
39.     discharge_country = models.CharField(max_length=20, blank=True, null=True)
40.
41.     class Meta:
```

Figure 25 Code Snippet of the Model

The class Meta gives extra instructions to Django in this case the management of this table oversees the admin and Django cannot do any modification also the table name must match with the table name of the database otherwise it will not work.

Django has a predefined User model, but for this project was not enough since in the future the possibility of giving different privileges of the users is desired, for this matter a new field of CharField is added to the user model that will help to identify the type of users for example if is a paid user or a free user.

```
1. class Profile(models.Model):
2.     user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='profile')
3.     type = models.CharField(max_length=1, blank=True, null=True)
4.
5. @receiver(post_save, sender=User)
6. def create_user_profile(sender, instance, created, **kwargs):
7.     if created:
8.         Profile.objects.create(user=instance)
9.
10. @receiver(post_save, sender=User)
11. def save_user_profile(sender, instance, **kwargs):
12.     instance.profile.save()
13.
```

Figure 26 Extend Django User Class Code Snippet

4.5.2. Views

The views contain the logic of the application, it manages the information that comes from the database and how it will be displayed by the templates to the users. The views manage the home page, user authentication, the stories with the data needed, and the AJAX requests. Each template needs its view to manage and render.

In the views is also manages the interactions of the user with the templates (User Interface) as it manages all the AJAX, POST, and GET response methods. Three utils scripts were written to help maintain order in the views.

4.5.2.1. Utils

Utils is a python module in charge of helping with diverse tasks for creating plots, querying and process information coming from Models and for the energy risk calculation.

The `utils_plot.py` script uses the “Plotly” library to create interactive graphs, this script oversees the creation of all the plots that are going to be managed by the views and display in the templates. The `utils_query.py` script is in charge of asking the database the information needed as well as some functions needed to transform the data that is going to be present in the template and the information needed for the plots to be generated. Finally, `utils_calculation.py` contains the functions that return the final risk calculation of each corridor.

4.5.3. Templates

The templates are the user interface in this section are presented the user interface and all the templates created in the project.

- Home Page:

Is the first page the user sees when it opens the application. Contains all the stories and the link to them. When the user Log in the Guest changes to the user’s name and the Log in/SignUp section of the navigation bar changes to Log out.

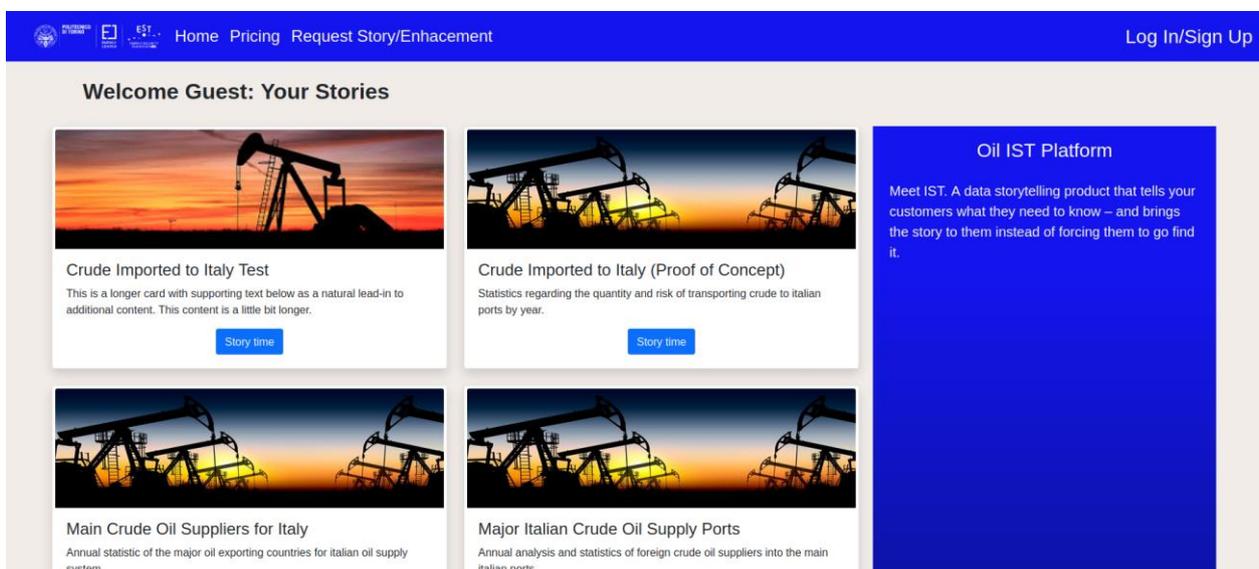


Figure 27 Home page without user login.

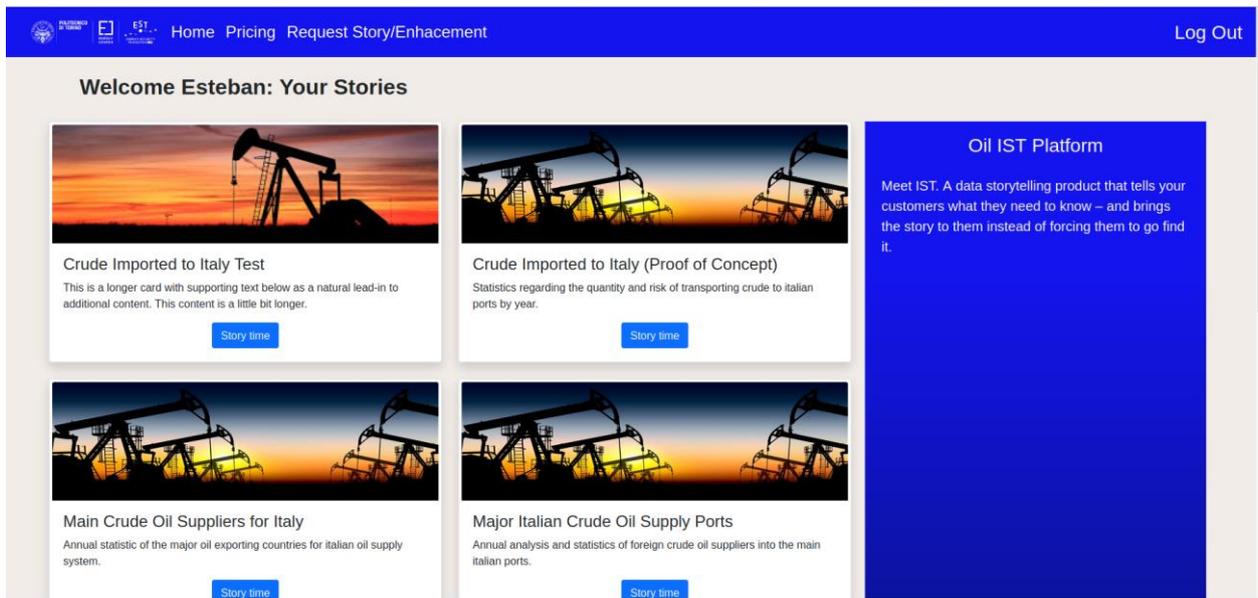


Figure 28 Home page after user logs in.

- Log in page:

Contains the form that will allow the users to log into the application. Besides, it has the link to forgot password and sign up. The logic of this page is contained in the view. After the user inserts their information, it will check if the form is valid (proper email address has been inserted) and perform a look-up on the database if the user is valid it will log in and redirect to the home page otherwise it will raise an error.

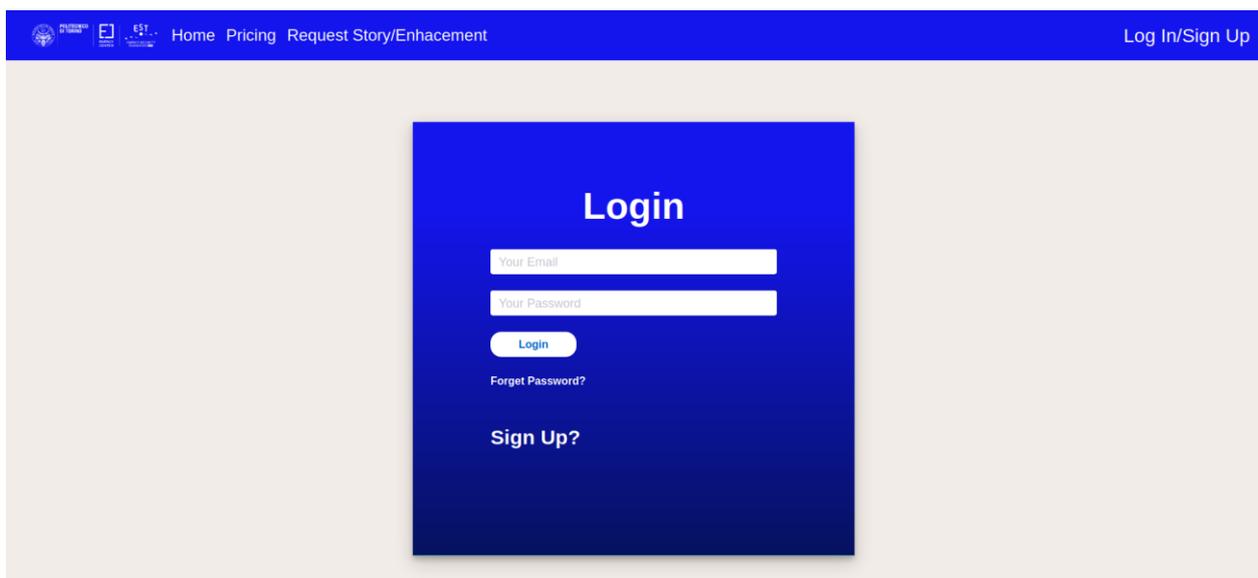


Figure 29 Login page.

- Sign up page:

Contains the form with the information required for the users to register in the application. After the user submits a valid form, a mail is sent to the email account provided by the user to verify its identity and activate the account. Django has its built-in password checker and validation so it will validate if the password is too short or is too common and rise and error if it is the case otherwise the user is redirected to the login page.

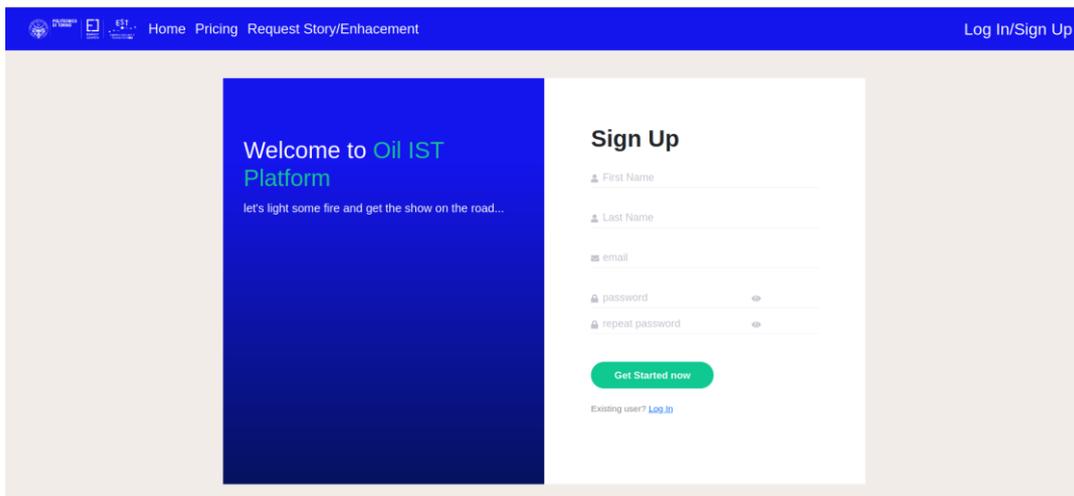


Figure 30 Sign Up page.

- Password reset page:

Ask the user to insert the email they used in the registration phase. It will send a verification link to their email address with the instructions to reset the password. The URL for password reset looks like this:

http://<domain>/users/password_reset_confirm/MjA/ao4o9k728d610d80433fecf080cb8c82bbf449.

Django will automatically create a token that Encodes a byte string to a base64 string for use in URLs, stripping any trailing equal signs with, in the case the byte string is obtained by converting the user primary key to bytes.

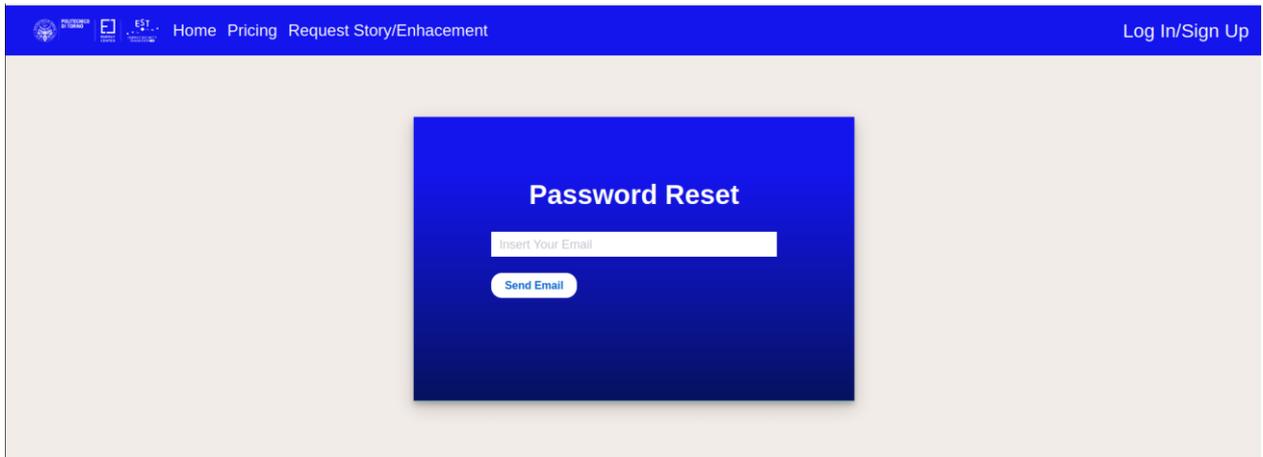


Figure 31 Password reset page.



Figure 32 Password reset after submit form.

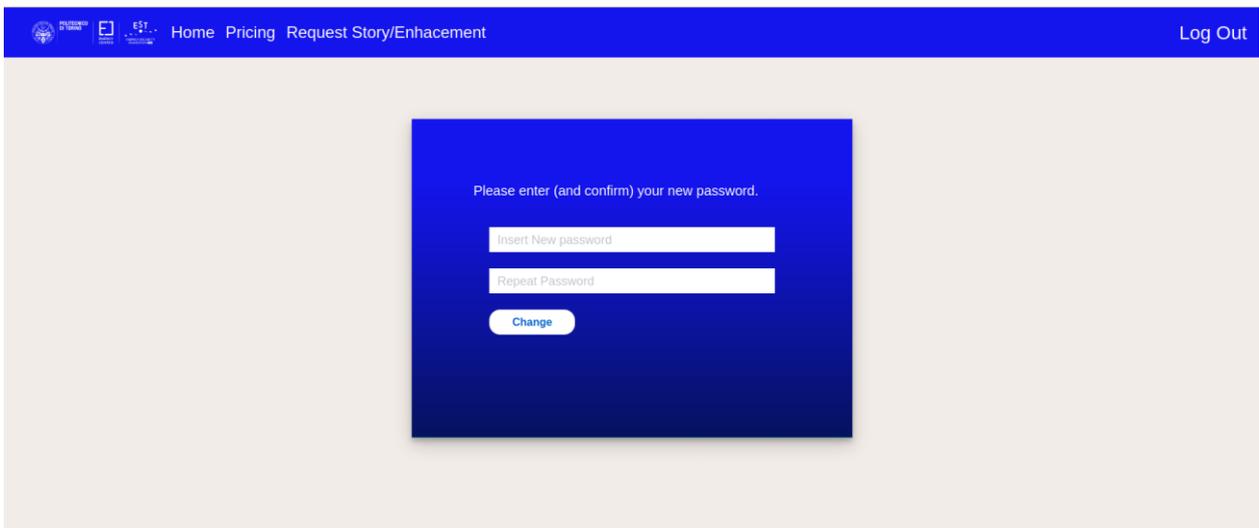


Figure 33 After clicking URL receive in the mail.

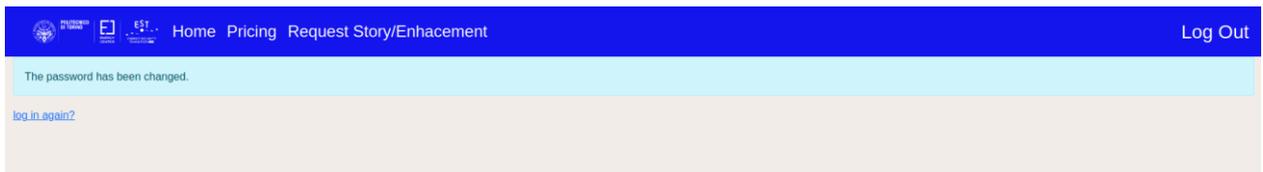


Figure 34 Password change successfully.

- Contact page:

This is a form that the users can fill to send contact the admins via email, where the user can request new stories based on their interest or if a story is already published and it needs an enhancement or information update also can be requested.

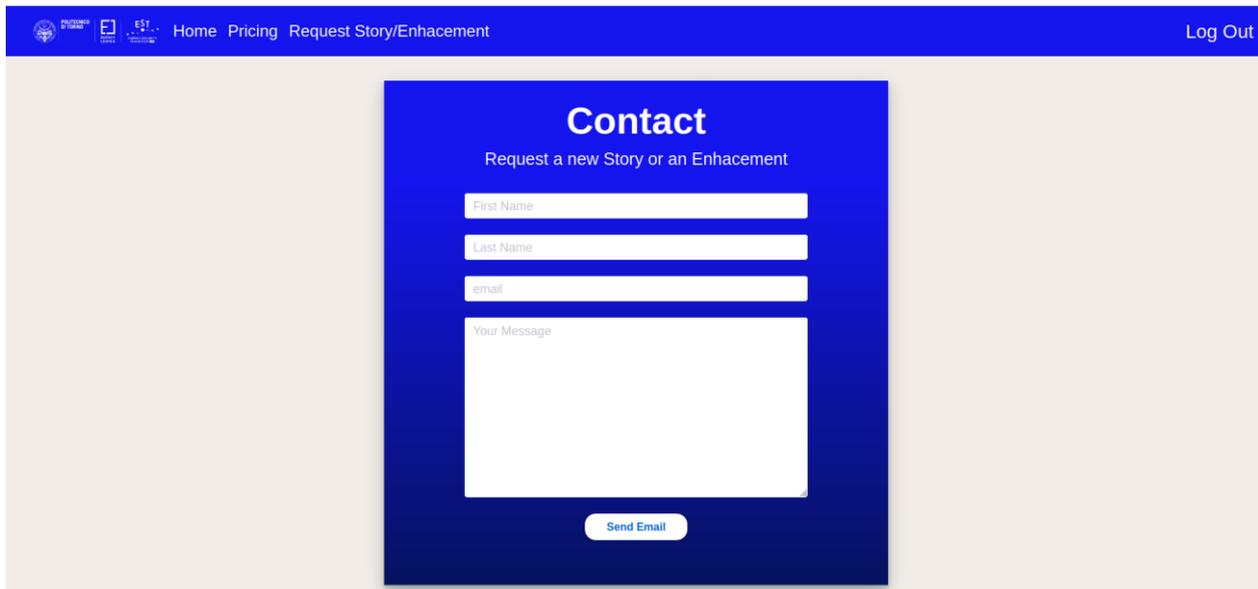


Figure 35 Contact page.

- Stories pages:

Is the core of the platform, each story is having a different narrative and different graphs as well as the interactivity with the users, all the stories have in common is a left panel with a general narrative and, on the right side a panel that when the user select their preferences it gets updated in an asynchronous way using AJAX so there is no need to refresh the page.

Each story has its own template (HMTL) following the structure of the left-side a general narrative and on the right-side user inputs. Also, it variates in what the users can make or interact with, for example in some stories they can choose a time frame but in other stories, they can only choose the year, as for location is concern sometimes the user can choose between countries, load ports or discharge ports. The plots by themselves are interactive with these features:

- Download
- Zoom in.
- Zoom Out.
- Hover text.
- Reset pan.

- Turn on/off labels.

All the logic of these stories as was mentioned before are managed by the views.



Figure 36 Example of a Story #1.



Figure 37 Example of a Story #2.

5. Validation

For this application, manual informal ad hoc testing was performed to check the correctness of the outputs. Each part of the application was tested to guarantee if the output was consistent and the desired one. A manual test consists in execute manually the test cases without the use of an automatic tool.

For the first part of the application the parsers, since the only scope of this part is to transform the input data into DataFrames the outputs were meticulously inspected by taking random pieces of data of the input file and see if the output DataFrame field matches them.

In the calculation part, the EST Lab had calculated the most important corridors like Russia, Egypt, and Turkey. So, the output of the scripts was compared to the data provided by the EST Lab to check for the correctness of the calculations. Also, some logs were dumped to check if the sea routes make sense, as well as the coordinates of the ports, were given correctly by the Nominatim API.

The web application was also informal tested and focus on feature and integration testing, each feature was tested individually for correctness and integration with the system as well as the integration of the database to the system. The User interface was tested to check for missing links, invalid paths, or bad user inputs.

In general, the application limits the user inputs most of the choices the user can make are dropdown menus, only the form for log-in and registration, the user can input data from the keyboard to minimize SQL injection also the methods inside Django help to mitigate and validate all the forms and inputs.

5.1. Data Collection Validation

As was mentioned before the validation done to these scripts was to select random data from the file and compare if the output matches it.

For example in the `parser_wgi.py`, five of the most important countries for oil supply were selected, Egypt, Russia, Turkey, United States, and Libya besides other five random to analyze if the script was working correctly so it considers the country, indicator, and year and extract the data and assert it. A similar approach was used for the other scripts such as `parse_piracy.py` to see if the value of piracy was correct for country and `paser_alphatanker.py` as well.

The custom files were created considering an easy translation into a DataFrames, the custom files are Microsoft Excel or CSV files and were visually checked if both tables match like for the parser_straits.py and parser_lvh.py.

5.2. Risk Index Calculation

For this part, the test cases were provided by the EST-Lab in which they had some corridors already calculated. So, it was possible to assert these values with the output of the script.

This part is very complex and uses an external program to calculate the sea routes created by Eurostat the output of this program is considered valid. Also, this program uses an input file with a for this there is a script java_input_csv_OSM.py that generates this input.

The java_input_csv_OSM.py script uses the Open Street Map API called “Nominatim” and there the way of test that the coordinates given were correct was done in two steps one was manually taken random coordinates from the output of the script, “java_input.csv”, and put them on Google Maps, also for this matter the most concurrent ports were selected like CPC Terminal, Sidi-Kerir, Corpus Christi and Ceyhan. A log was generated also to check if the API was unable to find a port as it has layers when it can find the coordinates for a specific port as well of determine where the coordinates come from, the second step is made when the calculation of the length of the intersection with the EEZ Zone is done as it is compared with the EST-Lab test case file to determine if start and end of the routes make sense if not there was a possible mistake with the coordinates.

```

C > Users > esteb > Desktop > Thesis > logs > java_input_csv_OSM_01_2020.log
1 DEBUG:urllib3.util.retry:Converted retries value: 2 -> Retry(total=2, connect=None, read=None, redirect=None, status=None)
2 DEBUG:urllib3.util.retry:Converted retries value: 2 -> Retry(total=2, connect=None, read=None, redirect=None, status=None)
3 WARNING:root:Cannot assign discharge country to port Porto Torres: index 0 is out of bounds for axis 0 with size 0
4 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Aliaga+Port%2C+Turkey&format=json&limit=1
5 DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): nominatim.openstreetmap.org:443
6 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Aliaga+Port%2C+Turkey&format=json&limit=1 HTTP/1.1" 200 None
7 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=CivitavecchiaPort%2C+Italy&format=json&limit=1
8 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=CivitavecchiaPort%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
9 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Civitavecchia%2C+Italy&format=json&limit=1
10 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Civitavecchia%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
11 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Antwerp+Port%2C+Belgium&format=json&limit=1
12 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Antwerp+Port%2C+Belgium&format=json&limit=1 HTTP/1.1" 200 None
13 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=SarrochPort%2C+Italy&format=json&limit=1
14 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=SarrochPort%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
15 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Sarroch%2C+Italy&format=json&limit=1
16 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Sarroch%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
17 WARNING:root:Cannot assign discharge country to port Vada: index 0 is out of bounds for axis 0 with size 0
18 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Aspropyrgos+Port%2C+Greece&format=json&limit=1
19 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Aspropyrgos+Port%2C+Greece&format=json&limit=1 HTTP/1.1" 200 None
20 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=NaplesPort%2C+Italy&format=json&limit=1
21 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=NaplesPort%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
22 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Naples%2C+Italy&format=json&limit=1
23 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Naples%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
24 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Bayport+Port%2C+United+States&format=json&limit=1
25 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Bayport+Port%2C+United+States&format=json&limit=1 HTTP/1.1" 200 None
26 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=TriestePort%2C+Italy&format=json&limit=1
27 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=TriestePort%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
28 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Trieste%2C+Italy&format=json&limit=1
29 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Trieste%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
30 WARNING:root:Cannot assign load country to port Bonny: index 0 is out of bounds for axis 0 with size 0
31 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Burgas+Port%2C+Bulgaria&format=json&limit=1
32 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Burgas+Port%2C+Bulgaria&format=json&limit=1 HTTP/1.1" 200 None
33 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Burgas%2C+Bulgaria&format=json&limit=1
34 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Burgas%2C+Bulgaria&format=json&limit=1 HTTP/1.1" 200 None
35 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=FiumicinoPort%2C+Italy&format=json&limit=1
36 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=FiumicinoPort%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None
37 DEBUG:geopy:Nominatim.geocode: https://nominatim.openstreetmap.org/search?q=Fiumicino%2C+Italy&format=json&limit=1
38 DEBUG:urllib3.connectionpool:https://nominatim.openstreetmap.org:443 "GET /search?q=Fiumicino%2C+Italy&format=json&limit=1 HTTP/1.1" 200 None

```

Figure 38 Java Input Log File

The validation of the intersection script “calculation_route_eez_intersection.py” a debug CSV file was generated as it contains the same information of the DataFrame, which is the countries the route traverse between the load and discharge port. Then was checked if each route has a logical and congruent route. For the length of the routes, the EST-Lab calculates some of them manually and the assert was made with these values. The routes can be different since the algorithm optimize them, but the model can handle these error margins since is based on a weight system.

Country	Corridor name	Type	Country crossed	Branch
Egypt	Sidi Kerir-AU	Open sea	Egypt	461
			Greece	789
			Libya	166
		Captive	Egypt	450
		Total	1866	
Egypt	Sidi Kerir-LI	Open Sea	Egypt	467
			Greece	942
			Libya	4
			Tunisia	42
		Captive	France	72
		Egypt	450	
		Total	1977	
Egypt	Sidi Kerir-TS	Open sea	Croatia	638
			Egypt	453

	Greece	1259
	Slovenia	21
Captive	Egypt	450
Total		2822

Table 4 Egypt Corridor provided by EST- Lab.

In table 4, it is possible to see the characterization of the corridor of Egypt provided by the EST-Lab. The countries crossed and the branch lengths were the values compared with the output of the script in figure 39 as it is possible to observe that the countries that corridors traverse are the same and the values are in the margin of what is expected, is not possible to do an exact match as the values calculated by the EST- Lab were done by hand using the ruler of GIS software.

	corridor_name character varying (150)	country character varying (150)	length double precision
1	Sidi Kerir-Trieste	Egypt	507.8084726499041
2	Sidi Kerir-Trieste	Greece	1250.9891861406124
3	Sidi Kerir-Trieste	Croatia	609.1780308013689
4	Sidi Kerir-Trieste	Slovenia	21.789495495222408

Figure 39 Output for the corridor Sidi Kerir-Trieste.

The process was done with all the corridor provided which are the most critical ones, is important to mention that the longer routes like Corpus Christi in the United States to Italy present a lot of discrepancies in the distances around 4000 kilometers, this because the route the script calculates takes the shortest route possible, however after averaging the weights in the next steps this error is minimized and do not affect the final calculation of the corridor failure as it can be compared with table 5 and figure 40.

Corridor name	Type	Country crossed	Branch
Corpus Christi/Huston - TS	Open sea	Algeria	1159
		Bahamas	1232
		Croatia	638
		Cuba	653
		United States	473
		Portugal	1300

	Malta	198
	Mexico	550
	Morocco	440
	Spain	470
	Tunisia	399
	Open sea	5878
Captive Internal	United States	3136
Total		16526

Table 5 Corpus Christi-Trieste provided by EST- Lab.

	corridor_name character varying (150)	country character varying (150)	length double precision
1	Corpus Christi-Trieste	Portugal	1366.294364
2	Corpus Christi-Trieste	Bahamas	777.065408
3	Corpus Christi-Trieste	Mexico	200.311867
4	Corpus Christi-Trieste	Morocco	39.504059
5	Corpus Christi-Trieste	Tunisia	377.150706
6	Corpus Christi-Trieste	Croatia	609.178031
7	Corpus Christi-Trieste	Slovenia	21.789495
8	Corpus Christi-Trieste	Algeria	1073.514335
9	Corpus Christi-Trieste	Spain	680.536922
10	Corpus Christi-Trieste	United Kingdom	763.647594
11	Corpus Christi-Trieste	United States	1653.722935
12	Corpus Christi-Trieste	Open sea	4647.519046

Figure 40 Output for the corridor Corpus Christi-Trieste.

RouteName	SOVEREIGN1	length
Sidi Kerir-Falconara	Egypt	507.8084726
Sidi Kerir-Falconara	Greece	1232.853914
RouteName	SOVEREIGN1	length
Sidi Kerir-Sarroch	Egypt	509.698667
Sidi Kerir-Sarroch	Libya	248.956441
Sidi Kerir-Sarroch	Greece	676.7699798
RouteName	SOVEREIGN1	length
Teesport-Santa Panagia Bay	Portugal	955.4007137
Teesport-Santa Panagia Bay	United Kingdom	186.8081171
Teesport-Santa Panagia Bay	Morocco	16.56239298
Teesport-Santa Panagia Bay	Tunisia	377.1507058
Teesport-Santa Panagia Bay	France	878.0569159
Teesport-Santa Panagia Bay	Algeria	1073.514335
Teesport-Santa Panagia Bay	United Kingdom	1096.721524
Teesport-Santa Panagia Bay	Spain	1333.475145
RouteName	SOVEREIGN1	length
Antwerp-Trieste	Portugal	955.4007137
Antwerp-Trieste	United Kingdom	108.2930374
Antwerp-Trieste	Morocco	16.56239298
Antwerp-Trieste	Tunisia	377.1507058
Antwerp-Trieste	Belgium	107.5905428
Antwerp-Trieste	Netherlands	110.9384462
Antwerp-Trieste	Croatia	609.1780308
Antwerp-Trieste	Slovenia	21.7894955
Antwerp-Trieste	France	1169.821485
Antwerp-Trieste	Algeria	1073.514335
Antwerp-Trieste	United Kingdom	252.8640415
Antwerp-Trieste	Spain	1349.749938

Figure 41 Sample of the debug file of the intersection script.

Figure 41 contains a sample of the files created to debug the intersection of all the corridors as it is not feasible to test all of them, at least it is possible to track the traversing countries and make sense of them since the main corridors were already verified the assumption is that all the corridors' data is trustable.

The last script of this subsection is the calculation of the indexes and as like the last point the results were asserted with the ones provided. The values asserted were the geopolitical risk, with and without the piracy index involved, alpha normalizes coming from the straits, and the most important the corridor failure. Corridor failure considers all the data mentioned before as it is the last data of the chain.

Country	Corridor name	Failure
Egypt	Sidi Kerir-AU	0.454
	Sidi Kerir-LI	0.421
	Sidi Kerir-TS	0.373

Table 6 Probability of Failure of main Egypt Corridors provided by EST-Lab.

	corridor_name character varying (150)	corridor_failure_captive double precision
1	Sidi Kerir-Augusta	0.45744393510471826
2	Sidi Kerir-Trieste	0.3948183988091444

Figure 42 Probability of Failure of Egypt corridors calculated by the script.

Egypt and Turkey are important corridors, so the data was provided to assert them, it is possible to observe the values are not the same as the route's length differs but are on a safe range as well as the values for Turkey taking into consideration that Turkey has more the one pipeline, so the probability of failure also depends on it as it can be seen in table 7 and figure 43.

Country	Corridor Name	Probability of failure
Turkey	Ceyhan-AU via BTC	0.418
	Ceyhan-AU via ITP	0.427

Table 7 Probability of Failure of main Egypt Corridors provided by EST-Lab.

	corridor_name character varying (150)	name character varying (100)	corridor_failure_captive double precision
1	Ceyhan-Augusta	BTC	0.4132850154700437
2	Ceyhan-Augusta	ITP	0.4212464318430843

Figure 43 Probability of Failure of Ceyhan corridor calculated by the script.

5.3. Web Application

The Web application was tested in three different browsers to check compatibilities issues such as Brave, Google Chrome, and Mozilla Firefox and not issues find the application renders and run correctly.

The next section contains the test divided by areas done to the application.

5.3.1. User Interface

The user interface was tested as if a normal user is using it, all the navigation links were tested to find missing links or loops, and if the navigation is intuitive itself.

All the navigation is working and there is no redundancy of the links nor any broken link now in the pricing section the purchase buttons do not have any link associated because the feature is not yet implemented, and the page is used as a placeholder for future development.

When the user successfully logs the home page immediately change the login text for log out and perform such action.

All the stories have interactive features where the user can select some features, the selection is limited to dropdown menus and there is no keyboard input from the users. Each option of the dropdown menu comes from a query so the input will exist and there is no way the user can insert its input.

When a selection is done the side panel will be automatically updated and if the data is not present on the database it will show the name of the selection and a 0, if it is a plot, it shows an empty placeholder with the message there is no data for the selection.

The user can interact with the plots, the interactive comes from the library plotly with by itself if robust and will guarantee the functionality of its features.

5.3.2. User Authentication

For user authentication, the built-in system of Django was used which guarantee validation of the user inputs as well as security for attacks as SQL injections. The user authentication system also oversees validation of the user password like if it is very short or common.

The sign-up page form, the user cannot submit a profile if all the fields are not filled, otherwise, an alert will be rise, the email is check and email verification mail is sent to the user to activate the account if the password does not match another error will be raised. Also, Django is in charge of hashing the password with a salt and storing it on the database, in case there is a data-leak, or some tries to still credentials, the hashing algorithm used is the Password-Based Key Derivation Function 2.

The mail verification was tested several times as a user that is requesting a new password or when the account is created, the token generated by the framework works all the time and there are no issues with the creation of the mail itself and the token. For this validation as there is no SMTP service active all the emails are sent to the backend console as pure text.

As it is possible to see in table 6 for this test user created to test the platform the password is hash and with the salt, it is almost impossible to brute force the plain text password.

PK	Password	username	Name	Surname	email
5	pbkdf2_sha256\$260000\$JDZkfl9d9vXTu90toFlzx8\$B3C0l/k9xfq2tlnVDwwcDNzFk/IDJ4H1TEVbTtZcEM=	vitoquesito@hotmail.com	Vito	Quesito	
10	pbkdf2_sha256\$260000\$uhFmtmequ9ooN5TrUyylv6\$bjo2mZLqKjgtNZue7nJha6xX7VPyjNkRKzniakbdFDY=	admin			
15	pbkdf2_sha256\$260000\$ZxoaTxUBBd6DBkQGxtEp9j\$zaA3mrRx1MDzvv1N3jqWfwEJcdNRp6+/gJRSI1rhM04=	mmarta@gmail.com	Maria	Marta	
18	pbkdf2_sha256\$260000\$BcyfYgvvOb6fqEkSHlrWj\$YrIPBNWcx0Dasw9+ya5xhAMNJ4rhInhxExaF4tqOO/J0=	mario_barrio@gmail.com	Maria	Barrio	
19	pbkdf2_sha256\$260000\$mdM24T6BrJIlogp41C0cse\$aNHGqRH0Hwz0DDH2ftq/CEcouDlNks9Ve2P2vspJFw=	jairo_quesito@hotmail.com	Jairo	Quesito	jairo_quesito@hotmail.com
20	pbkdf2_sha256\$260000\$oy3C3lp1Yzw0s9DAbCU2My\$CfNdAmvqBSNNdb8B1V6UJxyHjFJZRHQ6NHsRmqW48=	esteban.boh@gmail.com	Esteban	Boh	esteban.boh@gmail.com

Table 8 Database table for Users

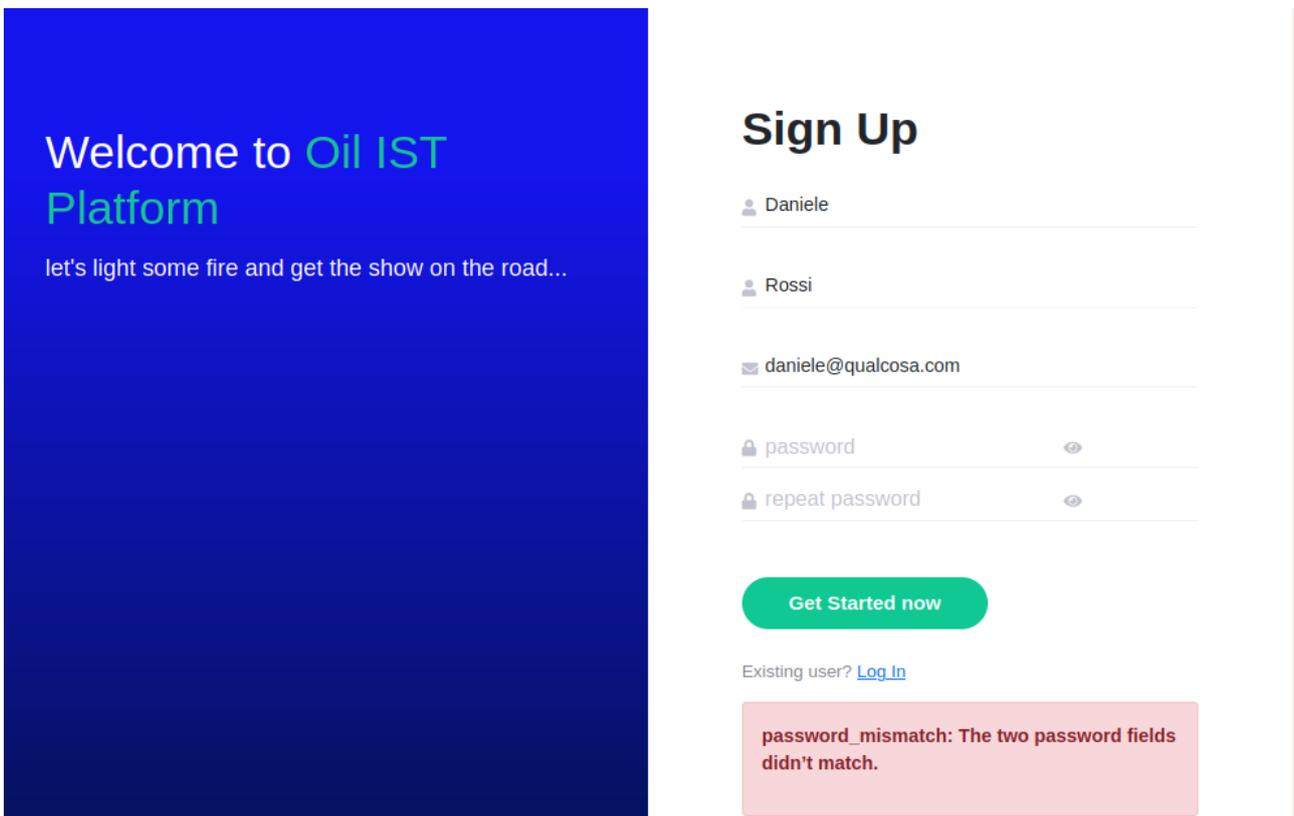


Figure 44 Password mismatch error

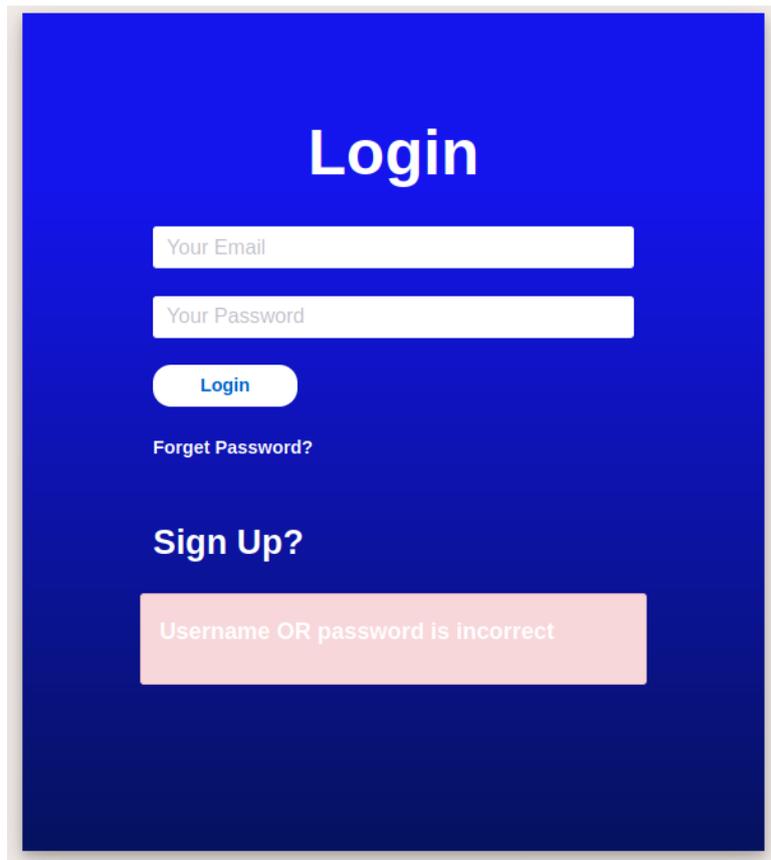


Figure 45 Email or Password Log In fail

5.3.3. Models and Queries

The queries were done with the ORM provided by Django. An inspection of the model file was done to determine if the import of the model was done correctly with all the tables and attributes (primary key, foreign, key, unique constraint, etc.)

In the module, `utils_query.py` are functions with the most common queries used in the project, the object "QueryObject" return by the application was transformed to a DataFrame which is easy to work with, the resulting DataFrames were confronted with the normal SQL commands typed on the PostgreSQL console to determine the correctness of the data provided by Django.

The queries are simple and generic because are used in more the one template and the transformation is done using Pandas methods. All the transformations were inspected manually to guarantee the integrity of the calculations.

6. Conclusions

Data storytelling surrounds data with a narrative for a better understanding of it, as it combines visualization, interactions, and storytelling to influence or enhance audience engagement, and these elements can directly affect future decisions. It can be a great technique and an integral part of the future decision-making process, through its clean, interactive, and impactful body of work.

This thesis focuses on the problem of oil supply in Italy by providing energy stakeholders with an Oil IST (Interactive Storytelling) scientific tool that is easy to use and accessible, capable of transforming raw data from different sources, processing it, calculate the EST Oil Risk indicators and, in the end, offer an attractive narrative that helps to better understand what is happening in the context of oil supply to make faster and better decisions.

The careful design of the architecture and then the actual implementation of the application resulted in the working prototype of a platform for data storytelling supporting the oil risk supply data. The application can be further enhanced and transformed into a production-ready system.

To conclude, the application was successful. It was designed and implemented, and every part of the process was described in the thesis.

REFERENCES

- [1] E. Desogus, *Modelling the role of oil in the Italian energy security*, Politecnico di Torino, 2020.
- [2] D. Kaufmann and A. Kraay, "Worldwide Governance Indicators," 2020. [Online]. Available: <https://info.worldbank.org/governance/wgi/Home/downloadFile?fileName=wgidataset.xlsx>. [Accessed 10 October 2020].
- [3] S. Kahur, "What is Data Storytelling?," 28 August 2021. [Online]. Available: <https://hackr.io/blog/what-is-data-storytelling>.
- [4] S. Tracy, "5 Examples of Awful Data Visualization," Analytical Blog, [Online]. Available: <https://analytical.com/blog/examples-of-awful-data-visualization>. [Accessed 15 May 2020].
- [5] C. D. Stolper, B. Lee, N. H. Riche and J. Stasko, "Emerging and recurring data-driven storytelling techniques: Analysis of a curated collection of recent stories.," 2016.
- [6] V. Analytics, "The Art of Storytelling in Analytics and Data Science," Vidhya Analytics, 8 May 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/05/art-storytelling-analytics-data-science/>. [Accessed 6 June 2020].
- [7] A. Ojo and B. Heravi, "Patterns in award winning data storytelling: Story types, enabling tools and competences.," *Digital journalism*, pp. 693-718, 2018.
- [8] U. Erikson, "Functional vs Non Functional Requirements," 5 April 2012. [Online]. Available: <http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>.
- [9] "Use Case Diagram," [Online]. Available: <https://www.smartdraw.com/use-case-diagram/>. [Accessed 18 June 2021].
- [10] "Use cases," Usability Gov, [Online]. Available: <https://www.usability.gov/how-to-and-tools/methods/use-cases.html>. [Accessed 18 June 2021].
- [11] S. Li, "Most People Can't Be Bothered to Download Apps," The Atlantic, [Online]. Available: <https://www.theatlantic.com/technology/archive/2014/08/most-people-cant-be-bothered-to-download-apps/378989/>. [Accessed 5 June 2021].
- [12] S. Jablonski, I. Petrov, C. Meiler and U. Mayer, *Guide to Web Application and Platform Architectures*, Heidelberg: Springer, 2004.
- [13] "MDN Web Docs," [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server. [Accessed 4 May 2021].
- [14] "Framework," Techterms, 7 March 2013. [Online]. Available: <https://techterms.com/definition/framework>. [Accessed 10 June 2021].

- [15] "Django Architecture," Data Flair, [Online]. Available: <https://data-flair.training/blogs/django-architecture/>. [Accessed 6 November 2020].
- [16] "Pandas," Pandas, 25 January 2019. [Online]. Available: <https://pandas.pydata.org/pandas-docs/version/0.24.0/>. [Accessed 10 June 2020].
- [17] "What is Numpy," Numpy Org, [Online]. Available: <https://numpy.org/doc/stable/user/whatisnumpy.html>. [Accessed 10 June 2020].
- [18] "Bootstrap," What Is, [Online]. Available: <https://whatis.techtarget.com/definition/bootstrap>. [Accessed 8 June 2021].
- [19] "JavaScript," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/javascript>. [Accessed 8 June 2021].
- [20] "jQuery," [Online]. Available: <https://jquery.com/>. [Accessed 8 June 2021].
- [21] "What is Ajax," w3schools, [Online]. Available: https://www.w3schools.com/whatis/whatis_ajax.asp. [Accessed 8 June 2021].
- [22] "Introduction of Ajax," [Online]. Available: http://www.httpdebugger.com/articles/introduction_of_ajax.html. [Accessed 8 June 2021].
- [23] L. P. Posadas, *SICTE-11 Database Management System*, School of Information and Knowledge Management, 2016.
- [24] "PostgreSQL," [Online]. Available: <https://www.postgresql.org/about/>. [Accessed 8 June 2021].
- [25] "Legacy Databases," Django Docs, [Online]. Available: <https://docs.djangoproject.com/en/3.2/howto/legacy-databases/>. [Accessed 15 June 2021].