People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of the

MASTER

In **Computer Engineering**

Option: **Computer Engineering**

Title:

Design and Implementation of a Web-based Geographical Information System

Presented by:

- **MEDDANE Sara**

Supervisor: **Dr. ZITOUNI Abdelkader**

Co-supervisor: **Ms.OURAGH Nawel**

Registration Number:181831034260/2023

Abstract

This study deals with the design and implementation of a web-based geographical information system for the management of hydrocarbon exploration sites(latter on refereed to as ”acreages”), at the level of the Exploration Division, SONATRACH, that should help the involved stakeholders to efficiently manage and exchange information.

This system has been designed using Unified Modelling Language (UML) and a conceptual relational database model, after understanding the users specifications and requirements. As for the implementation, we have used a set of tools in particular JavaScript and Leaflet for the client side and MySQL and Laravel, a PHP based framework, for the server side.

Dedication

I dedicate this modest work to my dearest parents,

To all my family members,

To all my teachers,

To my friends and all those who believed in me and encouraged me.

Sara

Acknowledgements

First of all, Praise is due to Allah, for leading me during all these years of study.

I would like to thank my university supervisor Dr.ZITOUNI Abdelkader for his support and valuable feedback throughout the period of this project.

I would like also to thank my co-supervisor Mrs.OURAGH Nawel, an IT engineer at SONATRACH, for allowing me to have this new experience, and for her kind co-operation and guidance throughout the period of the internship.

My thanks and appreciations also go to all the people who contributed directly or indirectly in this work.

Contents

General Introduction 1

1 Introduction to GIS **3**

1.1 Introduction 3

1.2 Definition 3

1.3 The development of GIS 3

1.3.1 Geospatial analysis 3

1.3.2 The use of computers in cartography 4

1.3.3 The first operational GIS 5

1.4 Web GIS 5

1.5 Web GIS components 8

1.5.1 Client's device 8

1.5.2 Web server 8

1.5.3 Database 11

1.5.4 Map server 12

1.6 Applications of GIS 13

1.6.1 Health Sector 13

1.6.2 Disaster Management 13

1.6.3 Climate Change 13

1.6.4 Oil and Gas Industry 13

1.7 Conclusion 14

2 System Design **15**

2.1 Introduction 15

2.2 Objective and System requirements 15

2.3 Specifications of the management of exploration acreages 16

2.3.1 Acreage report contents 16

2.3.2 Staff role identification 18

2.4 Unified Modelling Language . . . 18
2.5 Use case diagrams . . . 19
2.5.1 System actors and use cases . . . 19
2.5.2 Planning's use case diagram . . . 20
2.5.3 Data Management use case diagram. . . . 24
2.5.4 Asset's use case diagram . . . 26
2.5.5 Finance's use case diagram . . . 28
2.5.6 Manager's use case diagram . . . 30
2.5.6.1 Consult the overall data of an acreage from the map: . . . 30
2.5.7 Global use case diagram . . . 31
2.6 Activity diagram . . . 32
2.7 Sequence diagrams . . . 33
2.7.1 Sequence diagram 01: Authentication . . . 33
2.7.2 Sequence diagram 02: Adding a new acreage to the database . . . 33
2.7.3 Sequence diagram 03: Add a new acreage to the map . . . 34
2.7.4 Sequence diagram 04: Validate realizations . . . 35
2.7.5 Sequence diagram 05: Consult the acreage data from the map . . . 36
2.8 Conceptual database design . . . 36
2.8.1 Entity relationship diagram of the system . . . 38
2.9 Class diagram . . . 38
2.10 Conclusion . . . 40

3 Implementation and Testing **41**
3.1 Introduction . . . 41
3.2 Front-end Tools . . . 41
3.2.1 HTML . . . 41
3.2.2 CSS . . . 42
3.2.3 Bootstrap . . . 42
3.2.4 JavaScript . . . 42
3.2.5 Leaflet . . . 43
3.2.6 Chart.js . . . 47
3.3 Back-end Tools . . . 47
3.3.1 Laragon . . . 47
3.3.1.1 Apache HTTP Server . . . 48

3.3.1.2 MySQL . . . 48
3.3.1.3 HeidiSQL . . . 49
3.3.2 Laravel . . . 49
3.3.2.1 PHP . . . 49
3.3.2.2 What is Laravel? . . . 50
3.3.2.3 MVC architecture . . . 50
3.3.2.4 Laravel's Directory Structure . . . 52
3.4 System diagrams . . . 55
3.4.1 General diagram of the system . . . 55
3.4.2 System diagram with the Leaflet component . . . 56
3.5 System Testing . . . 57
3.5.1 Authentication page . . . 57
3.5.2 "Home" page, profile: Planning . . . 58
3.5.3 Adding a new acreage . . . 59
3.5.4 "Initialize acreage" Search page . . . 60
3.5.5 Initializing the contractual situation . . . 61
3.5.6 Initializing the contractual commitments . . . 62
3.5.7 Adding a new acreage to the map . . . 63
3.5.8 Modifying/deleting acreage's information . . . 64
3.5.9 Validation of realizations . . . 65
3.5.10 Consulting the general information of an acreage . . . 66
3.5.11 View or print the pdf version of the general information of the acreage . . . 67
3.6 "Statistics" page . . . 68
3.7 Conclusion . . . 69

General conclusion **70**

References . . . 77

A Company Overview **78**

B Assets, Blocs and Basins **81**

C UML diagrams **82**
C.1 Use case diagrams . . . 82
C.2 Activity diagrams . . . 83
C.3 Sequence diagrams . . . 84

C.4 Class Diagram 85
C.4.1 Visibility 85
C.4.2 Relationships between classes 85

D Relational database model **87**
D.1 Types of relationships in a relational database 87
D.2 Advantages of a relational database 88

E Listings **89**
E.0.1 89
E.0.2 89
E.0.3 90
E.0.4 91
E.0.5 92

List of Tables

2.1 System actors and their use cases. 19
2.2 Authentication use case textual description. 21
2.3 ”Add a new acreage to the database” use case textual description. 21
2.4 ”Initialize the contractual situation” use case textual description. 22
2.5 ”Add a new realization” use case textual description. 22
2.6 ”Add a new request or amendment” use case textual description. 23
2.7 ”Add 2023 program and 2024-2028 MTP” use case textual description. 23
2.8 ”Add an acreage to the map” use case textual description. 25
2.9 ”Add acreage map files” use case textual description. 25
2.10 ”Initialize the geographical information” use case textual description. 26
2.11 ”Initialize the petroleum system” use case textual description. 27
2.12 ”Add volumes in place” use case textual description. 27
2.13 ”Validate realizations” use case textual description. 28
2.14 ”Enter the costs of a realization” use case textual description. 29
2.15 ”Consult the up to-date data of an acreage from the map” use case textual description. . 30

List of Figures

1.1 Xerox PARC Map Viewer 1993 [15]. 7
1.2 General Structure of a Web GIS Application [18]. 8
1.3 client-server architecture [22]. 9
1.4 URL components [26]. 10
1.5 HTTP request components [28]. 10
1.6 Raster Vs Vector data [31] 11
1.7 Pyramid structure of vector tiles with zoom-levels [33]. 12

2.1 Planning's use case diagram. 20
2.2 Data Management use case diagram 24
2.3 Asset's use case diagram. 26
2.4 Finance's use case diagram. 28
2.5 Manager's use case diagram. 30
2.6 Global use case diagram. 31
2.7 Activity diagram of the system. 32
2.8 Authentication's sequence diagram. 33
2.9 "Add a new acreage" sequence diagram. 34
2.10 "Add a new acreage to the map" sequence diagram. 34
2.11 "Validate" realizations sequence diagram. 35
2.12 "Consult the acreage data" sequence diagram. 36
2.13 Entity relationship diagram of the system. 38
2.14 Class diagram of the system. 39

3.1 Leaflet logo 43
3.2 Map of Algeria from OSM with a polygon at zoom level 5. 45
3.3 Offline map of Algeria. 46
3.4 Laragon main UI, with quick toggles, server status, and a menu. 48

3.5 Apache Server logo. 48
3.6 HeidiSQL logo. 49
3.7 Interaction of Apache, MySQL and PHP in the server side with the clients[66]. 49
3.8 An SQL query using the select method of the DB facade. 50
3.9 MVC design pattern [74]. 51
3.10 Example of simple route that triggers the index function in the HomeController. 51
3.11 The handle method that verifies the user's profile. 52
3.12 A route with middleware 53
3.13 Configuring the database connection in .env. 54
3.14 General diagram of the system. 55
3.15 Leaflet with MVC. 56
3.16 "Authentication" page. 57
3.17 "Home" page 58
3.18 "Add a new acreage" page. 59
3.19 "Initialize contractual situation and commitments " Search page. 60
3.20 "Initialize contractual situation" page. 61
3.21 "Initialize contractual commitments" page. 62
3.22 "Add acreage to the map" page. 63
3.23 "Search existing petroleum system" page 64
3.24 "Edit petroleum system" page. 64
3.25 "Validate 2D Seismic realization" page. 65
3.26 "Acreage's general Information" page. 66
3.27 "Print general information section of the fact-sheet" page. 67
3.28 "General statistics" page. 68

A.1 Organization Chart of the Exploration Division. 79

C.1 Basic components of activity diagram. 84
C.2 Relationships in a class diagram 86

D.1 Database relationships symbols in ER Diagram 88

List of Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CGIS	Canada Geographic Information System
CRS	Coordinate Reference System
CSS	Cascading Style Sheets
DBMS	Database Management System
DNS	Domain Name System
DOM	Document Object Model
Esri	Environmental Systems Research Institute
EPSG	European Petroleum Survey Group
GIS	Geographical Information System
GUI	Graphical User Interface
G&G	Geological and Geophysical
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines
JSON	JavaScript Object Notation
MTP	Medium-term Plan
OOP	Object-oriented Programming
OS	Operating System
OSM	OpenStreetMap
PARC	Palo Alto Research Center
SQL	Structured Query Language
UML	Unified Modelling Language
URI	Universal Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

General Introduction

As we are now living a rapid technological change that affects almost all facets of life, business organizations of every size and from different sectors tend to integrate digital technologies and solutions to better manage their data and resources.

One of these digital solutions is the information system. An information system is an integrated set of components for collecting, storing, and processing data and for providing information, knowledge, and digital products. Business firms and other industrial organizations rely on information systems to carry out and manage their operations, interact with their customers and suppliers, and compete in the marketplace [1].

The oil & gas industry sector is not an exception. This industry generates large volumes of data. This includes data related to hydrocarbon exploration sites, latter on referred to as ”acreages”.
”Acreage” means an area of land [2]. This term is most commonly used to describe lands covered by a lease granted for oil and gas exploration and for possible future production [3]. The management of hydrocarbon exploration acreages is carried out through monthly PowerPoint reports, at the level of the Exploration Division which falls under the national state-owned oil and gas company of Algeria SONATRACH, where the internship took place (see Appendix A).

These reports contain all the needed acreage information (tabular data, maps, statistics) and are filled by different directorates of the Exploration Division, such that each directorate has associated parts to fill. These reports are prepared monthly for managers to help them in monitoring the overall performance of exploration in a given acreage. This management reporting method has several drawbacks, in particular:

- Poor communication between stakeholders: synthesizing data coming in from various directorates and stakeholders can become an unmanageable task.
- Lack of real time information: getting access to the latest information requires access to the latest version of the report. This can be time-consuming, especially when many stakeholders from various directorates are involved. Instead of analysing the data, a stakeholder might find himself spending more time searching for the latest data or talking to partners.
- Lack of relevance and validation over entered data.

In this work, we aim to design and implement a web-based geographical information system (GIS), which is a special class of information systems, to improve the management of exploration acreages. The exploration acreages should be visualized on the geographic map of Algeria and linked to the attribute

data inserted by the stakeholders.

Organization of the report :

- The first chapter covers a general overview of GIS, its development over time, Web-GIS, Web-GIS components, and some examples of the applications of GIS.
- The second chapter focuses on the system specifications and requirements. This includes identifying the system users (actors) and their roles and use cases. UML diagrams were also used to visualize how the system works. This chapter also includes an estimation of the database schema which is also based on the system requirements.
- The third chapter describes all the tools (front-end and back-end) used in implementing this system : programming languages, frameworks, libraries, DBMS, and other technologies. It describes as well how these tools work and how they interact to make up the system.
- The last chapter presents the results of this work. It illustrates some interfaces available to the system users, as well as a discussion of each interface.

Chapter 1: Introduction to GIS

1.1 Introduction

Today, cartography has evolved significantly with the rise of geospatial technology. The establishment of computer cartography led to the development of a new class of information systems which is geographical information systems. This chapter addresses an overview of geographical information systems. This includes a historical background on GIS in general and Web-GIS in particular, the components of Web-GIS and how it functions.

1.2 Definition

A Geographical Information System, or GIS, is a special class of information systems that creates, manages, analyses, and maps various types of data. It links maps to databases, integrating geospatial data with all types of descriptive information [4].

A GIS is composed of hardware, software, data, personnel, and a set of organizational protocols. These components must be well integrated for effective use of GIS, and the development and integration of these components is an iterative, ongoing process [5].

1.3 The development of GIS

1.3.1 Geospatial analysis

Many of the geographic concepts and techniques that GIS automates date back to the 1830s. In 1834, French geographer and cartographer Charles Picquet created an early thematic map, in which the 48 districts of Paris were represented by colour gradient according to the percentage of deaths from cholera per 1000 inhabitants [6].

On August 31, 1854, an outbreak of cholera hit the London district called Soho. Over the course of three days, 127 people died from the disease and by September 10, over 500 had died. English physician

John Snow, commissioned cartographer Charles Cheffins, to create a map that mapped out the locations of the local water wells and cholera deaths, by identifying the relationship between them, Snow was able to determine the source of cholera which was a water pump on Broad Street. [7]

1.3.2 The use of computers in cartography

In 1959, Waldo Rudolph Tobler - American-Swiss geographer and cartographer - pioneered the use of computers in cartography in his paper "Automation and Cartography" [8]. According to Tobler's paper, a general data processing system consists of four major steps : gathering of the data, manipulation, storage, and utilization. Further, data manipulation can be subdivided into the components of input, memory, arithmetic, output, and control. By inserting the map where applicable, we can get three possible systems of cartography [8]:

a.The map as a data-storage element

Data gets extracted from the map and translated into some symbology that is machine-readable, and then operated upon by the data manipulation unit. A good example would be the recording of positional data in terms of some coordinate system, punching this information on Hollerith-type cards and feeding the cards to a data manipulation system. This is done by an equipment often referred to as data reduction equipment; it extracts information from graphs or photographs of oscilloscope traces (maps), and then automatically prepares punched cards or tapes.

b.The map as input to a data manipulation system

This is done by transferring the map data directly into a computation system. The map might be fed into a slot, processed, then discarded, its entire informational content having been stored in the memory element of the machine. Although many techniques were in use for the wholesale transformation of maps into optical, electrical, magnetic, or other images, the conceptual problems of machine pattern recognition have not been solved, at that time.

c.The map as a data-processing output

Most of the available data-processing output units at that time, such as radar mapping which makes use of a cathode-ray tubes, were able to provide information for viewing or reading, and as maps are visual displays, it follows that they could be prepared with this equipment.

1.3.3 The first operational GIS

In the early 1960s, English-Canadian geographer Roger Tomlinson and the aerial company he was working at (Spartan Air Services), were asked to develop a map for site-location analysis in an east African nation.

Tomlinson suggested applying the new automated computer technologies to accomplish such a task more efficiently than humans. Eventually, **Spartan** met with **IBM** offices to begin developing a relationship to bridge the previous gap between geographic data and computer services.

Tomlinson contributed geographic knowledge, whereas IBM contributed computer programming and data management. The result of this collaboration was the Canadian Geographic Information System **(CGIS)**. This is believed to be the first truly operational GIS. The system was used to store, analyse, and manipulate data collected for the Canada Land Inventory to determine the land capability for rural Canada by mapping information about soils, agriculture, recreation, wildlife, waterfowl, forestry and land use [9].

In 1969, Jack Dangermond—a member of the Harvard Lab—and his wife founded the Environmental Systems Research Institute (Esri); which is an American multinational GIS software company. As computing became more powerful, Esri improved its software tools. Working on projects that solved real-world problems led the company to innovate and develop robust GIS tools and approaches that could be broadly used. In need of analysing an increasing number of projects more effectively, Esri developed **ARC/INFO**—the first commercial GIS product. The technology was released in 1981 and began the evolution of Esri into a software company [10].

ARC/INFO was originally a command-line based system. The first desktop **GUI GIS** product however, was developed in 1986 for **DOS** operating system, by American software company **Precisely Holdings**. This was renamed in 1990 to **MapInfo** for Windows when it was ported to the Microsoft Windows platform [11]. Today Esri is the leading supplier of GIS software, and it is best known for its ArcGIS products.

The first major **web mapping program** capable of distributed map creation Xerox PARC MapViewer appeared in 1993, developed by Steve Putz.

1.4 Web GIS

Web GIS is an architectural approach, for implementing a modern GIS. It is powered by standard web services that deliver data and capabilities, and connect network components [12].

Often the terms **Web-GIS** and **web mapping** are used synonymously, despite their distinct meanings.

Web mapping is the process of designing, implementing, generating, and delivering maps on the World Wide Web. While web mapping primarily deals with technological issues, web cartography additionally studies theoretic aspects: the use of web maps, the evaluation and optimization of techniques and workflows, the usability of web maps, social aspects, and more.
Web Geographical Information Systems or Internet GISs are related to web mapping but with an emphasis on analysis, processing of project-specific geodata, as well as exploratory aspects. To put it bluntly, it is some type of Internet application that makes use of a map [13].

The first Web GIS **Xerox PARC Map Viewer**, see Figure 1.1, appeared in June 1993, after 3 years of the creation of the World Wide Web. This web mapping program was developed by Steve Putz at Xerox Corporation's Palo Alto Research Center (PARC), using a custom server module written in the Perl scripting language.

This software application combined the ability of HTML documents to include graphical images - stored in a geographic database - with the ability of HTTP servers to create new documents in response to user input. The HTML documents and the HTTP protocol were used to provide a custom user interface for browsing and viewing geographic map data. A map server was used to both generate and interpret the map URLs.

In the main mapping web page, the user simply enters a search query (such as the name of a city or a country) and a list of matching places is returned as a formatted HTML document. Selecting from the list generates another HTML document consisting of two maps (small and large scale) with the location highlighted.Additional links in the document were used as controls allowing the user to change various map rendering options (pan, zoom, level of detail) [14].

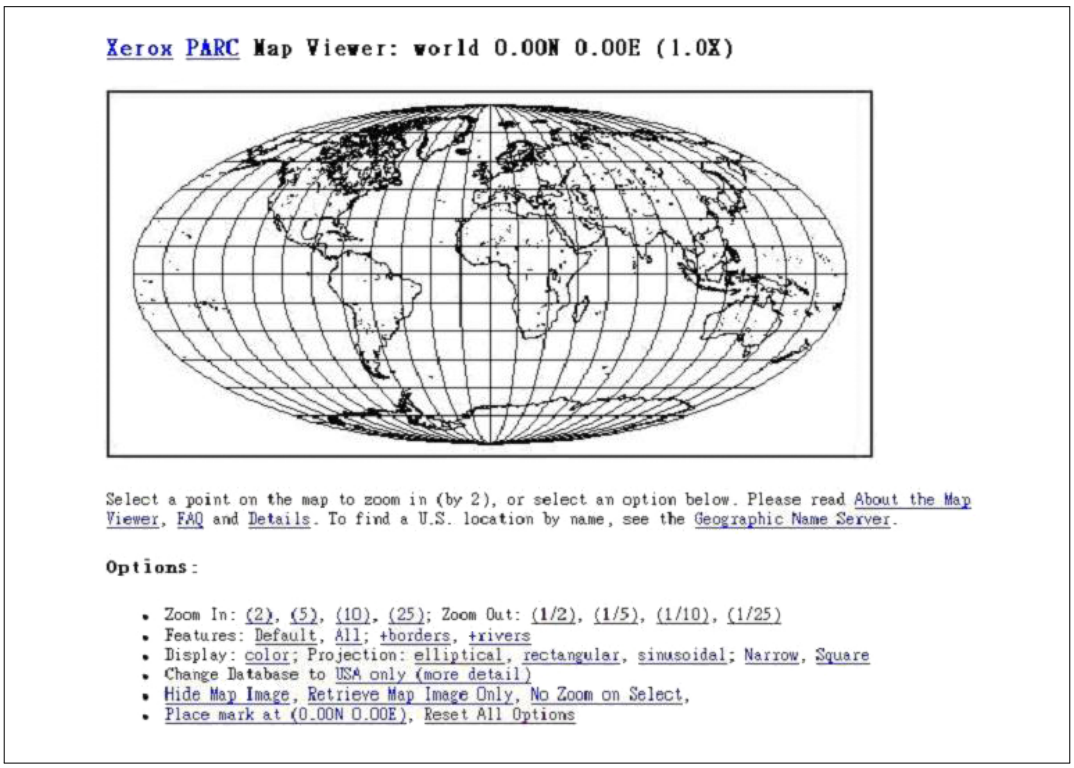


Figure 1.1: Xerox PARC Map Viewer 1993 [15].

More advanced web GIS applications were developed, such as MapServer in 1994, MapQuest in 1996 and Google Maps in 2006. At the end of 1999, Esri released the first phase of the ArcGIS system, a new integrated architecture for desktop GIS products.The 2012 release of ArcGIS 10.1 provided an integrated GIS platform that enabled greater collaboration, including a fully integrated ArcGIS Online[16]. This software simplified the job of creating interactive, web-based maps that geo-enable stories and events anywhere in the world [17].

1.5 Web GIS components

Figure 1.2 illustrates the general structure of a Web GIS :

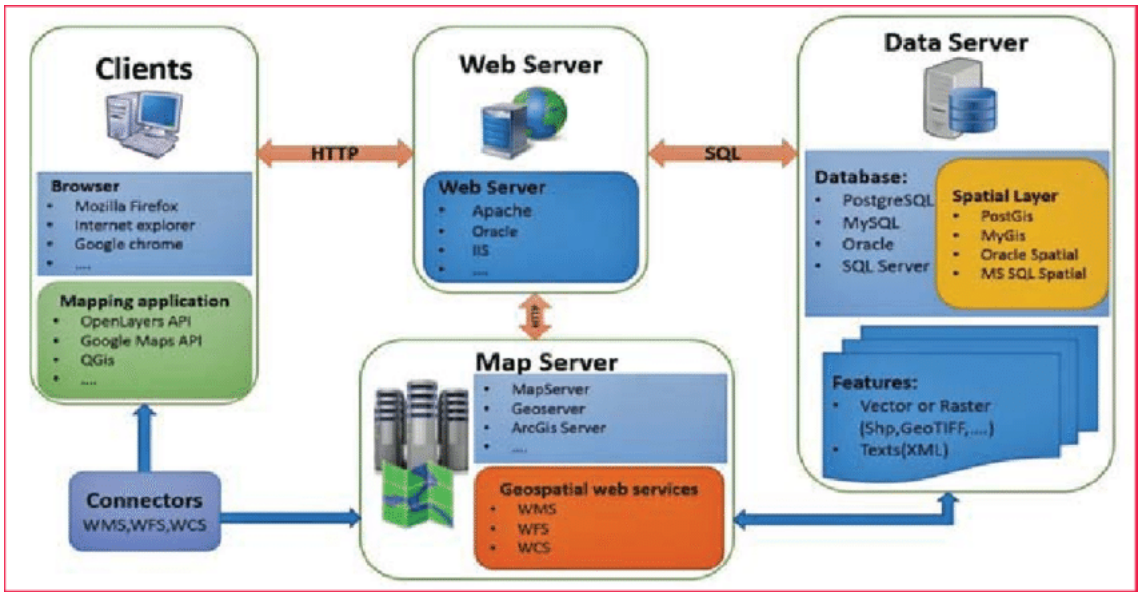


Figure 1.2: General Structure of a Web GIS Application [18].

1.5.1 Client's device

The user's device, specifically the web pages displayed on the web browser, represents the client side of the system. The source code for most web pages is hypertext markup language (HTML). The cascading style sheets (CSS) are used to improve the appearance, style, and layout of pages coded in HTML .

A scripting language, such as JavaScript, is often used to make the web pages dynamic and interactive. Mapping Application programming interfaces (APIs) can call modules and implement their classes in the application. ArcGIS JavaScript API, Google Maps API and Leaflet API, are popular APIs for the development of interactive web mapping applications, offering a wide range of spatial data web services, providing a wide range of geospatial web services, analytics, and processing [19].These APIs are used for presenting and manipulating the map data, but they do not provide the base map layer, nor the data.

1.5.2 Web server

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web or over an intranet. The main job of a web server is to display website content through storing, processing and delivering webpages to users [20].

In Web-GIS architecture, the web server is also communicating with the server-side GIS component (generally a map server). This is adding spatial analysis functionality to the system [21].

- **HTTP and HTTP servers**

  Web server software is accessed through the domain names of websites. It is comprised of several components, with at least an HTTP server. The HTTP server is able to understand HTTP requests and URLs. When a web browser, like Google Chrome, needs a file that is hosted on a web server, the browser will request the file by HTTP. When the request is received by the web server, the HTTP server will accept the request, find the content and send it back to the browser through HTTP [20], see Figure 1.3.

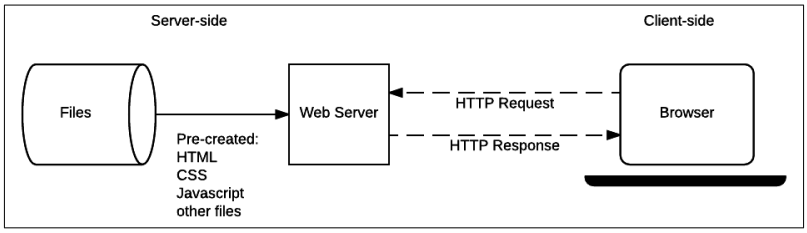


Figure 1.3: client-server architecture [22].

- **URL**

  URL stands for Uniform Resource Locator. It is a specific type of URI (Universal Resource Identifier). A **URI** is defined as any character string that identifies a resource. A **URL** is defined as those URIs that identify a resource by its **location** or by the means used to access it, rather than by a name or other attribute of the resource [23]. A URL for HTTP is normally made up of three or four components [23]:

  - **A scheme:** identifies the protocol to be used to access the resource on the Internet. It can be HTTP or HTTPS. HTTPS stands for Hypertext Transfer Protocol Secure, which is an HTTP protocol that secures communications with encryption [24].
  - **A host:** The host name (or domain name) identifies the host that holds the resource, for example, www.quora.com. Host names can also be followed by a port number. A port number uniquely identifies a network-based application on a computer. Each application/program is allocated a 16-bit integer port number. This number is assigned automatically by the OS, manually by the user or is set as a default for some popular applications [25].
  - **A path:** identifies the specific resource in the host that the web client wants to access.
  - **A query string:** If a query string is used, it follows the path component, and provides a string of information that the resource can use for some purpose (for example, as parameters for a search or as data to be processed). The query string is usually a string of name and value pairs; for example, term=bluebird.

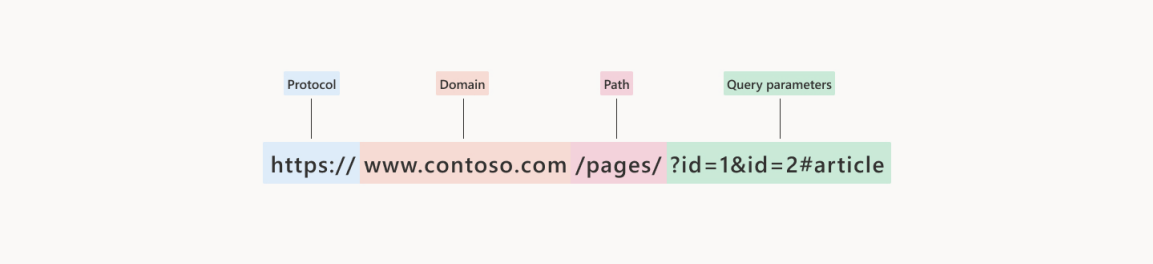


Figure 1.4: URL components [26].

- **HTTP request**

  An HTTP request is made from a client to a host located on the server in order to receive a resource needed to build the content. When they make a request, clients use a URL that contains the information needed to access the server resources [27]. An HTTP request is made out of three components [27]:

  - **Request Line:** Its purpose is to start the action on the server. It includes the following elements:
    - An HTTP method : GET, POST, PUT...
    - The request-target which can be a URI or URL to either a path or a protocol.
    - The HTTP version that defines the structure of the remaining message.
  - **Header:** allows for additional information to be passed between server and client such as cookies, information about the authorization token, or user agent using a special string that helps the server to identify client browser and OS version..
  - **Message Body:** the server uses the message body to deliver the information back to the client. The message body contains the information, the request line, headers, an empty line, and the message body that is optional.

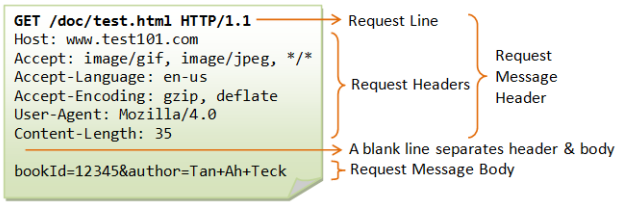


Figure 1.5: HTTP request components [28].

1.5.3 Database

A database is a set of **persistent** data that is used by the application systems of some given organization. A database management system DBMS, however, is a computerized system or a computer program that is used to store information and to allow users to retrieve and update that information on demand using a query language such as SQL [29].

The data in the database is said to be persistent because once it has been accepted by the DBMS for entry into the database in the first place, it can subsequently be removed from the database only by some explicit request to the DBMS, not as a mere side effect of, for instance, some program completing execution [29].

The database module is setup on the data server to store GIS data. GIS data is divided into two categories :

- **Spatial data**

  Represents features having known locations on earth. Spatial data can be further classified into two different types:

  - **Vector data:** uses X,Y coordinates, to represent points, polylines and polygons (areas). An advantage this has is that less storage space is used and it can be easier to combine different vector layers [30].
  - **Raster Data:** is data that is presented in a grid of pixels. Each pixel within a raster has a value, which can be either a colour or a unit of measurement. This communicates information about the element in question. Rasters typically refer to orthoimagery which are photos taken from satellites or other aerial devices [31].

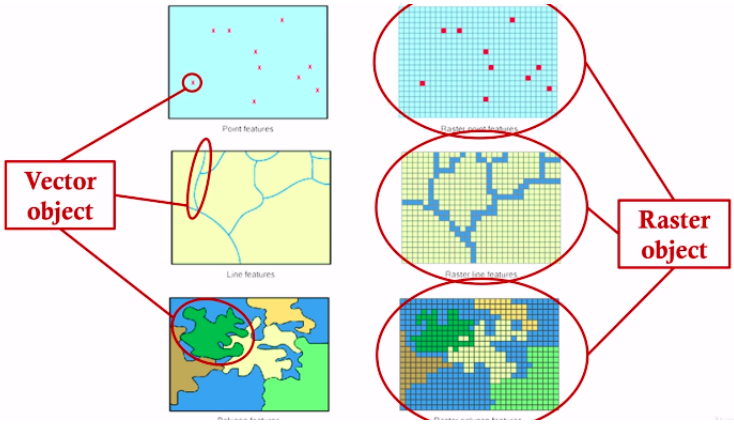


Figure 1.6: Raster Vs Vector data [31]

- **Attribute data** Represents non-graphic information linked to the geographical features, describing the properties of these features, such as the type of roads, the name of a town, the history of an area. Attribute data are stored in attribute tables that are connected to the spatial data [30].

1.5.4 Map server

Map servers or spatial servers or map tile servers are web servers which have an API for returning tiles of map imagery (either vector or raster) [32]. There is a grid of map tiles for each zoom level as it is described by Figure 1.7. Due to the large file sizes, latencies, and data caps, most software applications which provide map services do not download or store a copy of all of the map data. Alternatively, as a user pans and zooms on a map, requests for square tiles of map imagery at the user's current zoom level and area-of-interest are sent to the map server. . The map server either has pre-rendered, cached copies of these tiles or renders them spontaneously, compresses them, and sends them back to the user (the server typically sends back compressed .png or .jpeg files). The software application then inserts these tiles into the map providing a seamless browsing experience to the user [32].

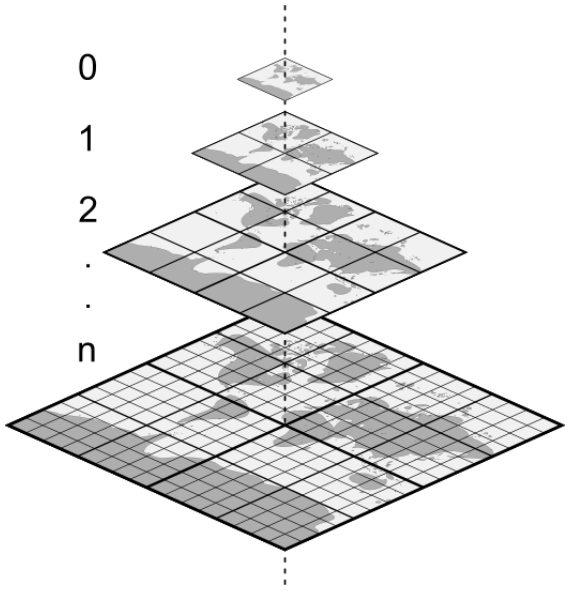


Figure 1.7: Pyramid structure of vector tiles with zoom-levels [33].

1.6 Applications of GIS

People working in various fields use GIS technology. Some examples are illustrated below:

1.6.1 Health Sector

- Epidemics like cholera, typhoid, and bilharzia are water-borne diseases that affect a lot of the population. GIS can be used to investigate the presence of such epidemics [34]. For example, in the COVID-19 pandemic, GIS provided straightforward and understandable visualization, real-time tracking of confirmed and reported case numbers, contact tracing, spread direction, and also to identify the hotspots to limit the dispersion and community spread [35].

- During the Ebola outbreak in 2013, government officials used GIS to site emergency treatment centres, manage bed capacity, and coordinate response efforts [36].

1.6.2 Disaster Management

GIS helps the disaster management sector to identify the areas which are disaster prone, to plan the rescue and evacuation of the affected people in any area. Thereafter, they also take the safety measures in order to prevent the highly effected area and people [37].

1.6.3 Climate Change

Meteorologists, climatologists, and GIS practitioners are increasingly integrating weather and climate data into their GIS workflows, combining them with other data to advance atmospheric science research; analyse the impact to populations, infrastructure, and ecosystems; and build myriads of applications enabling adaptation to climate change [38].

1.6.4 Oil and Gas Industry

- Pipeline monitoring: GIS can be used to continuously monitor pipelines to check for leaks and geo-hazards, and to manage and track inspections using remotely acquired data [39].

- Emergency response: GIS is important in response to emergencies such as oil spills and gas explosions, both in mitigation planning and response management. Data loaded into a GIS can be made available to all stakeholders regardless of their physical location, and may even be publicly available. This leads to better decision making during emergency situations [39].

- Land management: GIS stores land information as attributes such as lessor names, lease expiry dates, gross/net acreages. Centralising all land management data in an enterprise GIS environment helps enormously when generating the reports that are requirement by regulators [39]. Our project falls under this category.

1.7 Conclusion

A Geographic Information System, abbreviated GIS, is a computer system that creates, stores, processes and displays geographically referenced information. The first operational GIS appeared in the early 1960s, however the techniques automated by GIS date back to the 1830s. Web GIS is a form of GIS that uses web technology to communicate between a server and a client over a network, typically the Internet. GISs are used in different fields such as Health, disaster management and climate change. This study is concerned with the oil and gas sector, where we aim to design and implement a web-based GIS to manage data related hydrocarbon exploration acreages.

Chapter 2: System Design

2.1 Introduction

System design is a significant phase in building any type of software as it provides a deeper understanding of the system and its functionality. It helps in defining the requirements of the system, the user profiles and roles, the use cases, and the workflow. This helps in implementing a system conforming to the user's requirements.

2.2 Objective and System requirements

The objective of this project is to design and implement a web-based geographical information system for exploration acreages management which should replace the reports that are prepared monthly for managers. This computerization is therefore necessary and must include the following requirements:

- Authentication and profile management: employees working in the Planning, Asset, Data Management and Finance directorates of the Exploration Division, are involved in entering acreages data, whereas managers can consult all the data entered by these employees. Hence, five types of actors (profiles) are involved in the system. Each actor should be associated with access rights to the application, having a personalized menu.
- Data entry and validation, with the possibility to delete and edit all types of entered data except the deletion of an acreage which contains detailed information. The acreage data is described under the next section, in the "Acreage report contents" subsection.
- Visualization of acreages on the geographic map of Algeria and the possibility to consult the up-to-date acreage data from the map.
- The possibility to upload maps (images) and documents (pdf files of contracts, commitments, amendments).
- The possibility to visualize general statistics using digital dashboards.

- The possibility to generate pdf documents (report) from the previously entered data.

2.3 Specifications of the management of exploration acreages

Exploration acreages (in French *"périmètre d'exploration"*) are offered by **Alnaft**; which is the national agency for the valorization of Algerian hydrocarbon resources, its mission in particular, is the promotion of investments in the research and exploration of Algerian hydrocarbons, and the issuance of prospection and exploitation authorizations. The calls for tenders are intended for national and international companies [40]. For SONATRACH to start either prospection or exploitation in a specific acreage, it should request the permit from Alnaft first. If Alnaft licences SONATRACH, a contract, containing the needed specifications, is signed by both parties.

There are two types of exploration contracts:

1. **Research and exploitation contract:** when the petroleum system is already identified.
2. **Prospection contract:** when the area has not been studied before.

- The "research and exploitation contract" is newly called "upstream concession". In the old law, the contract duration is 7 years: 3 years for the first phase, two years for the second phase and 2 years for the third phase. In the new law, the contract duration is at most 7 years, having a maximum of three phases. The starting and ending dates of each phase are specified in the contractual commitments as well as the total of realizations that should be done within each phase.
- To pass from a phase to another, SONATRACH has to request Alnaft for a passage agreement. It is necessary to render a percentage of the acreage surface, 30% at most at the end of each phase.
- If the period of the phase is over and there is still ongoing work, the company requests Alnaft for an extension period (amendments).

2.3.1 Acreage report contents

The acreage report contains all the details of the exploration life cycle of the concerned acreage. It is created by the data management directorate, and it contains the following sections:

1) **General information:**

   - **Geographical information:** Asset, basins, blocs (see Appendix B), infrastructures...
   - **Petroleum system**

2) **Contractual framework:**

- **Information about the contract:** start and end dates, date of entry into force, contract type...
- **Requests and Amendments:** requests are made for Alnaft's agreement to pass to the next phase. An amendment is a small change or improvement that is made to a document (in this case contract) or proposed new law [41], amendments are made for Alnaft to extend the contract period.
- **Contractual commitments:** contain the starting and ending dates of each phase as well as the total of realizations that should be done within each phase.

3) **Physical and financial realizations:**

- **G&G studies:** with initial exploration and authorization completed, the company can begin the exploration realizations which include the geological and geophysical studies (G&G). Substructures of the earth are studied to localize areas where accumulations of hydrocarbons might occur.
- **Seismic achievements:** in oil and gas exploration, seismic waves are sent deep into the Earth and allowed to bounce back. Geophysicists record the waves to learn about oil and gas reservoirs located beneath Earth's surface [42]. There are two types of seismic achievements: 2D and 3D.
- **Drilling achievements:** after accomplishing all the geological and geophysical survey techniques, and if prospects are identified, the exploration company should first get exploration drilling authorization from Alnaft in order to test their theory that the selected prospects contain a commercial accumulation of hydrocarbons.
- Investment: the costs for each realization.

4) **Results of wells and volumes:**

- **Volumes in place:** volumes of oil and gas accumulations.

5) **2023 program and 2024/2028 Medium-term plan(MTP):** contains the program of the current year, and the medium-term plan for the next five years: Medium-term plans outline the company's direction for several years into the future (forecasts).

2.3.2 Staff role identification

This system consists of four main types of actors who work in the following directorates:

- Data Management: Employees working in the Data Management directorate create the report. They are responsible for initializing the geographical information, uploading maps, and adding acreages on the map.
- Planning Directorate: Employees working in the Planning Directorate are responsible for entering all information concerning the contract, commitments, amendments, and physical realizations including geological and geophysical achievements, seismic acquisition, well drilling.
- Asset Directorate: Employees working in the Planning Directorate Validate the physical realizations information entered by Planning directorate. They also enter volumes information.
- Finance Directorate: Feeds the financial achievements part under the Physical and financial realizations section.

  Finally, there are managers who can consult all the acreage data that is filled by the other actors.

2.4 Unified Modelling Language

The Unified Modelling Language, abbreviated UML, is a general-purpose graphical modelling language that is used to specify, visualize ,create and document the artefacts of a software system [43]. UML provides a set of diagrams that help in designing and communicating software systems. These diagrams fall under two major categories [43]:

- **Structural:** A system is modelled as a collection of objects (elements) that interact to perform work that ultimately benefits the system users. The structural classification defines the types of objects important to the system's implementation, as well as relationships among the objects. This category includes class, internal ,collaboration ,component , and use-case diagrams.
- **Dynamic:** describes the behaviour of a system and defines the history of objects *over time* and the communications among these objects to achieve specific goals. This category includes state machine, activity, sequence and communication diagrams.

For our design, we are going to use use-case, activity, sequence and class diagrams. For more details about constructing these diagrams see Appendix C.

2.5 Use case diagrams

The use case diagram captures the behaviour of a system as it appears to an outside user. It partitions the system functionality into meaningful transactions, known as use cases, and describes the interactions between these transactions and the system actors [43].

2.5.1 System actors and use cases

The table below shows each actor and their corresponding use cases. We choose to refer to the actors by the directorate names, for example ”Planning” actors refer to employees working in the planning directorate:

Actor	Use case(s)
Planning	-Authentication -Add a new acreage to the database -Initialize the contractual situation of an acreage -Add a new realization -Add a new request or amendment -Add 2023 program and 2024-2028 Medium-term plan (MTP)
Data Management	-Authentication -Add an initialized acreage to the map -Add acreage map files -Initialize acreage’s geographical information
Asset	-Authentication -Initialize the petroleum system -Validate realizations entered by Data Management -Enter Volumes
Finance	-Authentication -Enter the costs for each realization
Manager	-Authentication -Consult the data of an acreage from the map

Table 2.1: System actors and their use cases.

2.5.2 Planning's use case diagram

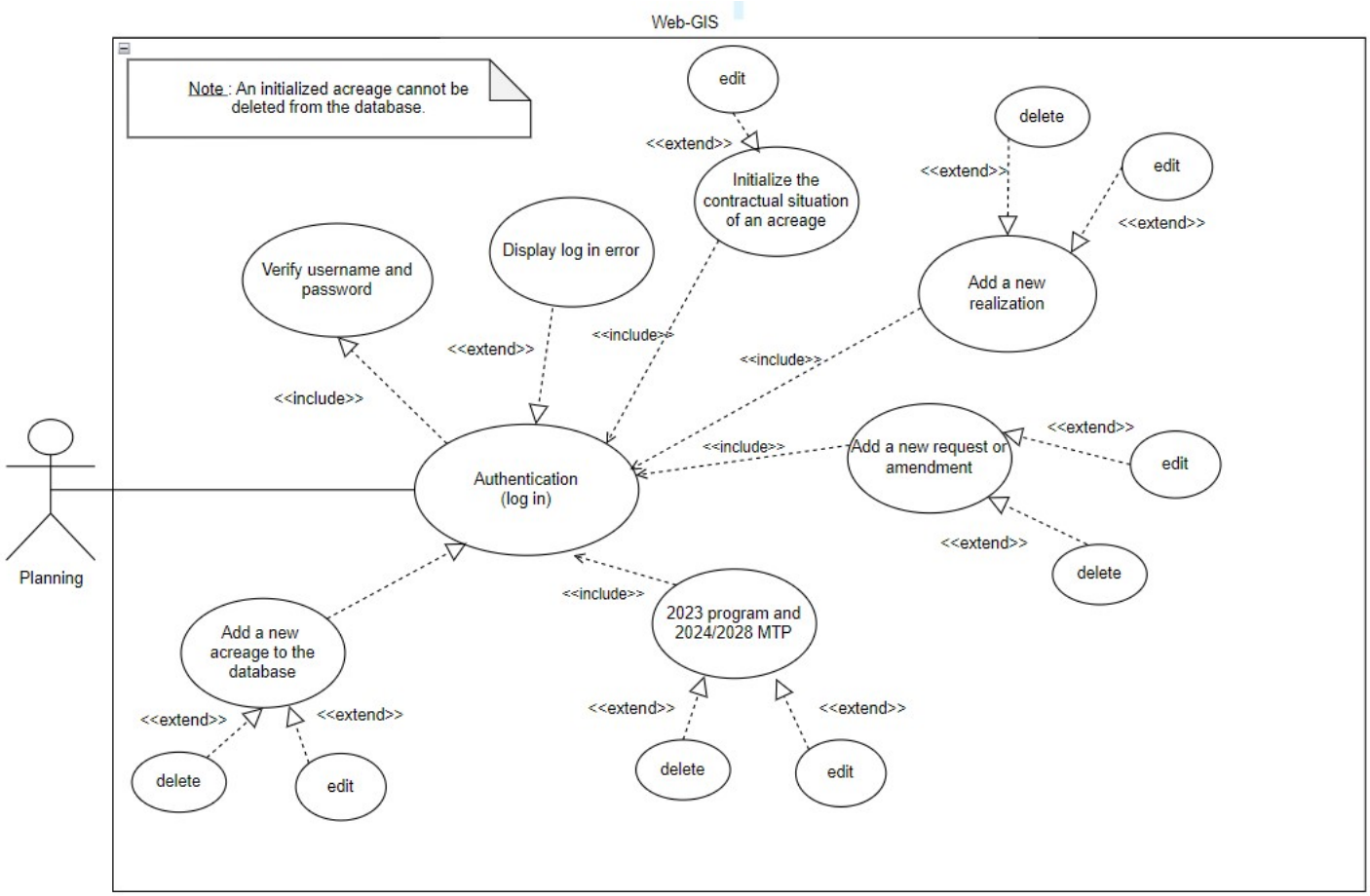


Figure 2.1: Planning's use case diagram.

Textual description of use cases:

Authentication: this use case is common for all actors.

Use case name	Authentication
Actor	Planning, Data Management, Asset, Finance, Manager
Objective	Ensure that the user is authorized to access the application.
Precondition	—
Scenario	1- The user launches the system. 2- The log in page requests the username and the password of the user. 3- The user enters his login credentials. 4- The system verifies the entered data by sending an authentication query to the database. 5- The user gets access to the home page with the appropriate navigation menu depending on his profile.
Alternative	If the log in credentials are not matched, the user gets redirected to the login page with an error message.

Table 2.2: Authentication use case textual description.

Add a new acreage to the database:

Use case name	Add a new acreage to the database
Actor	Planning
Objective	To add a new exploration acreage to the existing acreage listing.
Precondition	Authentication.
Scenario	1- The user accesses the ”Add a new acreage” page. 2- The user enters the required information for the new acreage: the reference, the name, the status, the asset, the basin. 3- The system validates the entered data and inserts the new acreage data to the database.
Alternative	If an acreage with the same reference or/and name already exists, the user gets redirected back to the ”Add a new acreage” page, with an error message that notifies him that the entered data already exists.

Table 2.3: ”Add a new acreage to the database” use case textual description.

Initialize the contractual situation:

Use case name	Initialize the contractual situation
Actor	Planning
Objective	To initialize the general contractual information and the contractual commitments of a specific acreage.
Precondition	Authentication
Scenario	1- The user accesses the ”Initialize acreage” page. 2- A list of acreages with two action buttons gets displayed, one button is for the general contractual information and another for the commitments. 3- The user clicks the desired button and a form gets displayed where the user enters the required contractual information for the selected acreage (general contractual information or contractual commitments). 4- The user clicks the ”Add” button to save the entered information.
Alternative	–

Table 2.4: ”Initialize the contractual situation” use case textual description.

Add a new realization:

Use case name	Add a new realization
Actor	Planning
Objective	To add a new realization in a specific acreage.
Precondition	Authentication
Scenario	1- The user first selects the realization type from the navigation drop-down menu. 2- The user accesses the ”Add a new realization” page. 3- The user selects/searches an acreage among the acreages list. 4- A list of all existing realizations gets displayed which the user can edit or delete. The user clicks the ”Add” button to add a new realization. 5- A form gets displayed where the user enters the required information for the concerned acreage. 6- The user clicks the ”Add” button to save the entered information to the database.
Alternative	–

Table 2.5: ”Add a new realization” use case textual description.

Add a new request or amendment:

Use case name	Add a new request or amendment
Actor	Planning
Objective	To add a new request for Alnaft to pass to the next phase, or to add a new amendment to, for example, extend the exploration end date depending on some factors.
Precondition	Authentication
Scenario	1- The user accesses the ”Add a new request or amendment” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A form gets displayed where the user can enter the required information. 4- The user should first select whether he wants to add an amendment or a request. 5- The form content automatically changes depending on the user’s choice. The user can also add a pdf document of the request/amendment, the document gets named automatically based on the acreage’s reference and document’s type. 6- The user clicks the ”Add” button to save the entered information.
Alternative	–

Table 2.6: ”Add a new request or amendment” use case textual description.

Add 2023 program and 2024-2028 Medium-term plan (MTP):

Use case name	Add 2023 program and the MTP of the next five years.
Actor	Planning
Objective	To add the program of the current year and MTP of the next five years.
Precondition	Authentication
Scenario	1- The user navigates to the ”program and MTP” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A form gets displayed where the user can enter the associated information. 4- The user clicks the ”Add” button to save the entered information to the database.
Alternative	–

Table 2.7: ”Add 2023 program and 2024-2028 MTP” use case textual description.

2.5.3 Data Management use case diagram.

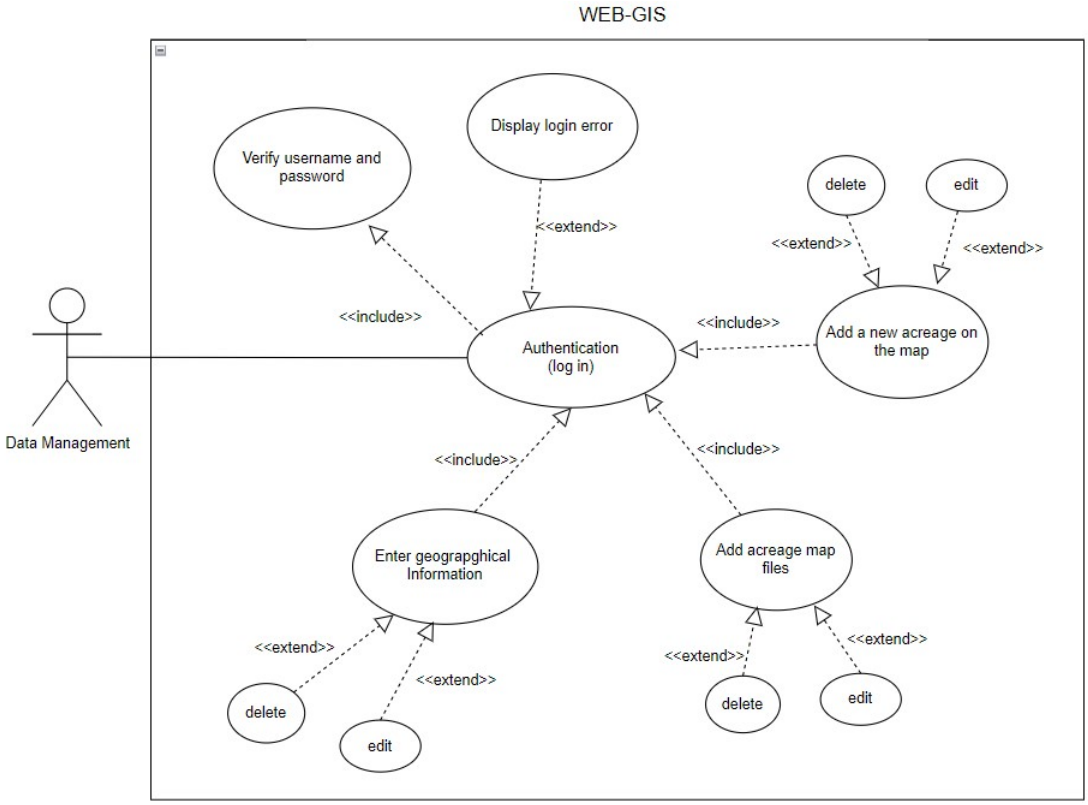


Figure 2.2: Data Management use case diagram

Textual description of use cases:

Add an acreage to the map:

Use case name	Add an acreage to the map
Actor	Data Management
Objective	To add a new initialized acreage to the map so that its data (report data) can be consulted after clicking on it.
Precondition	Authentication
Scenario	1- The user accesses the ”Add an acreage to the map” page. 2- The user selects/searches an acreage among the acreages that are not present on the map. 3- A form gets displayed where the user can enter the required information: the acreage coordinates, the colour. 4- The user clicks the ”Add” button to save the acreage data and add the acreage to the map.
Alternative	–.

Table 2.8: ”Add an acreage to the map” use case textual description.

Add acreage map files:

Use case name	Add acreage map files
Actor	Data Management
Objective	To add map image files that contain details of a specific acreage.
Precondition	Authentication.
Scenario	1- The user accesses the ”Add acreage map files” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- The user then selects the type of the map and uploads an image file of the map. 4- The file gets saved in the appropriate path and named automatically depending on the reference of the acreage and the selected map type.
Alternative	–

Table 2.9: ”Add acreage map files” use case textual description.

Initialize the geographical information:

Use case name	Initialize the geographical information
Actor	Data Management
Objective	To initialize the geographical information of a specific acreage.
Precondition	Authentication, the concerned acreage must already be initialized.
Scenario	1- The user accesses the ”Geographical information” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A form gets displayed where the user can insert the geographical information. 4- The user clicks the ”Add” button, the request gets sent to the server and the data gets saved in the database.
Alternative	–

Table 2.10: ”Initialize the geographical information” use case textual description.

2.5.4 Asset’s use case diagram

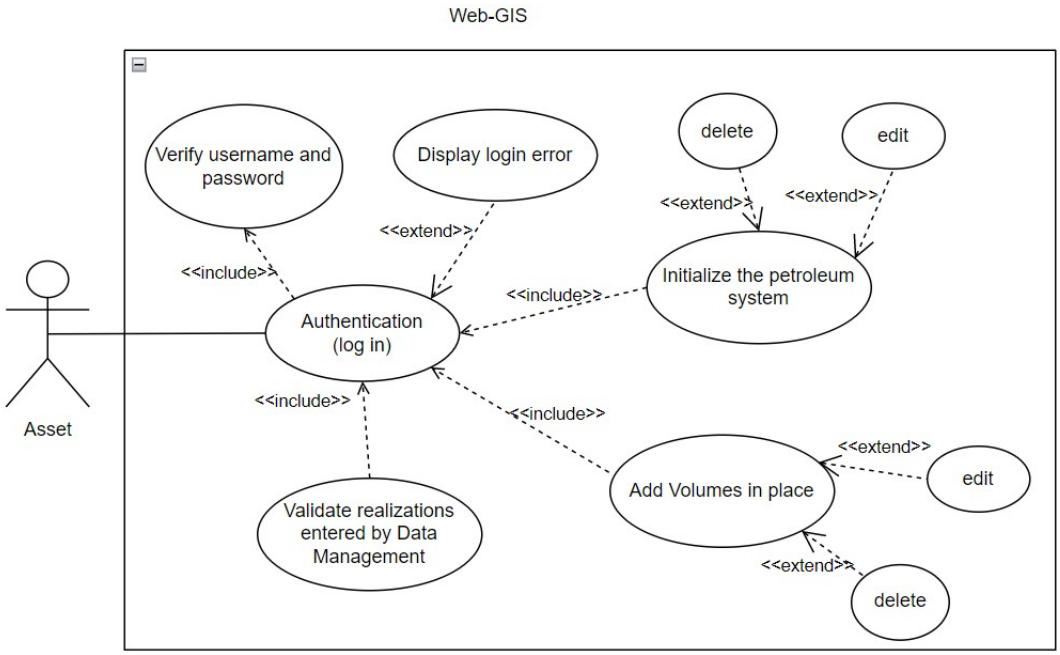


Figure 2.3: Asset’s use case diagram.

Textual description of use cases:

Initialize the petroleum system:

Use case name	Initialize the petroleum system.
Actor	Asset
Objective	To initialize the petroleum system of a specific acreage.
Precondition	Authentication
Scenario	1- The user accesses the ”Petroleum system” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A form gets displayed where the user can insert the petroleum system information. 4- The user clicks the ”Add” button, the request gets sent to the server and the data gets saved in the database.
Alternative	–

Table 2.11: ”Initialize the petroleum system” use case textual description.

Add volumes in place:

Use case name	Add volumes in place.
Actor	Asset
Objective	To add new volumes(oil & gas) to the associated acreage.
Precondition	Authentication
Scenario	1- The user accesses the ”Add volumes” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A list of existing volumes gets displayed. The user clicks the ”Add” button to add a new volume. 4- The user fills the data in the form that gets displayed. 5- The user clicks the ”Add button” to save the information into the database.
Alternative	–

Table 2.12: ”Add volumes in place” use case textual description.

Validate realizations:

Use case name	Validate realizations
Actor	Asset
Objective	To validate the realizations entered by Planning to ensure the accuracy and quality of data.
Precondition	Authentication
Scenario	1- The user accesses the ”Validate realizations” page. 2- The user selects/searches an acreage among the initialized acreages list. 3- A list of *unvalidated* realizations gets displayed. 4- The user selects a realization. 5- A form filled with data entered by Planning gets displayed. 6- If the data is accurate the user clicks the ”Validate” button, the request gets sent to the server and the data gets validated in the database. Else the user makes changes where necessary, the request gets sent to the server and the data gets updated and validated in the database. Only validated data is taken into consideration.
Alternative	–

Table 2.13: ”Validate realizations” use case textual description.

2.5.5 Finance’s use case diagram

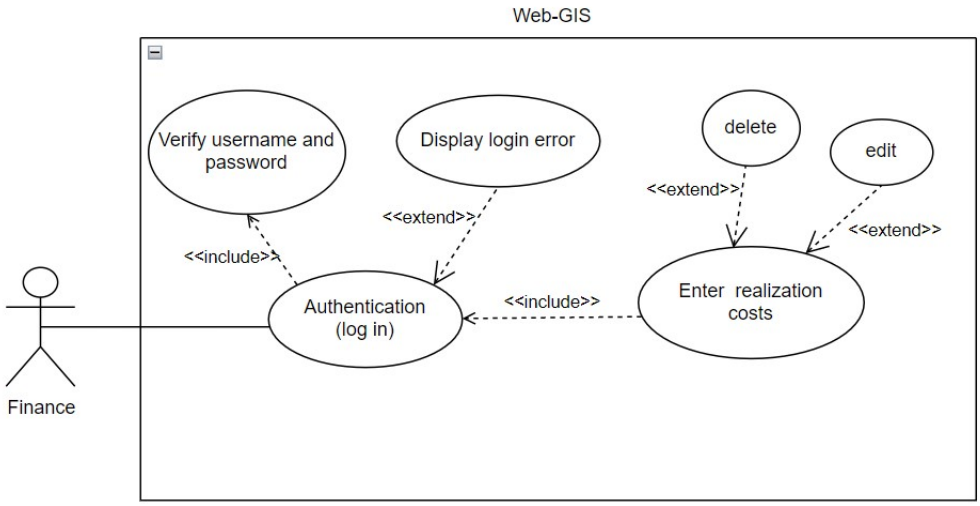


Figure 2.4: Finance’s use case diagram.

Textual description of use cases:

Enter the costs of a realization:

Use case name	Enter the costs of a realization.
Actor	Finance
Objective	To enter the costs spent on each realization.
Precondition	Authentication
Scenario	1- The user first selects the realization type from the navigation drop-down menu. 2-The user gets directed to the ”Enter costs” page corresponding to the selected realization type. 3- The user selects/searches an acreage among the initialized acreages list. 4- A list of all realizations (of the selected type) of the concerned acreage gets displayed, the user clicks the ”Add costs” button to add costs of the desired realization. 5- A form gets displayed where the user can insert the costs. 6- The user clicks the ”Add” button, the request gets sent to the server and the data gets saved in the database.
Alternative	–

Table 2.14: ”Enter the costs of a realization” use case textual description.

2.5.6 Manager's use case diagram

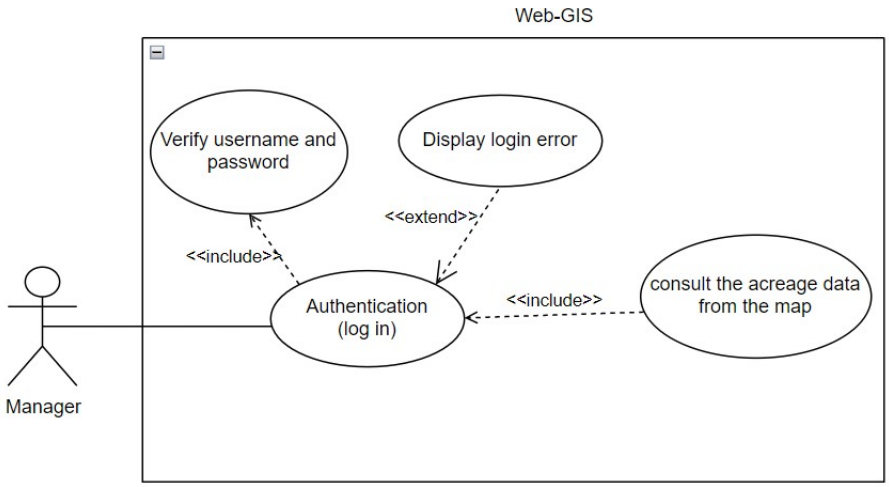


Figure 2.5: Manager's use case diagram.

Textual description of use cases:

2.5.6.1 Consult the overall data of an acreage from the map:

Use case name	Consult the up to-date data of an acreage from the map.
Actor	Manager
Objective	To consult all the map acreage information that has been inserted by other actors, when it is clicked.
Precondition	Authentication
Scenario	1- The user accesses the "Home" page. 2- The user clicks on the desired acreage on the map. 3- The user gets directed to a new window that displays the acreage information. 4- The user can consult the data online as well as view and print the pdf version of the acreage data (report).
Alternative	–

Table 2.15: "Consult the up to-date data of an acreage from the map" use case textual description.

2.5.7 Global use case diagram

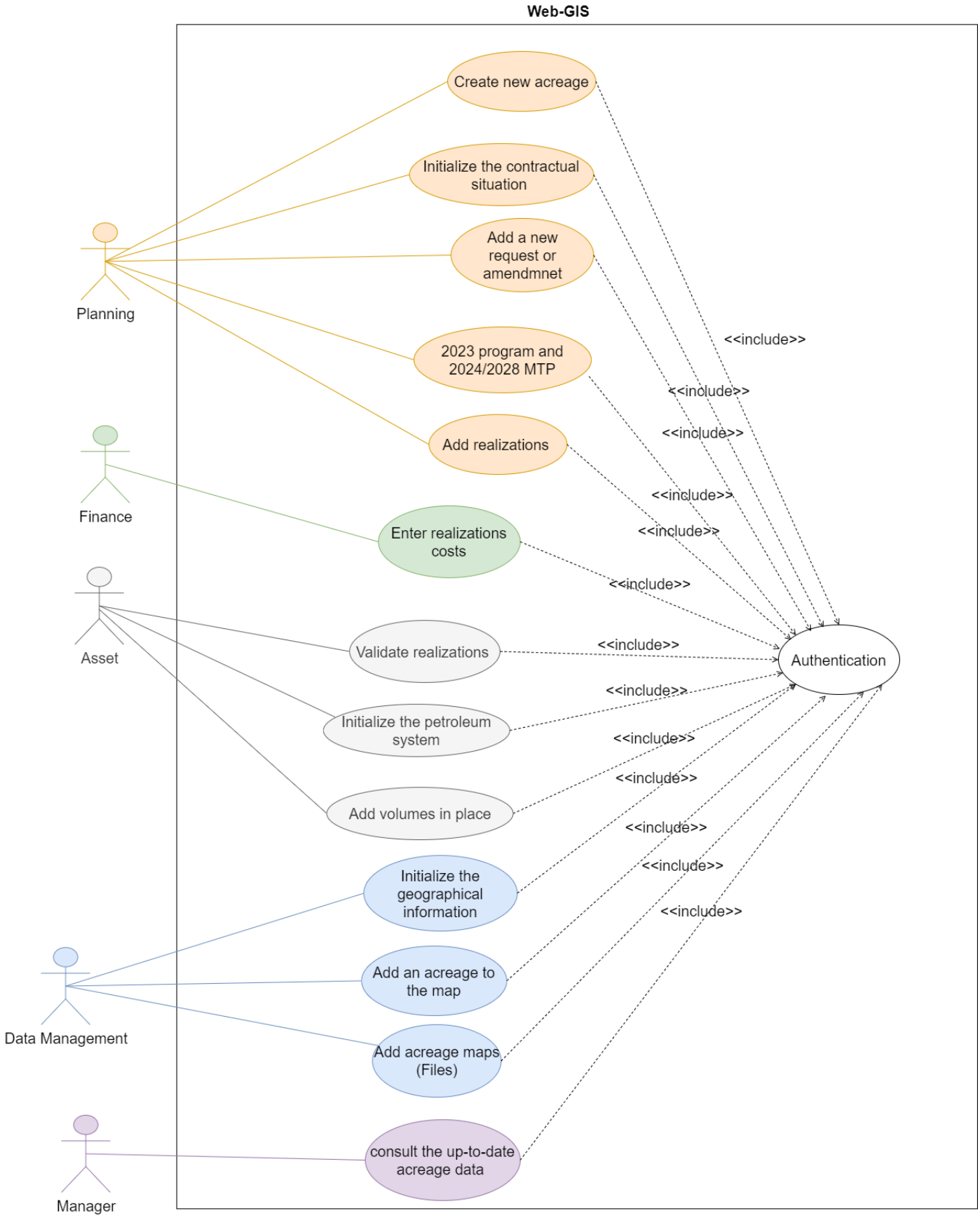


Figure 2.6: Global use case diagram.

2.6 Activity diagram

Use case diagrams are best sweeted for modelling the user interactions within a system, however they do not model the flow of these interactions. In contrast, an activity diagram is a version of flow chart that models the flow from one activity to another activity(use case) [44].

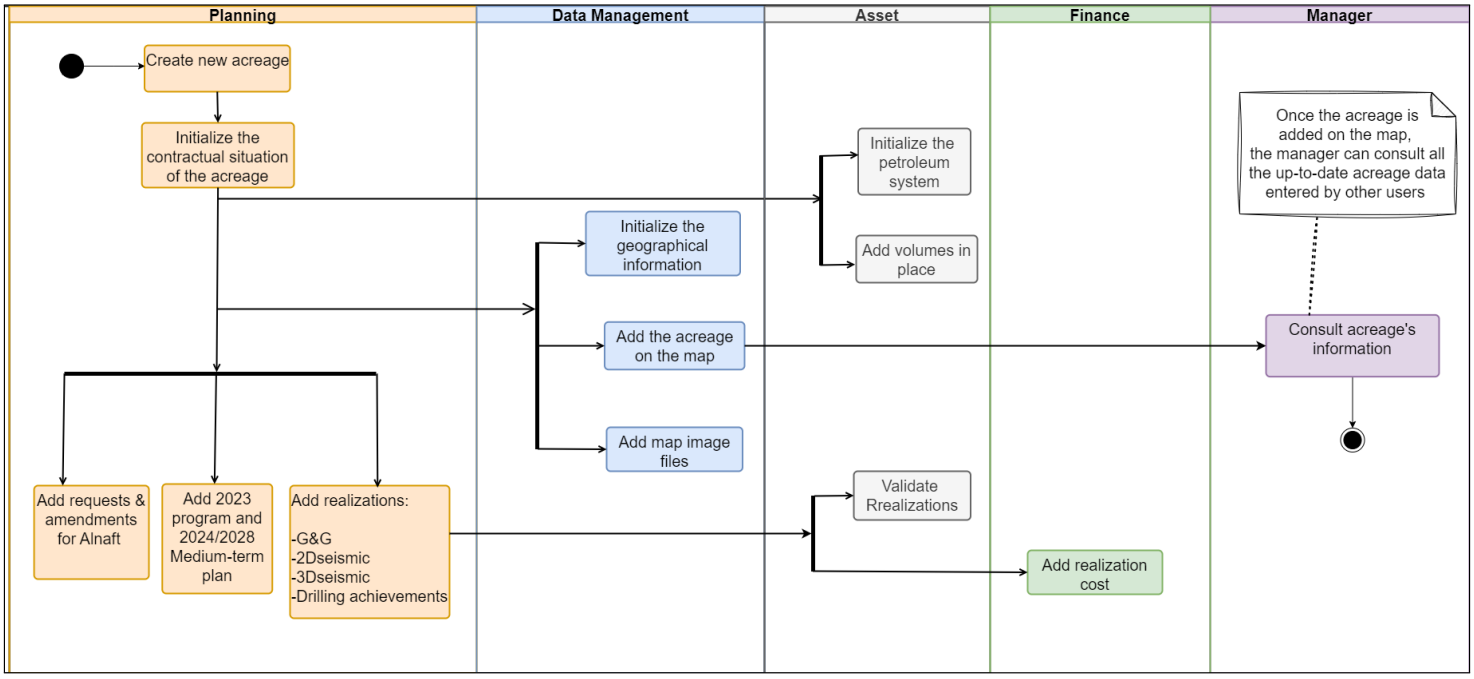


Figure 2.7: Activity diagram of the system.

Typically, actors should be authenticated so that they can perform actions in the system. Each actor is assigned a profile and a set of actions based on his job title. As illustrated in Figure 1.7, actors having profile ”Planning” can add a new acreage to the system’s database. The ”Planning” actor can also initialize the contractual situation of an existing acreage, including the contractual commitments. After that the contractual situation of a specific acreage is initialized, ”Planning” and other actors can add different types of information: ”Planning” can add requests, realizations, programs, ”Data management” can initialize the geographical information, add map images and they can also add the acreage on the map; once the acreage is added on the map ”Manger” can consult the up-to-date acreage data entered by other users. The ”Asset” actor can initialize the petroleum system, and add volumes in place(oil, gas). The ”Asset” actor can also validate realizations entered by the ”Planning” actor. The ”Finance” actor should enter the costs for each realization entered by the ”Planning” actor.

2.7 Sequence diagrams

A sequence diagram illustrates the objects involved in a use case and the the explicit sequence of messages that pass between them over time for one use case. Because sequence diagrams emphasize the time-based ordering of the activity that takes place among a set of objects, they are very useful for understanding real-time specifications and complex use cases [45].

2.7.1 Sequence diagram 01: Authentication

After the registered user launches the application, he gets directed to the Authentication page where he enters his login credentials. If the credentials are correct, the user gets directed to the Home page, else he receives an error message.

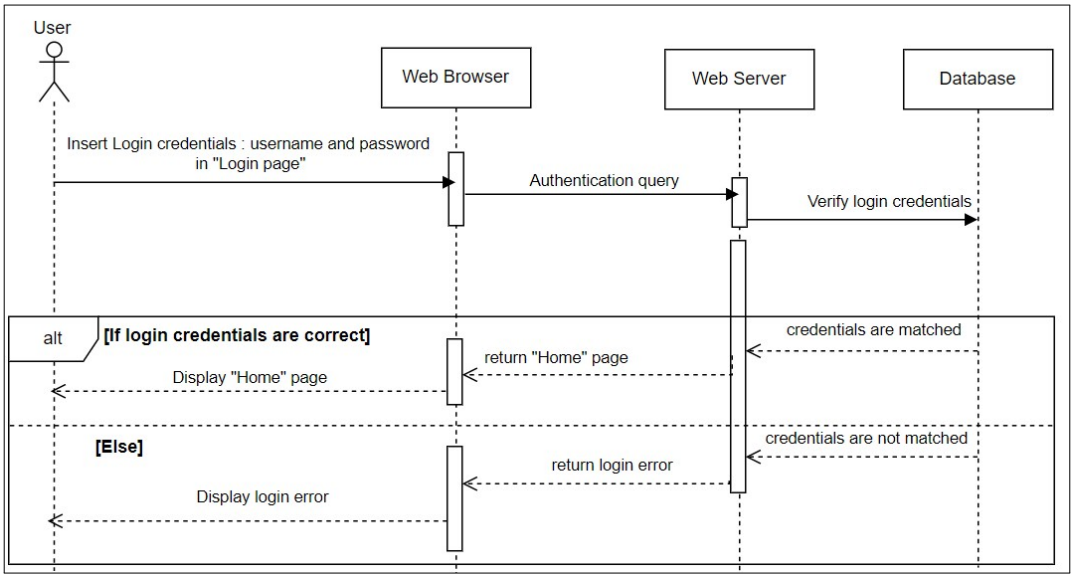


Figure 2.8: Authentication's sequence diagram.

2.7.2 Sequence diagram 02: Adding a new acreage to the database

The user with profile "Panning" accesses the "Add acreage" page that displays a form where he inters the information of the new acreage. If the entered information is valid, it gets successfully inserted into the database. Else, if the information is not valid, the user gets redirected to the "Add acreage" page with an error message.

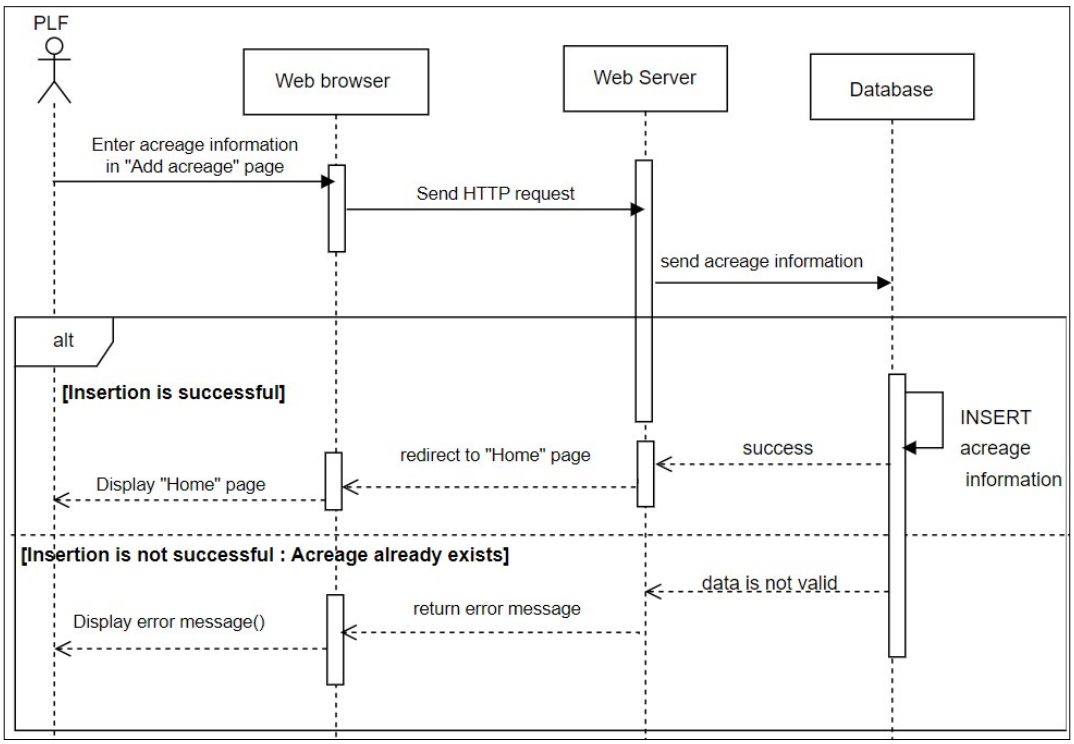


Figure 2.9: ”Add a new acreage” sequence diagram.

2.7.3 Sequence diagram 03: Add a new acreage to the map

The user with profile ”Data Management”, navigates to the page ”Add a new acreage to the map”, where a list of off-map acreages is displayed. The user selects an acreage and then a form gets displayed where the user enters the associated acreage data. The data gets sent to the server where it gets inserted to the JSON file that stores the geodata. The user then gets directed to the Home page that shows the updated map.

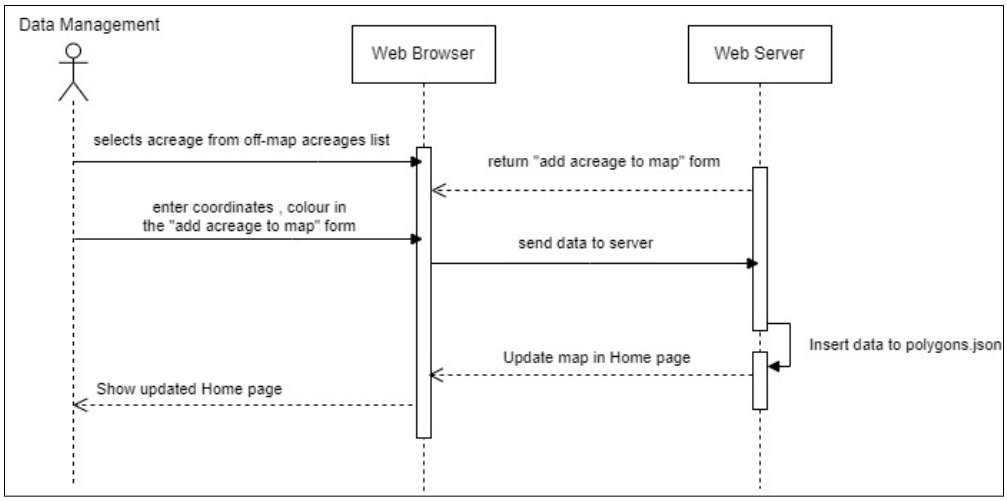


Figure 2.10: ”Add a new acreage to the map” sequence diagram.

2.7.4 Sequence diagram 04: Validate realizations

The user with profile ”Asset” selects a realization type from the navigation drop down menu. A listing of initialized acreages gets displayed. The user selects an acreage and a listing of all the corresponding realizations gets displayed. The user selects a realization and a form populated with the data entered by Planning gets displayed. The user makes changes where necessary and clicks the ”validate” button. The data then gets updated in the database.

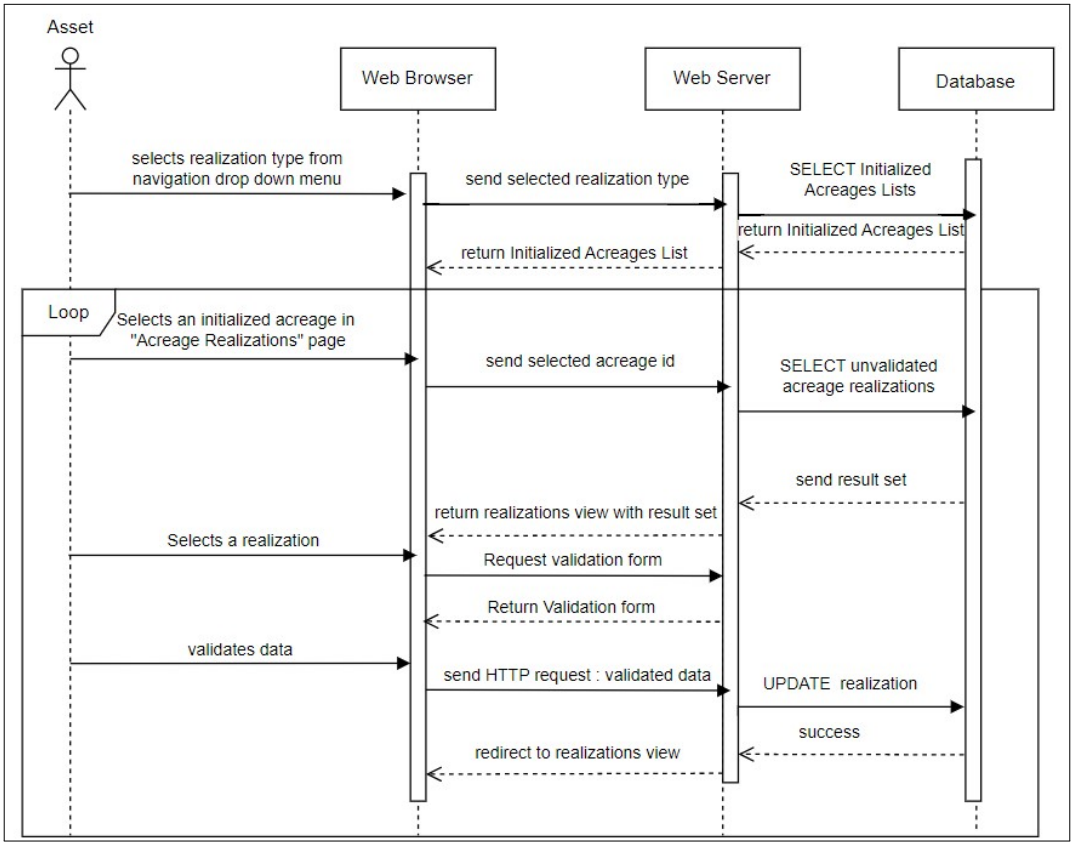


Figure 2.11: ”Validate” realizations sequence diagram.

2.7.5 Sequence diagram 05: Consult the acreage data from the map

The Manager accesses the Home page and clicks on a map acreage. A new page opens containing all the associated data to the the clicked acreage.

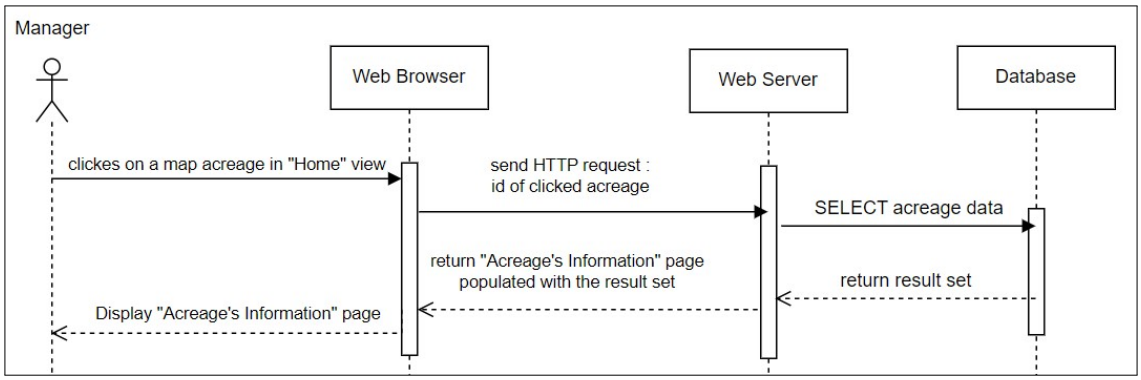


Figure 2.12: ”Consult the acreage data” sequence diagram.

2.8 Conceptual database design

The conceptual database design is a representation of the entire information content of the database, in an abstract manner in contrast to the way in which the data is physically stored.
The conceptual database design is based on the predefined specifications of the system. One of the commonly-used conceptual models is the entity-relationship model. An entity-relationship diagram (for more details about the types of relationships in a relational database, see Appendix D) models how the entities (the mathematical term for database tables) in a database relate to each other [29] [46].

For our database design, we are going to use a relational database model(for more details about relational databases, see Appendix D) which consists of four tables:

- The parent table ”acreages”, which contains the primary key.
- The child table ”general_data” which holds data concerning the ”general information” and ”contractual framework” sections of the acreage report.
- The child table ”realizations” which stores all the data of different types of realizations.
- The child table ”programs” which stores data concerning the program of the current year and the midterm review of the next coming four years.

For the table ”acreages”, it has the following attributes : acreage_id , a_name, status, basin, asset, Initialized and onMap.

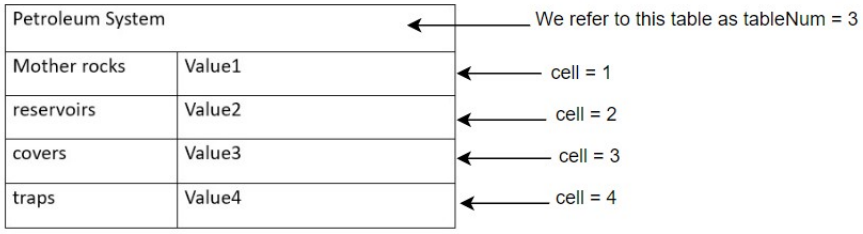
The ”Initialized” attribute is of type ”tinyint”. It is set by default to ’0’ meaning that the acreage is not initialized yet. It gets set to ’1’ when the acreage’s contractual situation is initialized but the contractual commitments have not been initialized yet. It gets set to ’2’ when both the contractual situation and the contractual commitments are initialized.

The ”onMap” attribute is also of type ”tinyint” and is set by default to ’0’. When the acreage is added on the map, its corresponding onMap attribute gets updated to ’1’. Both ”Initialized” and ”onMap” are going to be used to filter search results.

For the general_data table, it is going to merge several tables since the structure of the tables is unstable as the number of columns for each table is dynamic. Using a regular relational model we would have six tables: geographical information, petroleum system, general contract information, contractual commitments, requests and amendments. However, we choose to merge these tables in the following manner:

- acreage_id: holds the reference of the acreage.
- tableNum: represents the number of the table, as we are going to reference each table by a unique number.
- cell: represents the cell number of that table.
- cellValue: holds the value entered by the user for that specific cell .
- entered_by: holds the username of the user who inserted the data.

Hence, this table has a composite primary key of three columns : acreage_id, tableNum, cell. Taking for instance the ”petroleum system” table:



For the ”realizations” table, it is going to hold all types of realizations: G&G, 2D and 3D seismic achievements, and well achievements.

2.8.1 Entity relationship diagram of the system

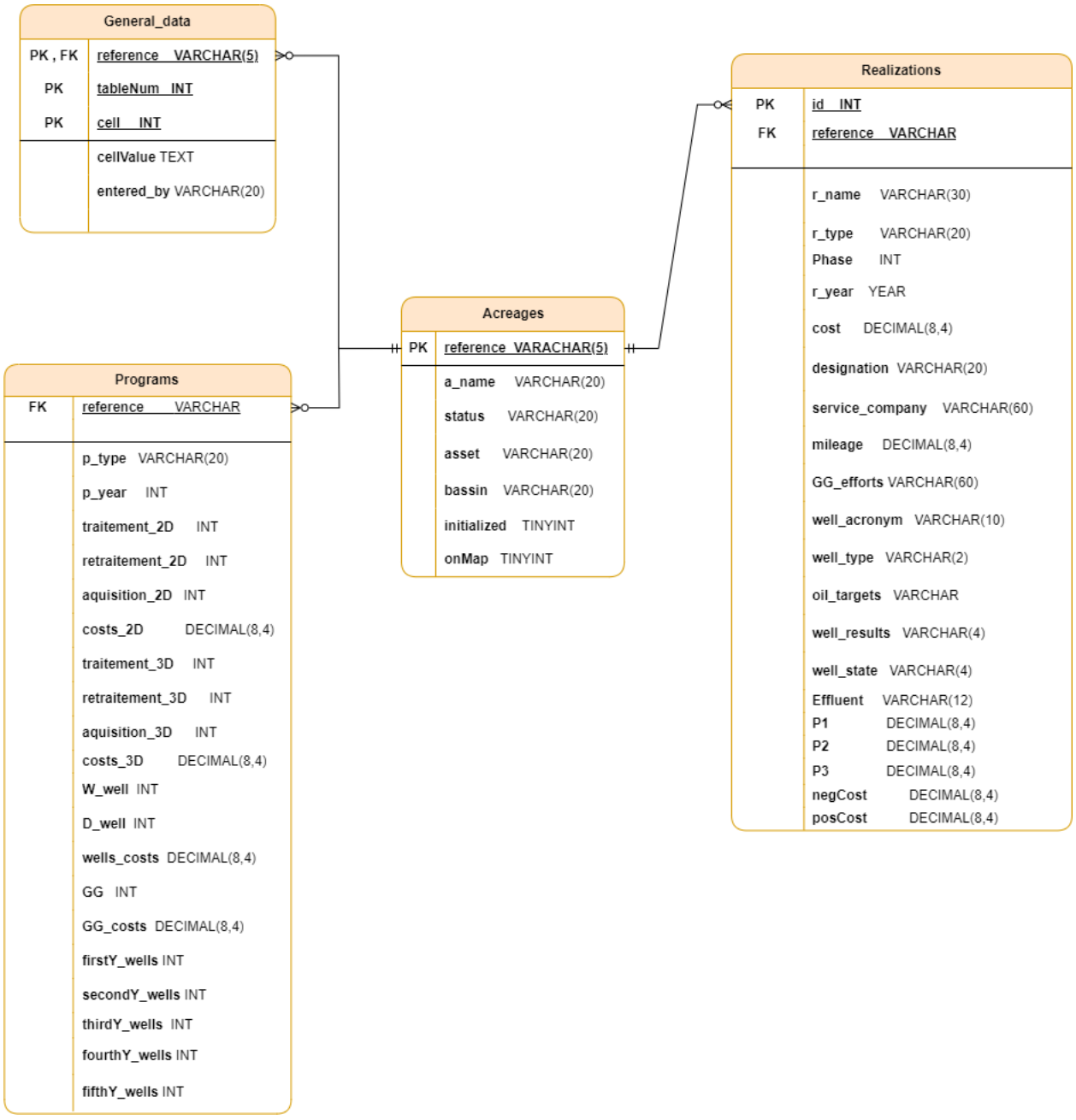


Figure 2.13: Entity relationship diagram of the system.

2.9 Class diagram

The class diagram is a structural UML diagram for building and visualizing object-oriented systems. It describes the structure of a system by showing: classes, their attributes, methods, and relationships between objects [47].

Since we are going to work with the MVC design pattern, our classes are going to be a collection of models and controllers. Basically, a controller is a class that handles the interaction between a view

(user interface) and a model class. A model class, in the other hand, is used to connect with the database. It defines a set of functions that execute SQL queries. The controller class defines a set of functions that, in response to some request or URL, call appropriate functions from the model class and return the corresponding view. For this design, each controller class is associated with a model class.

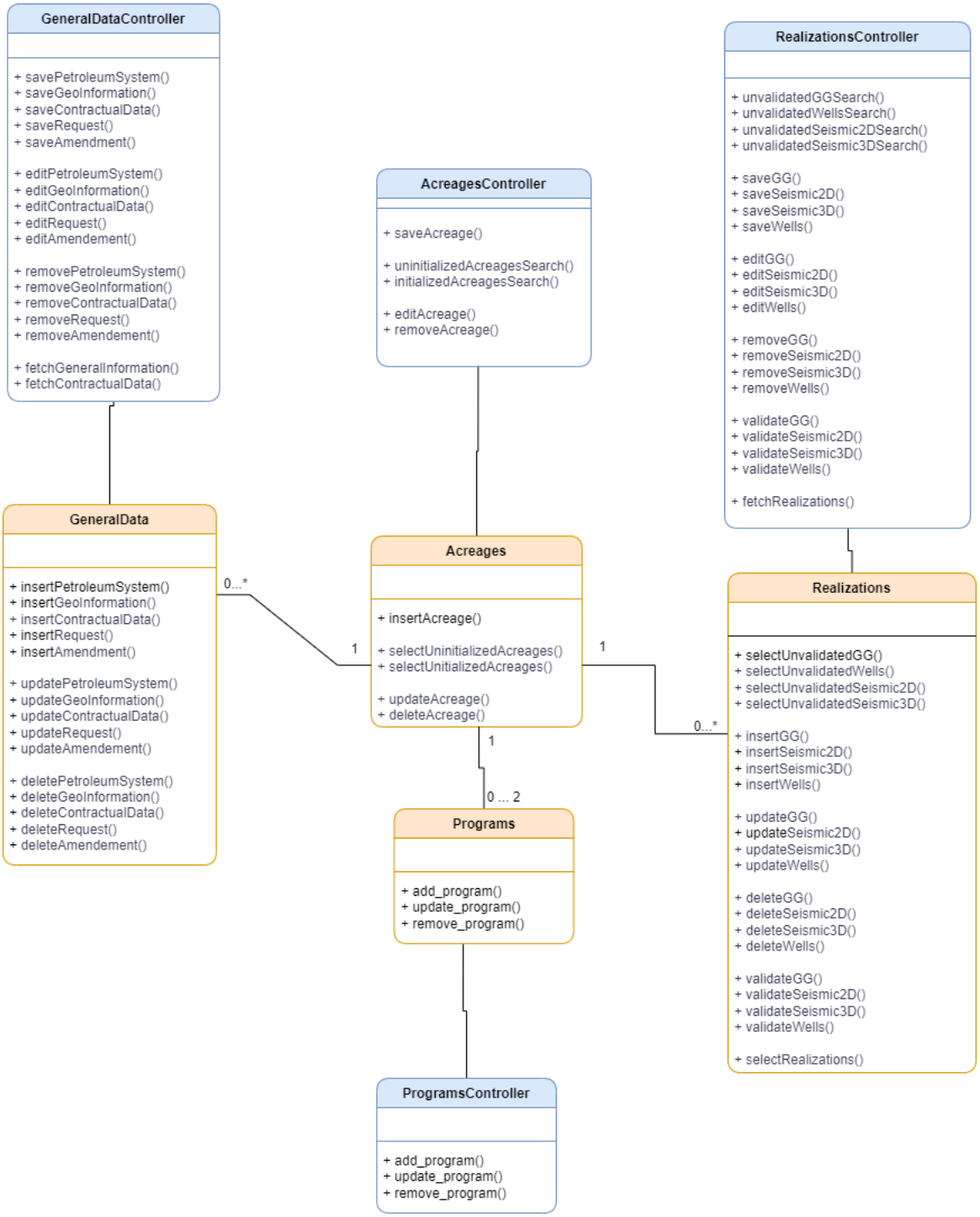


Figure 2.14: Class diagram of the system.

2.10 Conclusion

In this chapter, we have defined the requirements and specifications of the system. We have used different UML diagrams which provided a better understanding of the system at different levels. We used use case diagrams to visualize the use cases for each actor. The system basically consists of five actors: ”Planning”, ”Asset”, ”Data Management”, ”Finance” and also ”Manager” which can get access to all the attribute data through the basemap. Moreover, we have explained the relational model we have used and introduced the database schema, which consists of four main tables: acreages, general_data, realizations, programs. This theoretical basis, will guide us in the next stage of development which will be described in the next chapter.

Chapter 3: Implementation and Testing

3.1 Introduction

In this chapter we introduce all the tools used in implementing and developing the system and how these tools are inter-related. We also represent some user interfaces produced by testing the prototype in the local machine. Since our system is web-based, it permits the user to connect with a remote server via a web browser over a network. Hence, the technologies we are going to introduce fall under two categories: the client side category, also known as the front-end, and the server side category, also known as the back-end. These two types of technology communicate with each other via HTTP to form an integrated system.

3.2 Front-end Tools

The front-end is the code that is executed on the client side. This code (typically HTML, CSS, and JavaScript) runs in the user's browser and creates the user interface [48].

3.2.1 HTML

HTML stands for HyperText Markup Language. It is the authoring language used to create web page documents. It is a markup language, which means it is a system for identifying and describing the various components (HTML tags) of a document such as headings, paragraphs, lists, forms, tables and images. The markup indicates the document's underlying structure which can be thought of as detailed, machine-readable outline [49].

The HTML ***form*** is a very important element that is used to collect information from the user. Forms are generally made up of buttons, input fields, and drop-down menus. The other component of a web form is a script on the server side that stores or processes the information collected by the form and returns an appropriate response. The two commonly used attributes in the form element are *action* and *method*.

The action attribute provides the location (URL) of the application or script that will be used to process the form.

The method attribute specifies how the information should be sent to the server. There are only two methods for sending the encoded data to the server: POST or GET which is set by default.

3.2.2 CSS

While HTML is used to describe the content in a web page, Cascading Style Sheets (CSS) describe how that content should look. CSS is a style-sheet language that controls fonts, colours, background images, line spacing, page layout, and more. Special effects and basic animation can also be added to the page[49].

3.2.3 Bootstrap

Bootstrap is an open-source CSS framework that allows programmers to create responsive web pages with mobile-first considerations in mind [50].

A framework is a structure that software can be built on. It serves as a foundation, so that the programmer is not starting entirely from scratch. Frameworks are usually associated with a specific programming language and are suited to different types of tasks [51].

3.2.4 JavaScript

JavaScript is a client-side scripting language, which means it is reliant on the browser's capabilities and settings.

It is a *dynamic programming language*, which means it does not need to be run through any form of compiler. The browser effectively reads the code and interprets each line of code and runs it.

JavaScript is also *loosely typed* which means that type of the variables is not explicitly specified.

JavaScript is a way to add *interactivity* to a page. Whereas the *structural* layer of a page is HTML, and the *presentational* layer of a page is made up of CSS, the third *behavioural* layer is made up of JavaScript. All of the elements, attributes, and text on a web page can be accessed by scripts using the DOM (Document Object Model).

The **DOM** is a programming interface (API) for HTML and XML pages. It provides a structured map of the document and a set of methods for interfacing with the elements in the document. In effect, the markup is transformed into a format that JavaScript (and other languages) can understand [49].

- **JSON:**

  JSON stands for JavaScript Object Notation. It is a lightweight data format for transferring information from one program or system to another (typically between a server and web applications) [52].

  It defines only two data structures: objects and arrays. An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null [53].

- **jQuery :**

  jQuery is a fast, lightweight, and feature-rich JavaScript library. It makes HTML document traversal and manipulation, event handling, animation, Ajax and other things much simpler with an API that works across multiple browsers [54].

- **AJAX :**

  AJAX stands for Asynchronous JavaScript And XML. It is a web development technique that allows the web page to send and get data from the server in the background, and make updates to the page based on user interaction smoothly and *in real time , without the page needing to be reloaded*. Ajax may use XML (Extensible Markup Language) for data transfer, but it has become more common to use JSON [49].

3.2.5 Leaflet

Leaflet is a JavaScript library used to create simple and complex web-based maps, that can be interactive and editable [55].

Figure 3.1: Leaflet logo

Creating the basemap:

- The first step required to create a basemap, is to reference the JavaScript and CSS files of Leaflet. This can be done in one of two ways: either by referencing a hosted file or downloading a copy to the local machine and reference that copy.

- Secondly, an HTML tag is required to hold the map. The tag should be given an id so that it can be referenced by a map object in the script. The tag should also be assigned a width and a height using CSS.

- After creating the that holds the map, a **map object** can be created as shown in the line of code below:

```
1
2 var map = L.map('mapID').setView([28.0339, 1.6596] , 5);
```

Listing 3.1: Creating a map object in Leaflet

The variable "map" is a new instance of the Map class. In OOP, usually, classes are created using the "new" keyword, this also can be performed as follows [55]:

```
1
2 var map = new L.Map();
```

Listing 3.2: Creating a map object in Leaflet

however, this is not very commonly used since Leaflet implements factories that remove the need for the "new" keyword. This is achieved by complementing each class with a lowercase method called a factory method, for example [56]:

```
1
2 L.map = function (id, options) {
3     return new L.Map(id, options);
4 };
```

Listing 3.3: Creating a map object in Leaflet

L.map() has been given the id named mapID and two options: centre and zoom. The centre option positions the map on the screen with the latitude and longitude in the centre of the element, hence it takes the [latitude, longitude] parameters. The zoom option sets the default zoom level, the map can be zoomed in or out at the desired level. Zoom takes an integer parameter, the larger the number, the tighter the zoom. The minimum zoom level is 0 whereas the maximum zoom level is 18.

In listing 4.1 we have passed the coordinates of the centre of Algeria and a zoom level of 5 to the setView function .

Leaflet uses angular units for geographical coordinates, and **Spherical Mercator projection** (EPSG:3857), which is set in by default in Map's crs (Coordinate Reference System) option [57]. This projection is used by major online tile providers such as Google Maps due to its computational simplicity.

- **Adding a tile layer:** A tile layer is the basemap or the imagery that points, lines and polygons are added on top of. Tile layers are a service provided by a tile server. A tile server usually breaks up the layer into 256 x 256 pixel images called tiles. Needed tiles are retrieved based on latitude, longitude and zoom through a URL that requests /z/x/y.png. Only these tiles are loaded. As the user pans and zooms, new tiles get added to the map [55].

  In listing E.0.1 in the Appendix E, the tileLayer function is used to load and display tile layers on the map. A URL template is provided to OpenStreetMaps which is an open source tile layer provider.

- **Adding polygons on the map:** Leaflet allows the following geometric primitives of vector data that can be added to a map: points (markers), polylines, circles and polygons. For this project we are going to use polygons, as they are going to represent exploration acreages. A polygon is a polyline that is closed, hence it requires a minimum of 3 coordinates. In Leaflet, it is easy to add any geometry type to a map using the appropriate class. For polygons, we have to pass the coordinates to the Polygon class, with some styling if needed. An example of L.polygon is shown in listing E.0.2 in Appendix E.

  There is also L.geoJSON which supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, and GeometryCollection. Features in GeoJSON contain a Geometry object and additional properties for representing attribute data [58].

  The resulting map on the browser after adding listing E.0.2, Appendix E, is shown in the figure below:

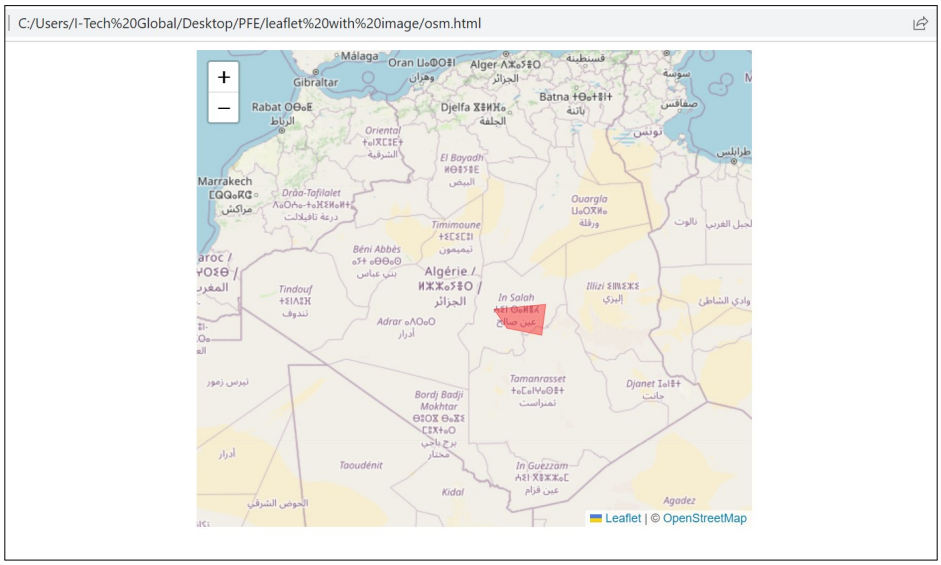


Figure 3.2: Map of Algeria from OSM with a polygon at zoom level 5.

As the page gets refreshed, the polygon is gone. Using AJAX together with JSON we were able to store the polygons and display them on the map permanently.

We created a JSON file, named polygons.json, that we used for storing spatial data of the polygons (coordinates), as well as the name and colour of each polygon. In Javascript, AJAX is used to fetch the content of the polygons.json file using a get request. The data is first passed to the L.geoJSON() function which parses GeoJSON data and display it on the map. The eachLayer() function is used to iterates over the polygons (layers) of the map, to attach a pop-up and a style to each polygon. The bindPopup method is used to show a pop-up containing the name of the acreage as the user hovers over it.

The geojson polygon objects are stored in the JSON file as illustrated by listingE.0.4 in Appendix E. There is a little problem however, the map tiles are stored in the OSM server, hence an Internet connection is required for retrieving the needed tiles. Since this web application is going to be hosted on an Intranet server, we want the map to be always displayed on the home page, even when the connection is lost. To achieve this, we replaced the map tiles by a single image using L.ImageOverlay which loads and displays a single image over specific bounds. In our case, the bounds represent the minimum and maximum coordinates of the map image of Algeria.

We used QGIS software to generate the map and get its exact bounds. We could easily do so by setting the CRS to EPSG4326, which is a common CRS that uses simple Equirectangular projection. This CRS uses degrees as coordinate units whereas EPSG:3857 uses meters. The code that implements this is shown in listingE.0.5 in the appendices. We had to change the map's CRS to EPSG4326. The resulting map is shown Figure 3.3:

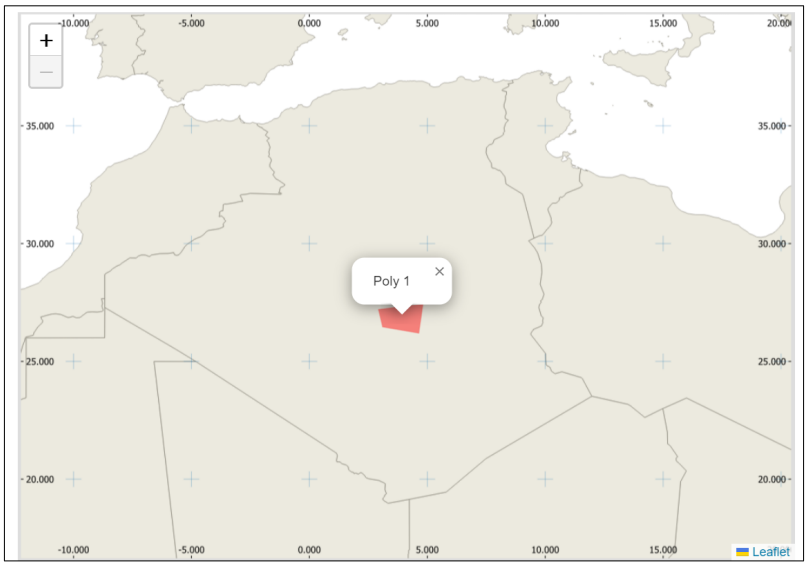


Figure 3.3: Offline map of Algeria.

3.2.6 Chart.js

Chart.js is an open source visualization JavaScript library. It provides ready-to-use interactive visualizations of data with minimal coding. After loading the data into a standard JavaScript array and adding other configurations using simple object-based declarative structures, Chart.js automatically scales the loaded data and generates sticks and grid lines, creates interactive tooltips, and fits the available space making the resulting chart automatically responsive. Chart.js comes with several built-in chart types such as line chart, pie chart, doughnut chart, and radar chart [59].

3.3 Back-end Tools

The back-end is all of the technology required to process the incoming HTTP requests, and generate and send the HTTP response to the client. This includes three major parts [48]:

1. The server: is the computer that receives requests.
2. The application: is the programs **running on the server** that listens for requests, retrieves information from the database, and sends a response.
3. The database: databases are used to organize and persist data.

3.3.1 Laragon

Laragon is a fast, lightweight, portable development environment for Windows. It allows users to create and test their programs on a local web server. It supports the following server-side languages: PHP, JavaScript (Node.js), Python, Java, Go, Ruby. Laragon integrates with PHP and Node.js by default [60][61].

Furthermore, Laragon ships with several web servers, database management tools, DNS services (for remote access to local applications), and a terminal emulator which makes it possible to run PHP/Node.js applications in terminal, without having to type the full path to the executable file.

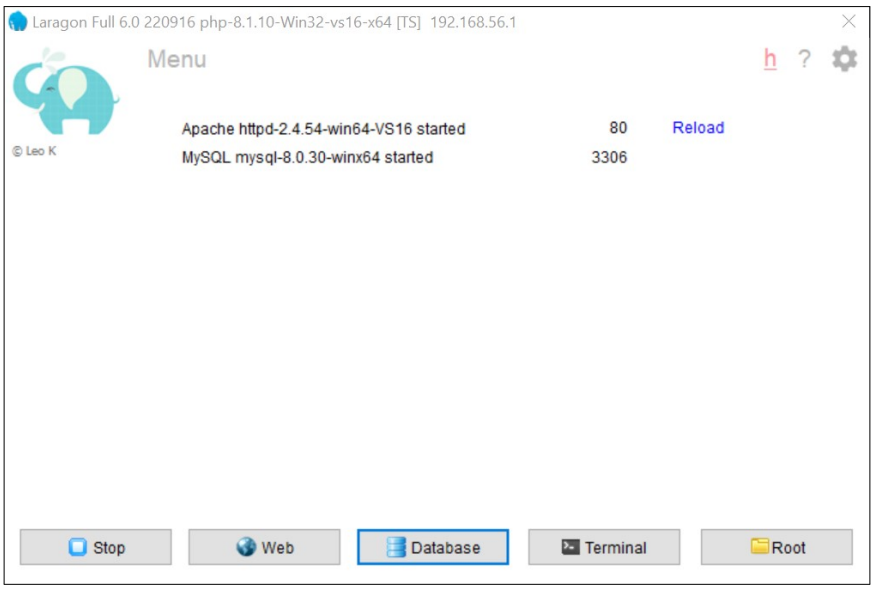


Figure 3.4: Laragon main UI, with quick toggles, server status, and a menu.

3.3.1.1 Apache HTTP Server

Laragon supports Apache 2 and Nginx web servers software.Apache, is an open-source Web server. As a Web server, Apache is responsible for accepting HTTP requests from Internet or intranet users and sending them their desired information in the form of files and Web pages. Much of the Web's software and code is designed to work along with Apache's features [62].

Figure 3.5: Apache Server logo.

3.3.1.2 MySQL

As already mentioned in Chapter1, a database management system is a computerized system or a computer program whose overall purpose is to store information and to allow users to retrieve and update that information on demand[29].

MySQL, MariaDB, PostgreSQL are the database management systems supported in Laragon. However, only **MySQL 8** is installed by default, and it is the DBMS used for this project. For MySQL, Laragon integrates **HeidiSQL** and phpMyAdmin.

3.3.1.3 HeidiSQL

HeidiSQL is a free and open-source database administration tool that runs under Windows. It supports the following DBMSs: MariaDB, MySQL, MS SQL, PostgreSQL, SQLite, Interbase and Firebird . It allows creating and editing tables, generating SQL-exports, writing queries with customizable syntax-highlighting and code-completion, browsing and editing table-data and more [63].

Figure 3.6: HeidiSQL logo.

3.3.2 Laravel

3.3.2.1 PHP

PHP is a recursive acronym for Hypertext Preprocessor. It is an open-source and cross-platform scripting language used to create dynamic and interactive HTML Web pages. A server processes PHP commands when a website visitor opens a page, then sends results to the visitor's browser.

PHP runs most efficiently on an Apache server. As an Apache module, PHP is very fast and lightweight, allowing for quick turnaround [64].

PHP also provides a set of useful libraries, among which is the **Dompdf** library. This library provides a simple way to convert HTML to PDF documents [65]. We have used this library to convert acreage data web pages to pdf files (a report).

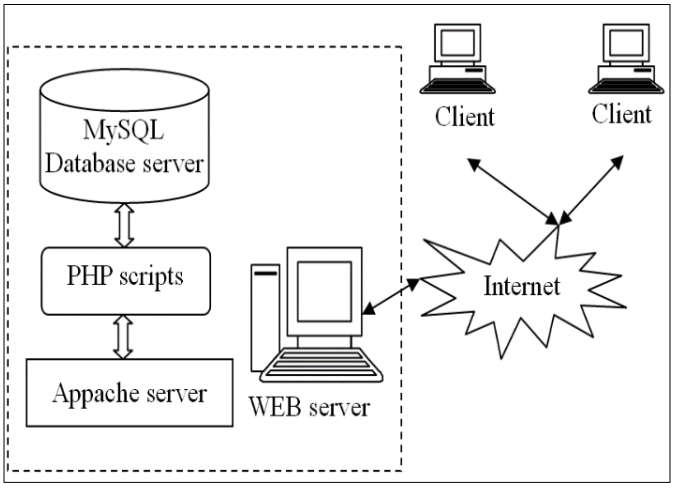


Figure 3.7: Interaction of Apache, MySQL and PHP in the server side with the clients[66].

3.3.2.2 What is Laravel?

Laravel is a PHP-based Web framework that is fully based on the MVC architecture [67].
Laravel utilizes **Composer** to manage its dependencies. Composer is a dependency management tool in PHP. For any PHP project, it is needed to use a library of codes. Composer easily manages that task helping to declare those codes. It also allows to install or update any code in the library [68].

For example it is possible to install Laravel by issuing the Composer create-project command in the terminal emulator :
composer create-project laravel/laravel directory projectName

3.3.2.3 MVC architecture

MVC is both a design and architecture pattern that divides an application into three major logical sections:

1. **Model:** each model class represents an individual database table. It can interact with the application's database(s), execute queries, retrieve, update, delete, and create data. It is also responsible for guaranteeing the evolution of the database structure from one stage to another by maintaining a set of **database migrations** [67]. Migrations are like version control for the database. They allow modifying and sharing the application's database schema [69].

   One of the ways that Laravel uses to interact with the database is the DB facade which runs raw SQL queries, and it is the one that we used for our implementation. A facade is a structural design pattern that provides a simplified interface to a complex set of classes. The machinery that makes this work is in the Facade class [70][71].

   A simple query using the DB facade is shown below:

```
$geo_info = DB::select("SELECT cellValue
                        from global__data
                        where perimetreID='".$ref."'
                        AND   tableNum = '1' ");
```

Figure 3.8: An SQL query using the select method of the DB facade.

2. **View:** represents the template that outputs the data to the end user. It is made up of HTML/CSS and possibly JavaScript. Laravel uses a powerful templating engine called **Blade**. Blade does not restrict using plain PHP code in the templates. Blade template files use the ".blade.php" file extension. It allows echoing the results of any PHP function using the Blade echo statement, for example :

   The current UNIX timestamp is {{ time() }}.

It also provides convenient shortcuts for common PHP control structures, such as conditional statements and loops. We, for example, can write if statements using the @if, @elseif, @else, and @endif directives [72].

3. **Controller:** The controller acts as a link between the model and the view. The end user interacts with the controller by sending an HTTP request using their browser. In order for the request to get to its appropriate controller it has to pass by the router first. **Routing** is an essential concept in MVC, that allows to route all the URLs and requests to their appropriate controller. The controller, in response to that request, may write data to and/or pull data from the model (database). The controller will then likely send data to a view, and then that view will be returned to the end user to display it in their browser [73]. Blade views may be returned from routes or controllers using the global **view** helper.

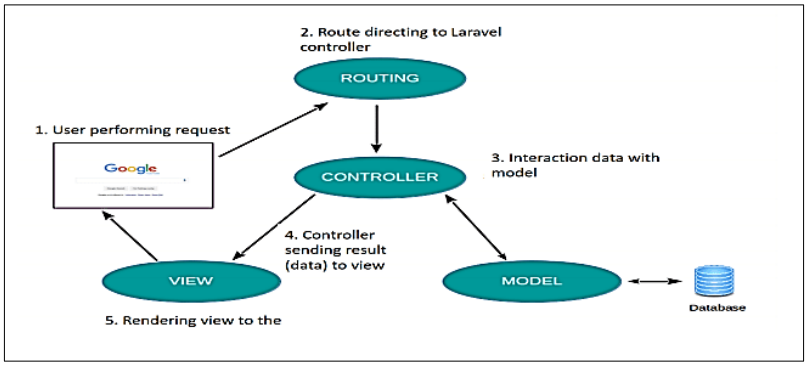


Figure 3.9: MVC design pattern [74].

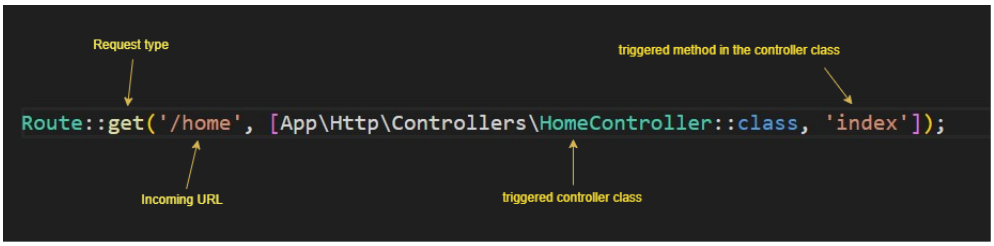


Figure 3.10: Example of simple route that triggers the index function in the HomeController.

3.3.2.4 Laravel's Directory Structure

The root directory contains the following folders by default:

1. **app:** this folder is extremely important. It has five significant subfolders:

   - **HTTP:** has a file named "kernel.php" and two subdirectories, one for **controllers** and another for **middlewares** .

     The kernel.php file defines the property *$middlewareAliases*. This property contains entries, such as auth, can, signed, that will be run before the request is executed. These entries configure error handling, configure logging, detect the application environment, and perform other tasks that need to be done before the request is actually handled. The HTTP kernel also defines a list of HTTP middlewares that all requests must pass through before being handled by the application [75].

     Generally speaking, **Middleware** is a software that enables one or more kinds of communication or connectivity between applications or application components in a distributed network [76].

     In Laravel, Middleware is the program responsible for providing a convenient mechanism for filtering HTTP requests before the actual logic process (the controller). Laravel includes the Authentication middleware by default. It verifies the user of the application is authenticated. If the user is not authenticated, the middleware will redirect the user to the application's login screen [77].

     For this project, we have multiple profiles: asset , planning , data management , finance and manager. Hence, we have added a new column to the users table called 'profile'.

     We have also added the *profile* entry to the middlewareAliases property, and created a new file under the middleware folder called "VerifyUserProfile.php" which contains the function handle() that handles incoming requests before passing to their appropriate controller.

```php
public function handle(Request $request, Closure $next , string $profile): Response
{
    if($request->user()->profile === $profile)

        return $next($request);

    abort(403 , 'Unauthorized');
}
```

Figure 3.11: The handle method that verifies the user's profile.

The abort function throws an HTTP exception which will be rendered by the exception handler. So, whenever a user tries to access a URL that is not authorized for his profile, he gets a

403 Forbidden Error. We have assigned middleware to specific routes, using the *middleware* method when defining the route. Like this each user profile is provided with access to specific routes and specific views.

```
Route::get('/Asset/SystemPetrolier', [App\Http\Controllers\PerimetreController::class, 'petroleumSystemSearch'])->middleware('profile:asset');
```

Figure 3.12: A route with middleware

- **Models:** contains the models classes.
- **Console:** contains all of the Artisan commands. Artisan is the name of the command-line interface included with Laravel [78].
- **Exceptions:** contains the application's exception handler.
- **Providers**

2. **bootstrap:** is needed for the startup of Laravel. It contains the app.php file which bootstraps the framework. It has also a cache directory which contains framework generated files for performance optimization.

3. **config:** contains all of the application's configuration files.

4. **database:** The database directory contains the database migrations, model factories, and seeds [79]. Database seeding is the process of filling the database tables with the test data programmatically [80].

5. **public:** The public directory contains the index.php file, which is the entry point for all requests entering the application. This directory also houses assets such as images, JavaScript, and CSS [79].

6. **resources:**The resources directory contains all the views.

7. **routes:** The routes directory contains all of the route definitions for the application. By default, several route files are included with Laravel: web.php, api.php, console.php, and channels.php.For most applications, routes are defined in the web.php file. These routes may be accessed by entering the defined route's URL in the browser[**?**].

8. **storage:** contains logs, compiled Blade templates, file based sessions, file caches, and other files generated by the framework.

9. **tests:** contains automated tests, for running portions of code and reporting any errors.

10. **vendor:** contains the Composer dependencies.

The root directory also contains some files. Among these files is the .env file. This file contains some common configuration values that may differ based on whether the application is running locally or on a production web server [81]. For example, it is in this file where we have set the database connection:

```
11  DB_CONNECTION=mysql #DBMS type
12  DB_HOST=127.0.0.1
13  DB_PORT=3306
14  DB_DATABASE=gis_db  #Database name
15  DB_USERNAME=root
16  DB_PASSWORD=
```

Figure 3.13: Configuring the database connection in .env.

3.4 System diagrams

3.4.1 General diagram of the system

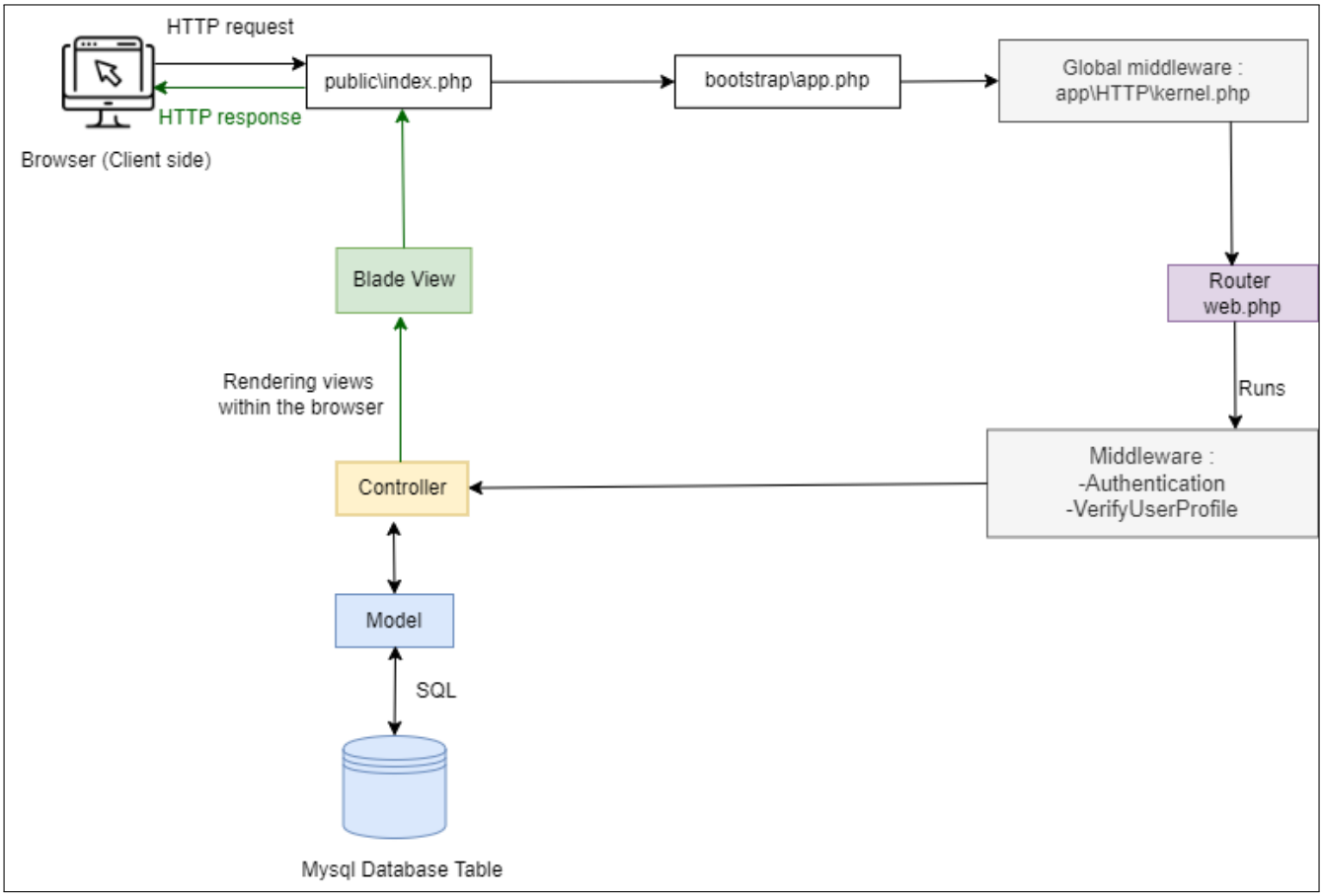


Figure 3.14: General diagram of the system.

As illustrated by Figure 3.14, the system user performs an HTTP request on the browser. The request gets sent to the index.php file which is the entry point for all requests entering the system. Then the request is forwarded to the app.php file, under the bootstrap folder, which bootstraps the framework. After that, a set of middleware entries inside the app/HTTP/kernel.php file should be run before the request is executed. The request then gets sent to the router, routes/web.php, which runs the authentication middleware, and the VerifyUserProfile middleware which verifies the user profile; if a certain user tries to access a URL that is not authorized for his profile, the user will get a Forbidden error 403. The router then sends the incoming request to its appropriate controller function. The controller may communicate with the database by executing some model functions that run SQL queries. The controller then returns the appropriate view to the browser, which may be populated with data retrieved from the database, depending on the user's request.

3.4.2 System diagram with the Leaflet component

The ”home” page and the ”add acreage to the map” page are the main pages which have direct interaction with Leaflet. Figure 3.15 illustrates how these pages interact with the system:

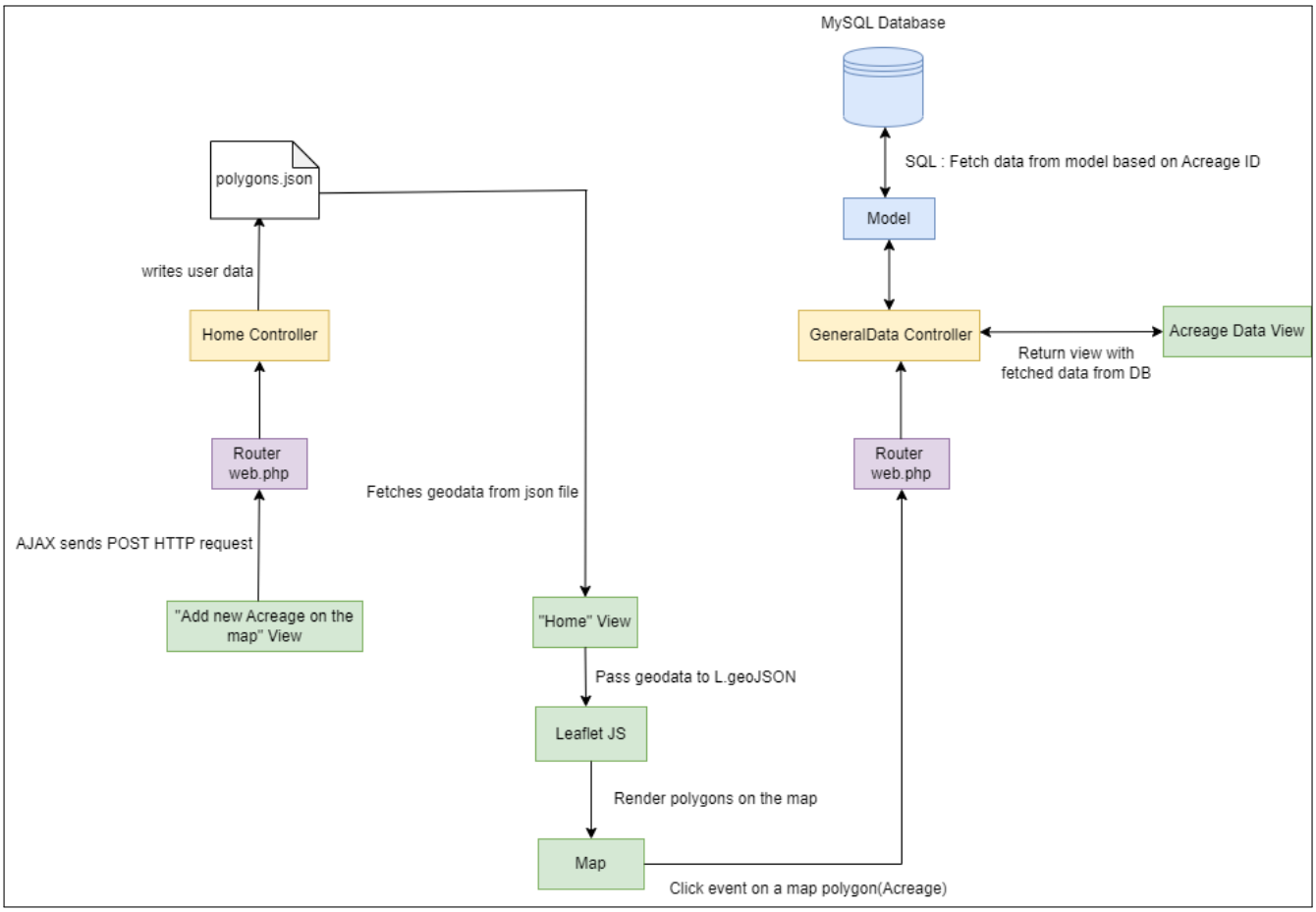


Figure 3.15: Leaflet with MVC.

The ”Data Management” actor accesses the ”Add a new acreage on the map” interface where he can enter the acreage’s geographical coordinates. The entered data gets processed and sent in, the form of geo-json, via AJAX to the appropriate route. The router calls the appropriate function in the HomeController, which in turn writes the geojson data to polygons.json. The Home view which displays the basemapgets updated as the contents of the json file are retrieved using AJAX, then passed to the L.geojson() function.

3.5 System Testing

In this section we describe the functionality of the client prototype. The core features are explained by demonstrating the user interfaces and specifying interactions between code components.

3.5.1 Authentication page

The authentication page is the entry point to the system. It displays a form that asks the user to enter a valid username and password to access the system. If the user enters invalid login credentials, he gets redirected to the authentication page with an error message being displayed.

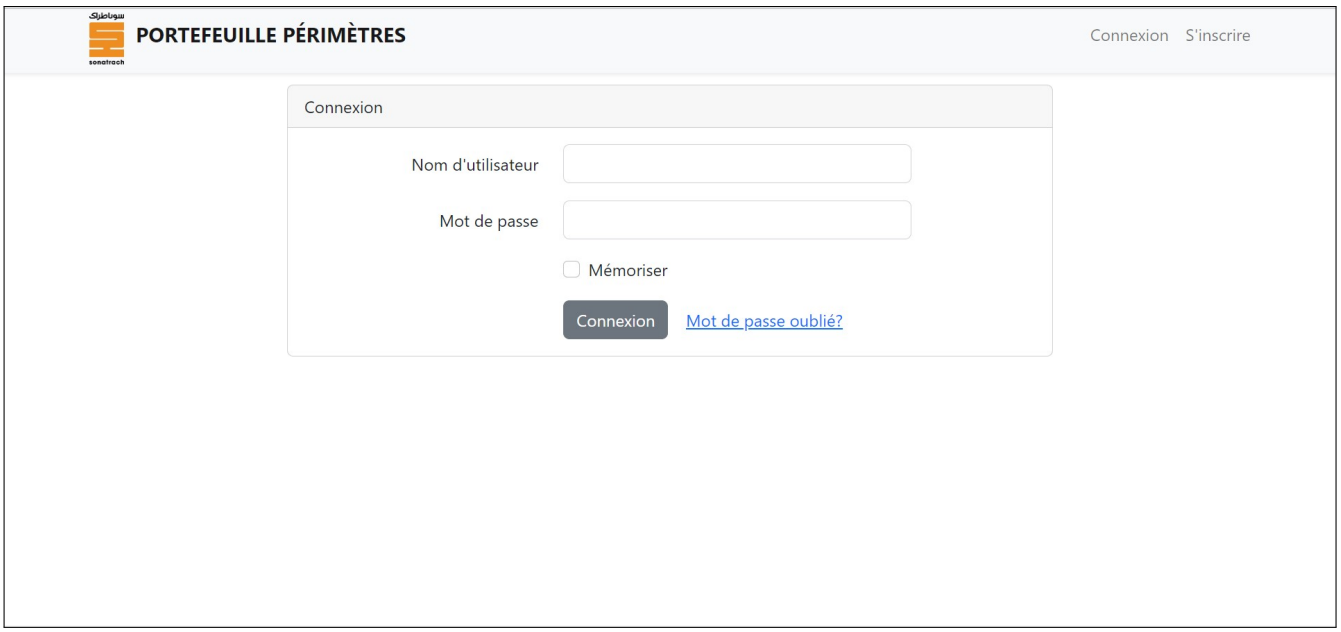


Figure 3.16: ”Authentication” page.

3.5.2 ”Home” page, profile: Planning

After the user gets successfully authenticated, he can get access to the home page. The body of the home page is the same for all user profiles and it shows the exploration acreages map. The navigation bar however, changes depending on the user profiles as each profile has restricted tasks within the system. The figure below shows the home page for a user with profile ”Planning”.

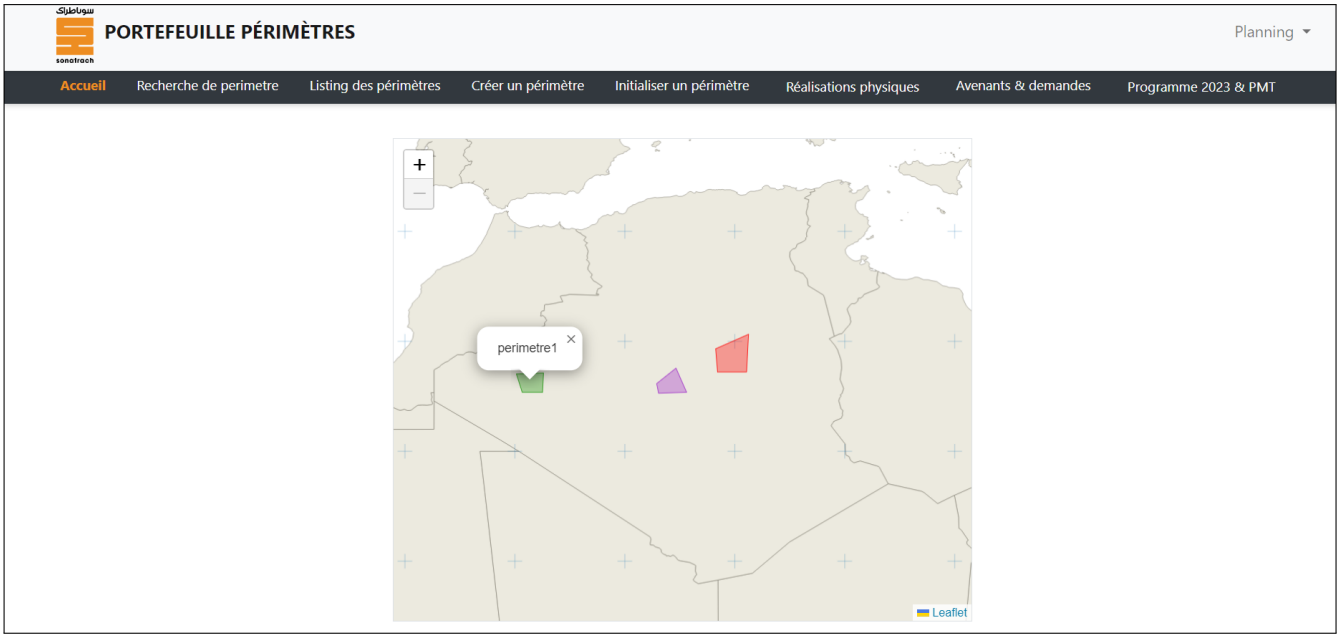


Figure 3.17: ”Home” page

3.5.3 Adding a new acreage

This page displays a form where the user can enter the information of a new acreage. By clicking the ”add” button, the information gets sent to the database. If the information is valid it gets inserted to the database, else the user gets redirect to that page with an error message.

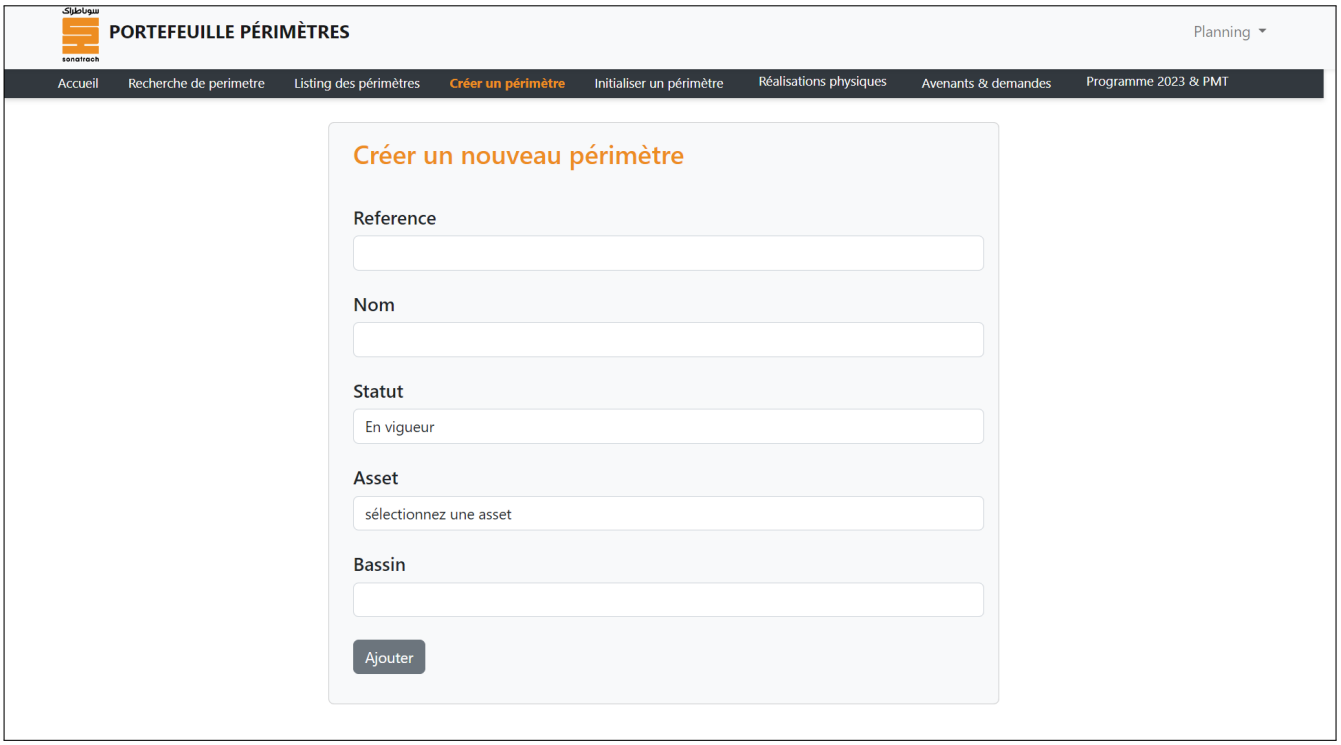


Figure 3.18: ”Add a new acreage” page.

3.5.4 ”Initialize acreage” Search page

Typically, before the user accesses any form page to add specific data to a certain acreage, the user first selects the acreage in the search page then gets directed to the form page. The ”Initialize acreage” search page shows filtered action buttons depending on the contractual situation of acreages. This is controlled via the ”Initialized” attribute as explained in chapter2 section8.

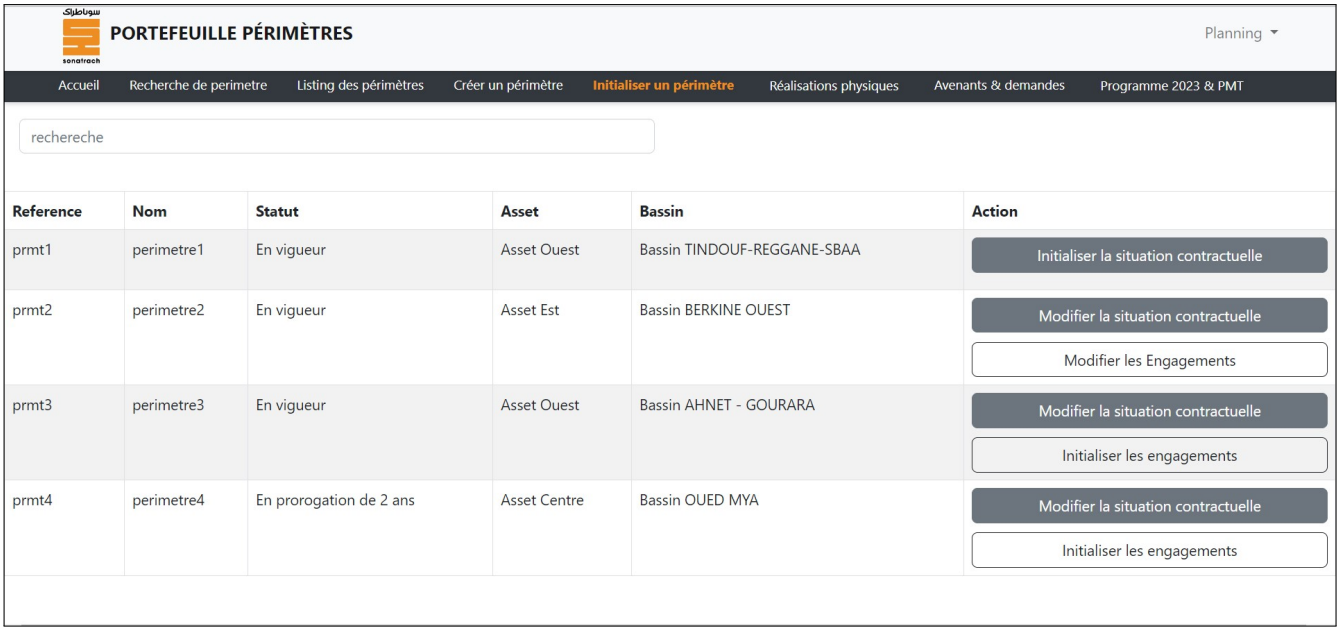


Figure 3.19: ”Initialize contractual situation and commitments ” Search page.

3.5.5 Initializing the contractual situation

The user clicks the ”initialize contractual situation” button, as shown in the previous figure. The user then can enter the general contractual information of the selected acreage in the form as shown in the figure. The user can select the number of phases of the contract such that the maximum number is 3. After the user clicks the ”Initialise” button, the data gets inserted to the database.

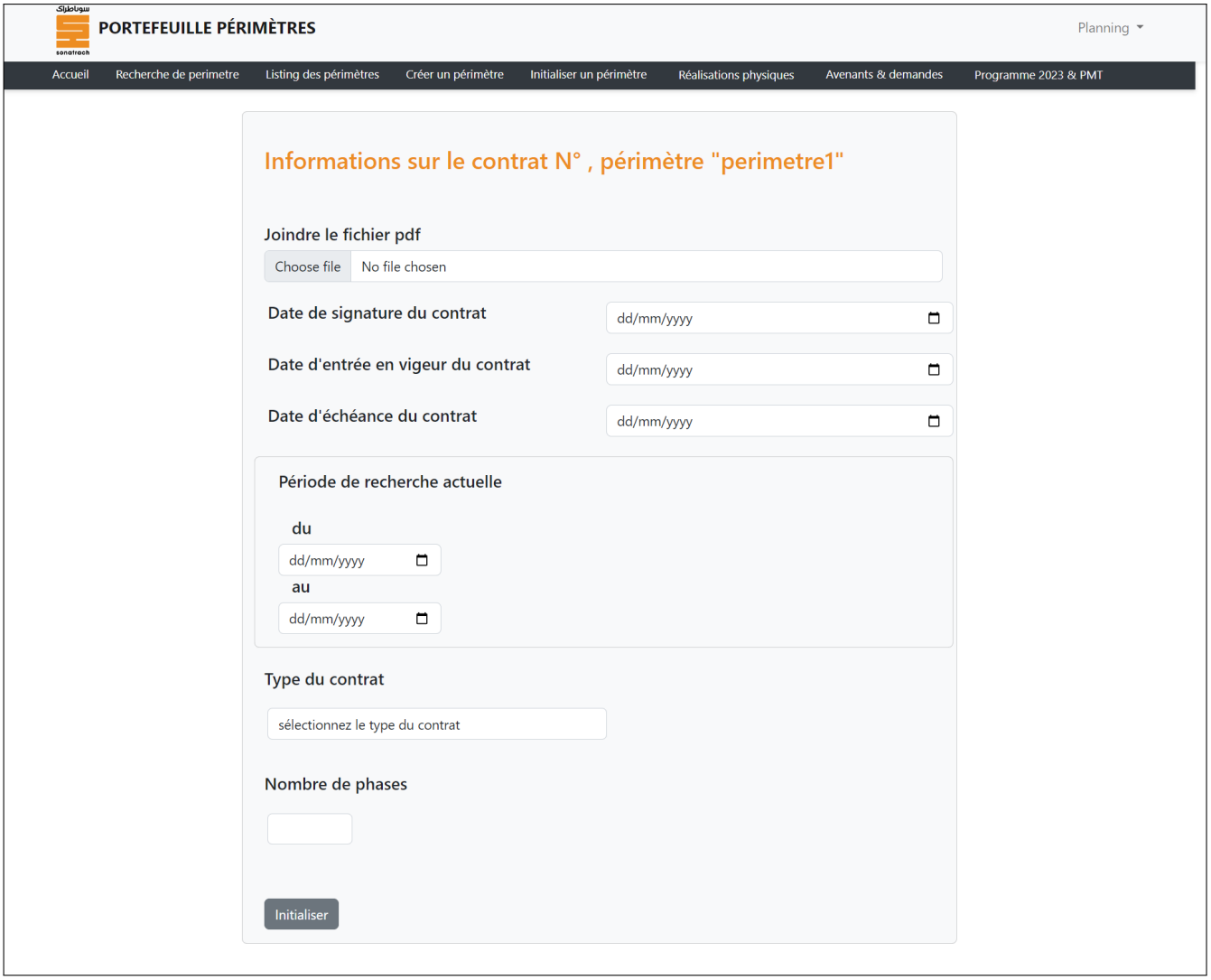


Figure 3.20: ”Initialize contractual situation” page.

3.5.6 Initializing the contractual commitments

After the acreage's contract is initialized, the user can get access to the "Initialize contractual commitments" page. This page displays a form where the user can enter the number of physical realizations per phase as well as the start and end dates of each phase.

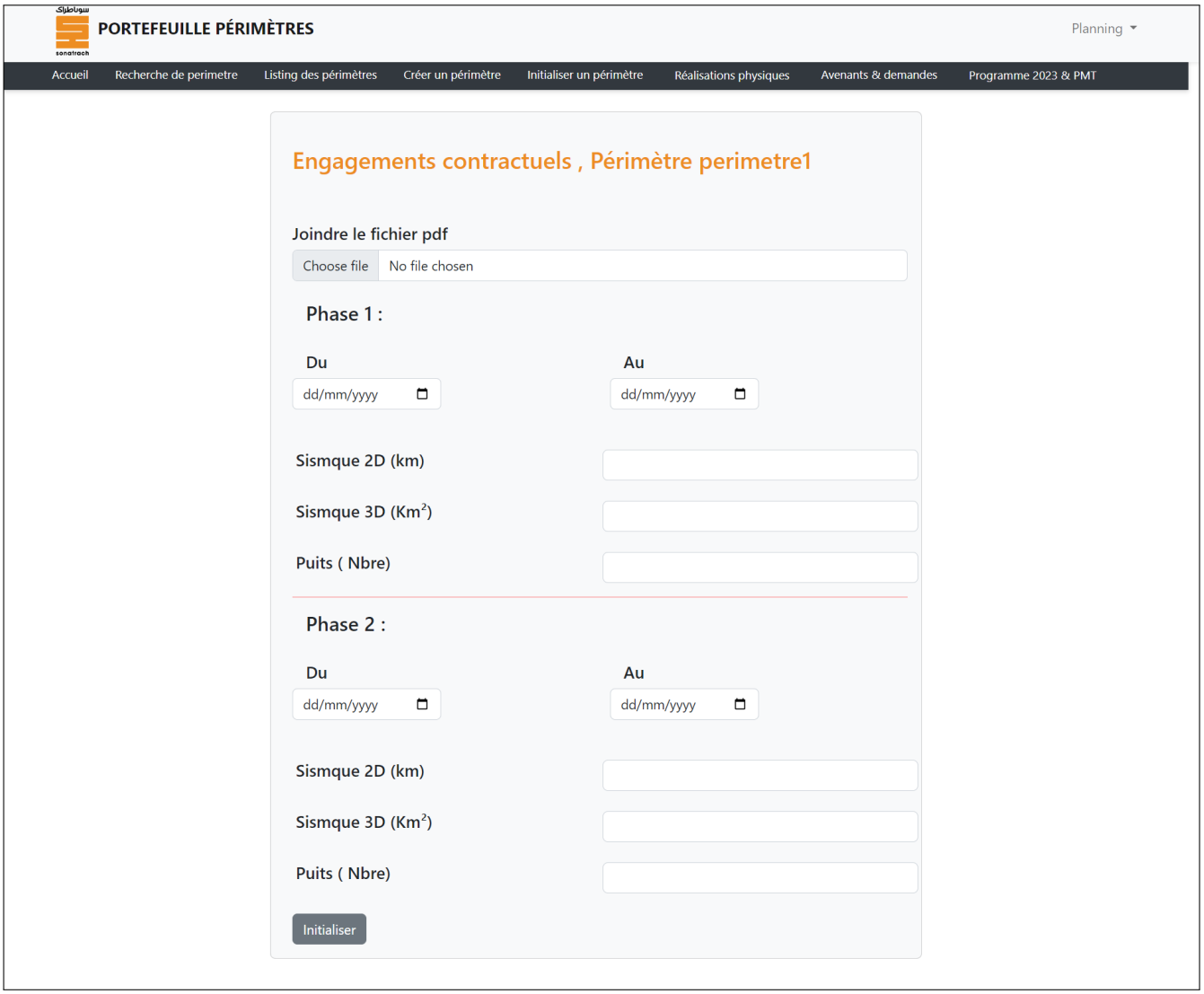


Figure 3.21: "Initialize contractual commitments" page.

3.5.7 Adding a new acreage to the map

This page can only be accessed by users having profile Data Management. The user first selects an acreage from the ”Add a new acreage to the map” search page which lists only acreages that are not shown on the map. After that, a form gets displayed where the user can insert the acreage’s geographical coordinates. After clicking the ”Add” button, an AJAX post request is sent to the appropriate controller to insert the data in the ”polygons.json” file. The user then gets directed to the updated home page.

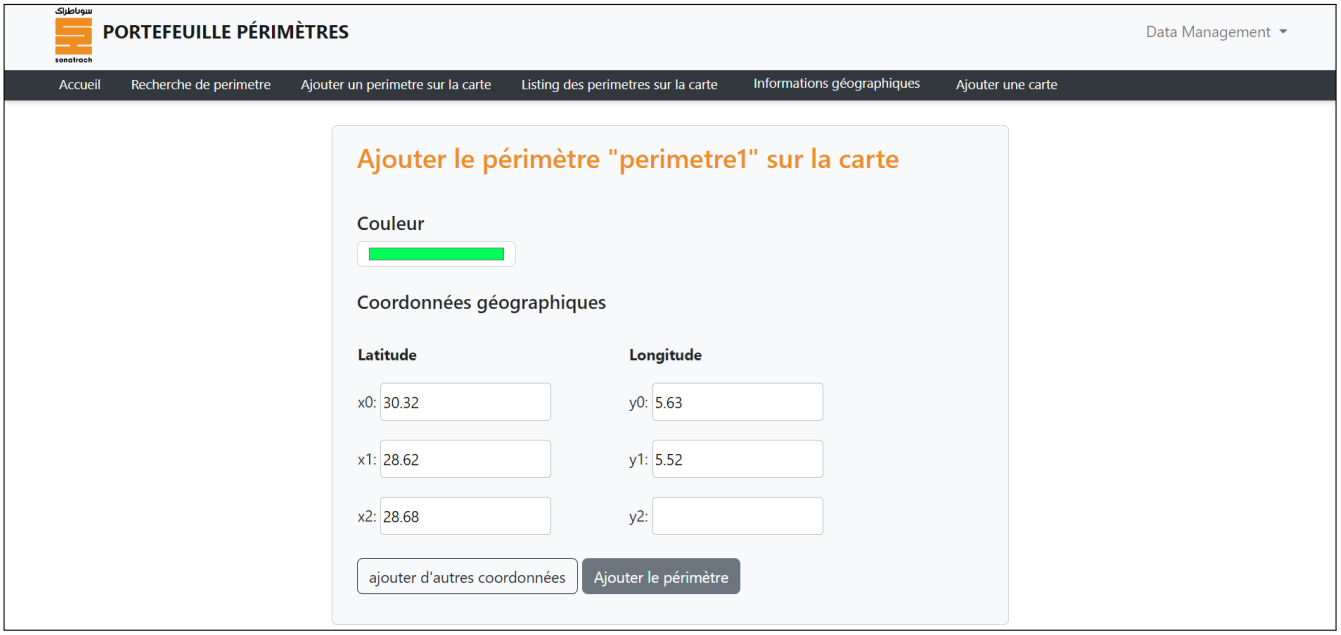


Figure 3.22: ”Add acreage to the map” page.

3.5.8 Modifying/deleting acreage's information

All the information entered by the users can be deleted or modified except an initialized acreage. Once an acreage is initialized, it cannot be deleted. We take for instance the petroleum system. Below is shown a listing of acreages which have a defined petroleum system:

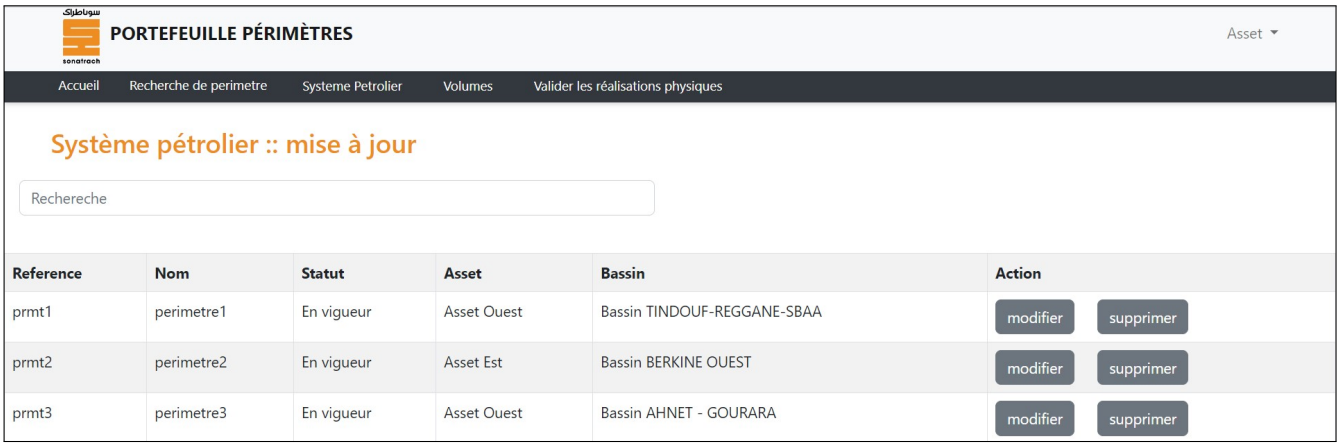


Figure 3.23: "Search existing petroleum system" page

by clicking on the "edit" button of "périmètre 1", the form below gets displayed. The form is populated with the old petroleum system information, the user makes changes where necessary and clicks the "edit" button to update the information in the database.

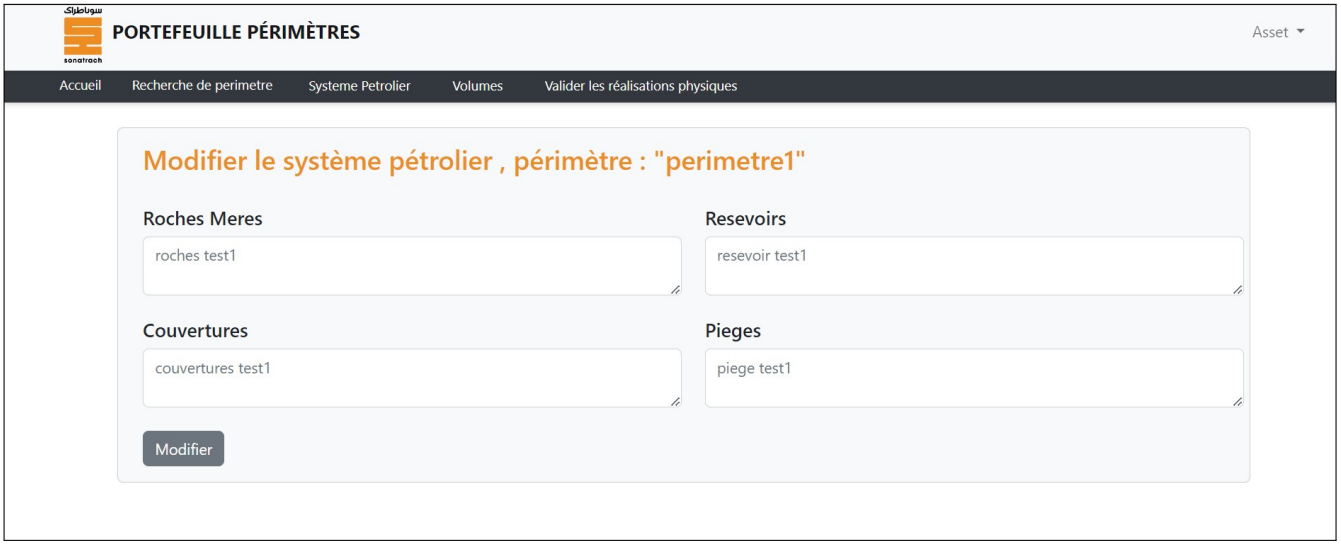


Figure 3.24: "Edit petroleum system" page.

3.5.9 Validation of realizations

Validation of realizations can be done only by users having profile Asset. The user first chooses the realization type from the drop-down menu in the navigation bar. A listing of acreages gets displayed. The user selects an acreage and then a listing of all unvalidated realizations gets displayed. The user selects a realization and then a form gets displayed. The figure below shows an example of validating a 2D seismic realization. The form is populated by data entered by Planning. If the data is correct, the user (Asset) just clicks the validate button. Else, the user makes changes where necessary and then clicks the validate button. The data then gets updated in the database. Only validated data is taken into consideration, meaning that managers can consult the validated data only.

Figure 3.25: ”Validate 2D Seismic realization” page.

3.5.10 Consulting the general information of an acreage

This page can be accessed by users having profile ”manager”. It can typically be accessed by clicking on an acreage on the map. The map acreages have associated ids stored in ”polygons.json” which are used to link to the attribute data stored in the database. This section contains all the general information data entered by other users. The manager can navigate to different sections through the sidebar.

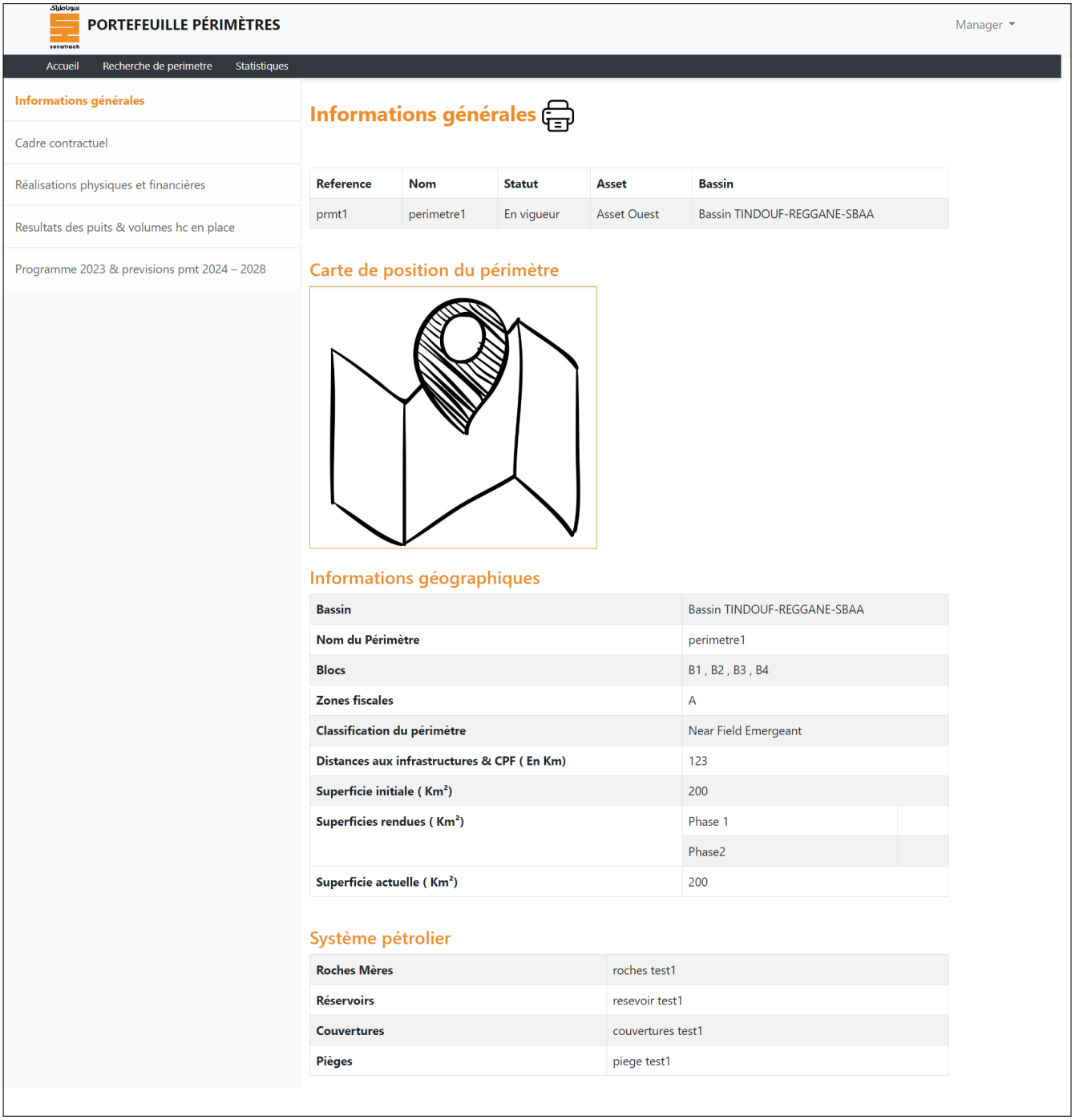


Figure 3.26: ”Acreage’s general Information” page.

3.5.11 View or print the pdf version of the general information of the acreage

After the manager clicks on the ”printer” icon shown on the previous page, ”Acreage’s general Information”, the pdf file shown below gets loaded, populated with concerned acreage data. This pdf is generated from the previous page using the DOMpdf library. The manager can downland as well as print the file.

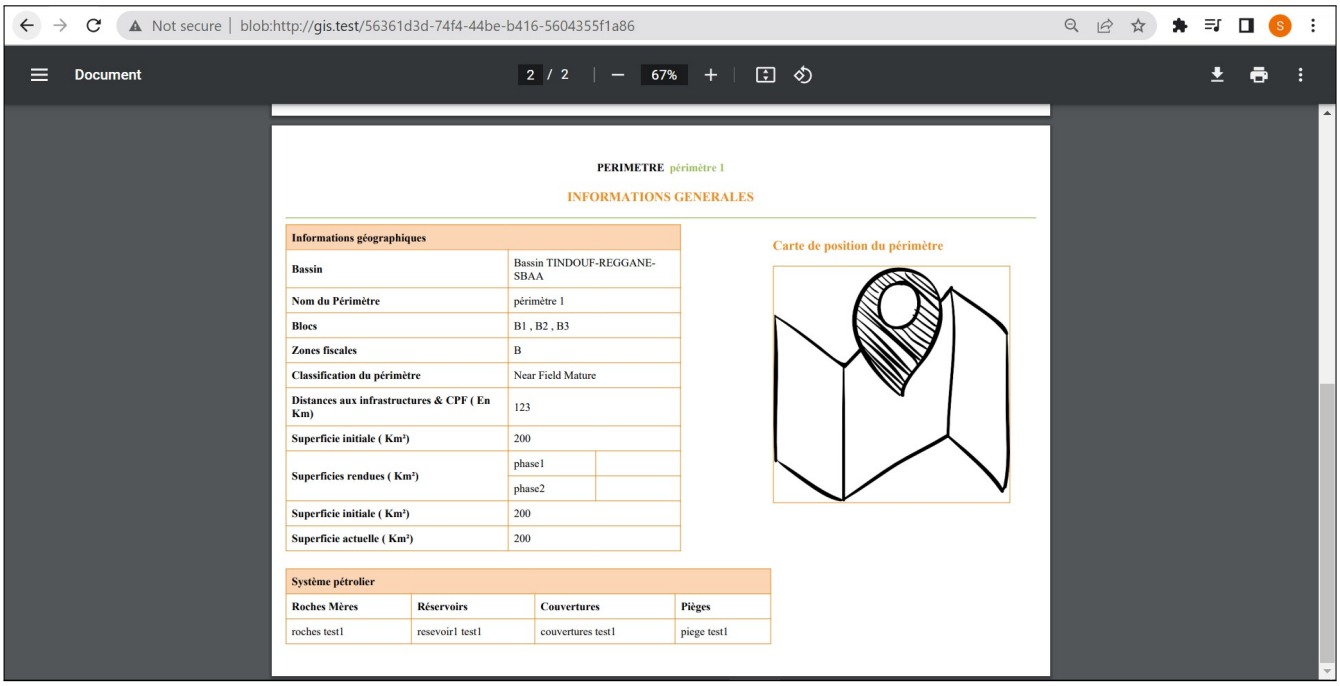


Figure 3.27: ”Print general information section of the fact-sheet” page.

3.6 ”Statistics” page

This page can be accessed only by users having profile ”manager”. It displays three charts that enable the user to visually compare multiple sets of data. The first chart for example, is a pie chart that represents the number of acreages per asset.

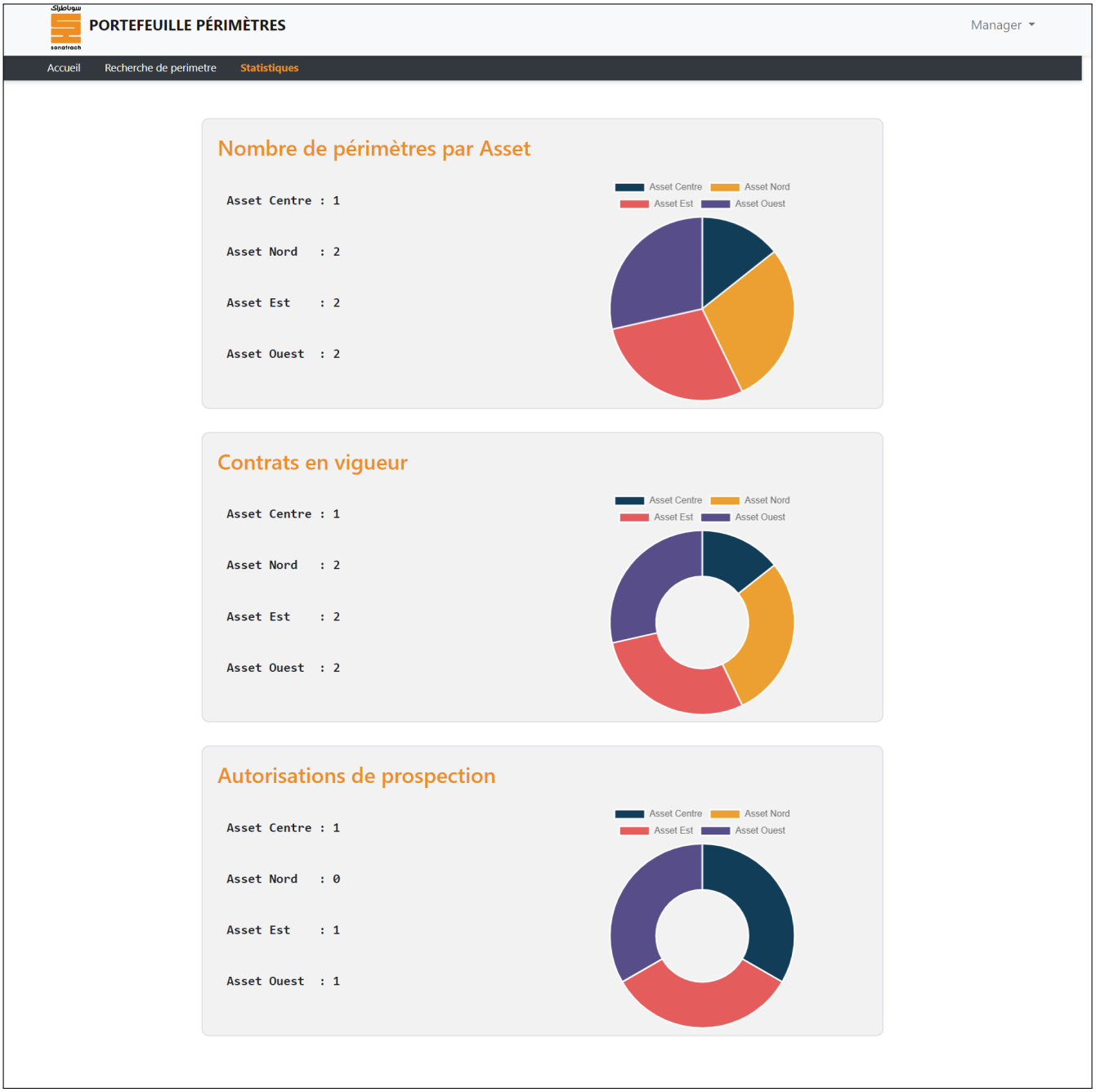


Figure 3.28: ”General statistics” page.

3.7 Conclusion

Throughout this chapter, we have introduced the design of the software architecture solution that answers the users requirements. We have defined all the tools used both in the client side and the server side, and how these tools interrelate in the system. For the client side we have used HTML, CSS, Bootstrap, and JavaScript (including AJAX, Leaflet, Chart.js). As for the server side we have used Laravel which is a back-end framework that was imposed by the company where the internship took place(Exploration Division).

We have also presented in the last section some of the main interfaces for different profiles with a brief description on each interface.

General conclusion

This study consisted of creating a basic Web-based GIS platform for the management of hydrocarbon exploration acreages. Since this system is web-based, it can enable all the concerned directorates in the Exploration Division to collaborate with each other. The stakeholders can view and share information easily and securely across the system. This can go a long way in reducing the amount of time it takes to collect and manage the data and to reach certain important decisions for managers. Managers also have the possibility to view and print the acreage reports that are generated by PHP scripts by fetching the appropriate information from the database.
This project has been an interesting experience for me, as it has expanded my knowledge, especially in programming and database management systems.

Faced problems and challenges:

- Many new concepts and skills have been learned during the period of this project. It was a little bit difficult to balance between learning the new concepts and the implementation as well as trying to link between the technical notions and understanding the system requirements that are related to a different field of study(hydrocarbon exploration).

- One technical challenge that we have faced at the beginning of the implementation of this project, is storing the acreage's geographical coordinates, to keep the acreages permanently displayed. After doing some research and learning new concepts, we have managed to store the spatial data using a JSON file together with the AJAX and L.geoJSON() function of Leaflet.

- Another technical challenge that we have encountered is the need to display the map without requiring remote map tiles, so that the map can be displayed even when the user is not connected to the Internet. We have succeeded to find a solution, by generating a map using QGIS software and specifying its bounds to the L.imageOverlay() function of leaflet.

Future work:

- It is desirable to divide the map into assets and blocks to better visualize the position of acreages.

- We would also suggest integrating the system with Single Sign-On authentication which is an authentication method that enables users to securely authenticate with multiple applications and websites by using just one set of credentials.

References

[1] Information system — definition, examples, & facts — britannica. https://www.britannica.com/topic/information-system. (Accessed on 06/10/2023).

[2] Oxford languages and google - english — oxford languages. https://languages.oup.com/google-dictionary-en/. (Accessed on 04/17/2023).

[3] energy-pedia – upstream oil & gas information. https://www.energy-pedia.com/glossary.aspx. (Accessed on 04/17/2023).

[4] What is gis? — geographic information system mapping technology. https://www.esri.com/en-us/what-is-gis/overview, . (Accessed on 06/10/2023).

[5] Paul Bolstad and Steven Manson. *GIS fundamentals: A first text on geographic information systems*, volume 620. Eider Press White Bear Lake, MN, 2008.

[6] Charles picquet maps one of the first applications of spatial analysis in epidemology : History of information. https://www.historyofinformation.com/detail.php?entryid=4237.

[7] Explore john snow's cholera map using gis data - google search. https://www.gislounge.com/john-snows-cholera-map-gis-data/. (Accessed on 03/22/2023).

[8] Waldo R. Tobler. Automation and cartography. *Geographical Review*, 49(4):526–534, 1959. ISSN 00167428. URL http://www.jstor.org/stable/212211.

[9] Canada geographic information system - wikipedia. https://en.wikipedia.org/wiki/Canada_Geographic_Information_System#Development. (Accessed on 03/24/2023).

[10] History of gis — timeline of early history & the future of gis. https://www.esri.com/en-us/what-is-gis/history-of-gis. (Accessed on 06/16/2023).

[11] Geographic information system - wikipedia. https://en.wikipedia.org/wiki/Geographic_information_system, . (Accessed on 03/25/2023).

[12] What is web gis? https://www.esri.com/about/newsroom/insider/web-gis-simply/, . (Accessed on 03/31/2023).

[13] Rouhollah Nasirzadehdizaji and Rahmi ÇELİK. Infrastructure design for making your own web-gis application with open source geoinformation technology, 05 2017.

[14] Interactive information services using world-wide web hypertext. https://web.archive.org/web/20110628195239/http://www2.parc.com/istl/projects/www94/iisuwwwh.html, . (Accessed on 03/31/2023).

[15] Plotting the past — digital archaeology. http://digital-archaeology.org/plotting-the-past/. (Accessed on 06/15/2023).

[16] Arcgis - gis wiki — the gis encyclopedia. http://wiki.gis.com/wiki/index.php/ArcGIS. (Accessed on 03/31/2023).

[17] Introducing arcgis 10.1 — arcnews. https://www.esri.com/news/arcnews/spring12articles/introducing-arcgis-101.html, . (Accessed on 04/01/2023).

[18] Rouhollah Nasirzadehdizaji and Rahmi ÇELİK. Infrastructure design for making your own web-gis application with open source geoinformation technology, 05 2017.

[19] Farid Javadnejad, Daniel Gillins, Christopher Higgins, and Matthew Gillins. Bridgedex : Proposed web gis platform for managing and interrogating multiyear and multiscale bridge-inspection images. *Journal of Computing in Civil Engineering*, 31:04017061, 11 2017. doi: 10.1061/(ASCE)CP.1943-5487.0000710.

[20] What is a web server and how does it work? https://www.techtarget.com/whatis/definition/Web-server, . (Accessed on 04/09/2023).

[21] Georg Held[1], Alias Rahman, and Sisi Zlatanova. Web 3d gis for urban environments. 09 2004.

[22] Client-server overview - learn web development — mdn. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview. (Accessed on 04/09/2023).

[23] The components of a url - ibm documentation. https://www.ibm.com/docs/en/cics-ts/5.3?topic=concepts-components-url. (Accessed on 04/09/2023).

[24] http vs https ibm - google search. (Accessed on 06/10/2023).

[25] What is a port number? - definition from techopedia. https://www.techopedia.com/definition/15702/port-number,. (Accessed on 04/09/2023).

[26] Path filters operators — microsoft learn. https://learn.microsoft.com/it-it/clarity/filters/path-filters-operators. (Accessed on 04/09/2023).

[27] Http requests - ibm documentation. https://www.ibm.com/docs/en/cics-ts/5.3?topic=protocol-http-requests. (Accessed on 04/11/2023).

[28] In introduction to http basics. https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/http_basics.html. (Accessed on 07/07/2023).

[29] C.J. Date. *An introduction to database systems*. The Systems Programming Series. Addison-Wesley Pub. Co., 1986. ISBN 9780201142013. URL https://books.google.dz/books?id=s4gk0XHjTTcC.

[30] Geographical information system (gis) - coastal wiki. https://www.coastalwiki.org/wiki/Geographical_Information_System_(GIS),. (Accessed on 06/10/2023).

[31] What is spatial data and non-spatial data? — safe software. https://engage.safe.com/blog/2021/10/non-spatial-data-difference-fme/,. (Accessed on 06/10/2023).

[32] Map tile servers — mbedded.ninja. https://blog.mbedded.ninja/space/map-tile-servers/. (Accessed on 04/09/2023).

[33] Optimization of openstreetmap background vector tile production for an on request service: Makina maps — by frédéric rodrigo — medium. https://medium.com/@frederic.rodrigo/optimization-of-openstreetmap-background-vector-tile-production-for-an-on-request-service-makina-5f6dba98a232. (Accessed on 07/07/2023).

[34] Application of gis in health sciences. https://grindgis.com/gis/application-of-gis-in-health-sciences,. (Accessed on 04/03/2023).

[35] Applications of gis and geospatial analyses in covid-19 research: A systematic review - pmc. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8822139/,. (Accessed on 04/03/2023).

[36] Gis systems lead response to covid-19. https://www.esri.com/about/newsroom/arcuser/gis-systems-lead-response-to-covid-19/. (Accessed on 04/03/2023).

[37] 1 new message. https://www.igismap.com/gis-in-disaster-management-system/. (Accessed on 04/03/2023).

[38] Addressing the climate crisis. https://storymaps.arcgis.com/stories/a68973822394482d97c06e0e1b40ac59. (Accessed on 04/03/2023).

[39] Top 10 uses of gis in oil and gas - exprodat. https://www.exprodat.com/blog/top-10-uses-of-gis-in-oil-and-gas-4/. (Accessed on 06/13/2023).

[40] Agence nationale pour la valorisation des ressources en hydrocarbures — wikipédia. https://fr.wikipedia.org/wiki/Agence_nationale_pour_la_valorisation_des_ressources_en_hydrocarbures. (Accessed on 06/13/2023).

[41] Oxford learners dictionaries : amendment. https://www.oxfordlearnersdictionaries.com/definition/english/amendment. (Accessed on 07/07/2023).

[42] Using seismic technologies in oil and gas exploration — earth — earthsky. https://earthsky.org/earth/bob-hardage-using-seismic-technologies-in-oil-and-gas-exploration/, . (Accessed on 07/08/2023).

[43] James Rumbaugh. *The unified modeling language reference manual*. Pearson Education India, 2005.

[44] What is activity diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/, . (Accessed on 06/15/2023).

[45] Alan Dennis, Barbara Wixom, and David Tegarden. *Systems analysis and design: An object-oriented approach with UML*. John wiley & sons, 2015.

[46] Database design phase 2: Conceptual design - mariadb knowledge base. https://mariadb.com/kb/en/database-design-phase-2-conceptual-design/. (Accessed on 05/31/2023).

[47] Uml class diagram tutorial. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/. (Accessed on 06/02/2023).

[48] Back-end web architecture — codecademy. https://www.codecademy.com/article/back-end-architecture. (Accessed on 05/06/2023).

[49] Jennifer Niederst Robbins. *Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics*. " O'Reilly Media, Inc.", 2012.

[50] what is bootstrap codecademy - google search. https://www.codecademy.com/resources/docs/open-source/bootstrap. (Accessed on 05/09/2023).

[51] What is a framework? https://www.codecademy.com/resources/blog/what-is-a-framework/, . (Accessed on 05/06/2023).

[52] Json. https://www.json.org/json-en.html. (Accessed on 05/08/2023).

[53] 19.1 introduction to json - java platform, enterprise edition: The java ee tutorial (release 7). https://docs.oracle.com/javaee/7/tutorial/jsonp001.htm. (Accessed on 05/08/2023).

[54] jquery. https://jquery.com/. (Accessed on 06/13/2023).

[55] Paul Crickard III. *Leaflet. js essentials*. Packt Publishing Ltd, 2014.

[56] L.class. https://www.wrld3d.com/wrld.js/latest/docs/leaflet/L.Class/. (Accessed on 05/02/2023).

[57] Documentation - leaflet - a javascript library for interactive maps. https://leafletjs.com/reference.html#, . (Accessed on 05/04/2023).

[58] Using geojson with leaflet - leaflet - a javascript library for interactive maps. https://leafletjs.com/examples/geojson/, . (Accessed on 05/20/2023).

[59] Helder Da Rocha. *Learn Chart. js: Create interactive visualizations for the web with chart. js 2*. Packt Publishing Ltd, 2019.

[60] Laragon: Simple, flexible, and modern development server environment for windows • php.watch. https://php.watch/articles/laragon-windows-php. (Accessed on 05/05/2023).

[61] Documentation — laragon - portable, isolated, fast & powerful universal development environment for php, node.js, python, java, go, ruby. https://laragon.org/docs/index.html, . (Accessed on 05/05/2023).

[62] Apache — definition & facts — britannica. https://www.britannica.com/technology/Apache-Web-server. (Accessed on 05/05/2023).

[63] Heidisql - mariadb, mysql, mssql, postgresql and sqlite made easy. https://www.heidisql.com/. (Accessed on 05/06/2023).

[64] What is php: Hypertext preprocessor (php)? - definition from techopedia. https://www.techopedia.com/definition/24406/php-hypertext-preprocessor-php, . (Accessed on 05/06/2023).

[65] Convert html to pdf in php with dompdf - codexworld. https://www.codexworld.com/convert-html-to-pdf-php-dompdf/. (Accessed on 06/12/2023).

[66] Giedrius Varoneckas, Arvydas Martinkenas, Jurgita AndruŁkienė, Albinas Stankus, Lina Mažrimaite, and A. Livens. Monitoring system for evaluation of operator functional status on sea ships. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 10:309–314, 07 2016. doi: 10.12716/1001.10.02.13.

[67] Building mvc applications in php laravel: Part 1. https://www.codemag.com/Article/2205071/Building-MVC-Applications-in-PHP-Laravel-Part-1. (Accessed on 05/06/2023).

[68] Sanjib Sinha and Hiren J Dave. *Beginning Laravel*. Springer, 2017.

[69] Laravel , database: Migrations. https://laravel.com/docs/5.1/migrations, . Accessed: 2023-05-06.

[70] Learn the facade design pattern - learncsdesign. https://www.learncsdesign.com/learn-the-facade-design-pattern/. (Accessed on 05/22/2023).

[71] Laravel , facades. https://laravel.com/docs/10.x/facades, . Accessed: 2023-05-06.

[72] Laravel , blade. https://laravel.com/docs/10.x/blade, . Accessed: 2023-05-06.

[73] Matt Stauffer. *Laravel: Up & running: A framework for building modern php apps*. O'Reilly Media, 2019.

[74] İsa Avci, Murat Koca, and Büşra Uysal. The place of stock photography as a digital commerce in turkey. 10 2021.

[75] Request lifecycle - laravel - the php framework for web artisans. https://laravel.com/docs/10.x/lifecycle#http-console-kernels. (Accessed on 05/09/2023).

[76] What is middleware? — ibm. https://www.ibm.com/topics/middleware, . (Accessed on 05/09/2023).

[77] Middleware - laravel - the php framework for web artisans. https://laravel.com/docs/10.x/middleware. (Accessed on 05/20/2023).

[78] Artisan cli - laravel - the php framework for web artisans. https://laravel.com/docs/5.0/artisan. (Accessed on 05/14/2023).

[79] Directory structure - laravel - the php framework for web artisans. https://laravel.com/docs/10.x/structure#the-providers-directory. (Accessed on 05/14/2023).

[80] How to perform database seeding in laravel - artisans web. https://artisansweb.net/database-seeding-laravel/. (Accessed on 05/14/2023).

[81] Laravel documentation: Configuration. https://laravel.com/docs/10.x/configuration, . (Accessed on 07/09/2023).

Appendix A: Company Overview

SONATRACH is the national state-owned oil and gas company of Algeria. It was founded in 1963 and nationalized on 24/2/1971. The mission of the company is realized through several divisions:

- **Production:** responsible for the development and exploitation of hydrocarbon deposits.
- **Transportation:** ensures the operation and management of the pipeline transportation network.
- **Liquefaction:** takes charge of the operation of natural gas liquefaction complexes.
- **Marketing:** takes charge of the sale of hydrocarbons at the national level.
- **Exploration:** takes charge of the development of hydrocarbon research and oil prospecting activities.

Exploration Division

In 1972, the exploration directorate was created. In 1987, this directorate was elevated to the rate of division, which was reorganized into several directorates by decision A.364 of 18/07/1988. The exploration division has been one of the operational divisions of the upstream activity[1] of SONATRACH, and its main missions are:

- Conduct and develop prospecting and research activities of Hydrocarbons.
- Participate with the other divisions in the exploration tenders in Algeria and abroad.
- Participate in the evaluation of partnership offers on exploration projects in Algeria and abroad.
- Support the implementation of the company's exploration strategy.
- Prepare, establish and recommend technical exploration programs and their follow-up.
- Manage and follow up the contracts in person and in association .

[1] upstream activities include exploration , acquisition , drilling , developping and producing oil and gas. These activities take place before oil production.

- Develop and conduct analytical work in the field of geology and geophysics.
- To develop expertise in the field of exploration.

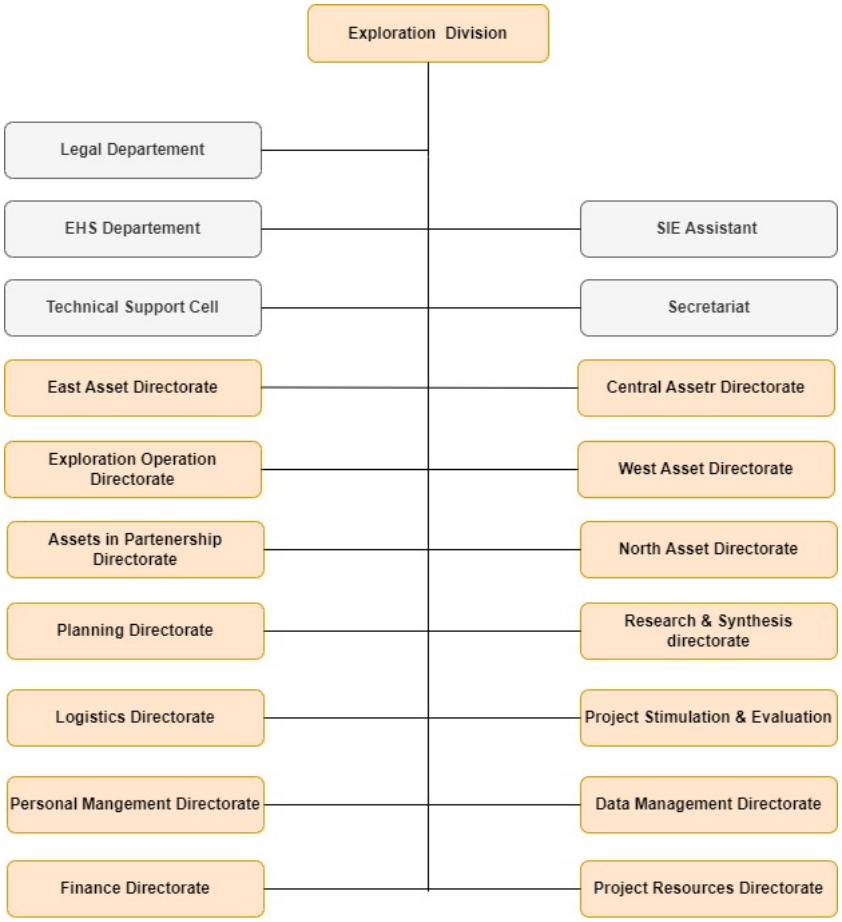


Figure A.1: Organization Chart of the Exploration Division.

Furthermore, the Data Management Directorate is organized into five departments:

- Data Bank Department.
- Processing Department.
- Data Servicing Department.
- Data Heritage Department.
- Information Technology Department.

The information technology department, where the internship was held, is organized into 4 groups:

- Networks and Computer Security Group.
- Maintenance and Computer Systems Group.

- G&G Business Software Support Group.
- Internet Services/ Intranet & Help-desk Group.

Some essential missions of the information technology department are:

- The provision of information technology services and the provision of specific and specialized IT resources and solutions to the structures of the exploration division.
- The management of the IT activity on all its aspects: hardware and software, at the level of all the structures of the division.
- Management, control and loading of geological and geophysical project data for business software platforms.
- Integration of interpretation processes in the Data Management environment.
- Management and administration of the different platforms (Windows, Linux).
- Participation in the development and implementation of the IT policy of the exploration division and the company.
- The standardization of the acquisition of equipment, the updating of the IT infrastructure.
- Coordination with the other IT structures of the upstream branch and the company regarding upstream or enterprise IT projects.
- Development and implementation of the security of the division's information systems and implementation of a network infrastructure.

Appendix B: Assets, Blocs and Basins

The geographic area of Algeria is divided up into four assets : north ,centre ,west ,east.
The term **asset** means a resource with economic value that an individual, corporation, or country owns or controls with the expectation that it will provide a future benefit [1].

A **Hydrocarbon asset** means, at any time, all hydrocarbon (oil and/or gas) fields owned or held by any group company (SONATRACH)[2]. The Asset directorates mentioned in the company overview deal with hydrocarbon exploration in these four geographical assets, in addition to an asset in partnership which is a directorate that deals with contracts in association.

On the other hand, each asset is divided up into **basins**.

These basins refer to the Asset's **departments**, there exist also a "petroleum basin" or sedimentary basin which includes a diverse collection of rocks and sediments, but most importantly, it contains source rocks which are a specific shale formations in a basin and are where oil and gas are born.

Furthermore, a single basin is composed of multiple blocks. A **block** is an area designated for hydrocarbon exploration and **production** including **mines**, oil and gas. Each block is identified by a unique number. An exploration acreage can be part of a single block, or can take a portion from multiple blocks.

[1] ADAM BARONE. What is an asset? definition, types, and examples. https://www.investopedia.com/terms/a/asset.asp

[2] Hydrocarbon assets definition — law insider.https://www.lawinsider.com/dictionary/hydrocarbon-assets

Appendix C: UML diagrams

C.1 Use case diagrams

A use case diagram consists of four major elements described as follows:

(a) System: the product in question that is being developed. This is represented by a box that encompasses the use cases[1].

(b) Actor: an actor is an idealization of a role played by an external entity (or entities) interacting with a system or a subsystem. An actor may be a human, a computer system, or some executable program. A human actor is drawn as a small stick person with the name below it. Each actor participates in one or more use cases[2]. There exist two types of actors in use case diagrams[1]:

- Primary actors: they initiate use cases and interact with the system. They are usually placed on the left side of the system.
- Secondary actors: are used by the system but they do not interact with the system on their own, hence they are dependent and reactionary. They are displayed on the right side of the system.

(c) Use case: A use case represents a functionality that the system is expected to implement. A use case is drawn with an ellipse with its name inside. The diagram does not visually represent the use case details other than the unique name. Depending on the application domain, a use case may be broken down into several use cases which are connected through *include* or *extend* relationships. Use cases are generally initiated by primary actors. Secondary actors such as a database are used by the system through a set of use cases[3].

[1]What is a use case diagram? `https://www.educative.io/answers/what-is-a-use-case-diagram`

[2]The unified modeling language reference manual, Rumbaugh, James, 2005

[3]Informal Semantics for UML Use Case Diagrams: `https://cs.uwlax.edu/~mzheng/CS743Fall19/UseCaseDiagrams.html`

(d) Relationships: a relationship represents a connection between the system elements. In use case diagrams, there are four main types of relationships:

- **Association:** represents the communication path between an actor and a use case that it participates in. It is represented by a solid line[2].
- **Include:** represents a relationship between two use cases. If a use case A (the base use case) includes another use case B (the included use case), then the implementation of A requires the implementation of B to fulfil its task. However, B is independent on its own. The include relationship is represented by a dashed line with an arrow that points towards the included use case with the notation <<include >>[3].
- **Extend:** represents a relationship between two use cases. If a use case B extends another use case A, then the implementation of A *may conditionally* include the implementation of B to fulfil its task. B in this case is dependent on A and cannot exist on its own. For this reason, B cannot extend more than one use case. The extend relationship is represented by a dashed line with an arrow that points towards the base use case with the notation <<extend >>[3].
- **Generalization:** it is a relationship between a general use case and a more specific use case that inherits and adds features to it. It is represented by an arrow that points to the parent use case or actor[2].

C.2 Activity diagrams

Basic components of activity diagram [4]:

- **An initial node:** portrays the beginning of a set of actions or activities.
- **An action:** is a simple, non-decomposable piece of behaviour. It is labelled by its name.
- **A control flow:** shows the sequence of execution.
- **A decision node:** is used to represent a test condition to ensure that the control flow only goes down one path. It is labelled with the decision criteria to continue down the specific path.
- **A merge node:** is used to bring back together different decision paths that were created using a decision node.

[4] Book: Systems analysis and design: An object-oriented approach with UML, by Dennis, Alan and Wixom, Barbara and Tegarden, David 2015

- **A fork node :** is used to split behaviour into a set of parallel or concurrent flows of activities/actions.
- **A join node :** is used to bring back together a set of parallel or concurrent flows of activities/actions.
- **A final-activity node:** is used to stop all control flows and object flows in an activity/action.

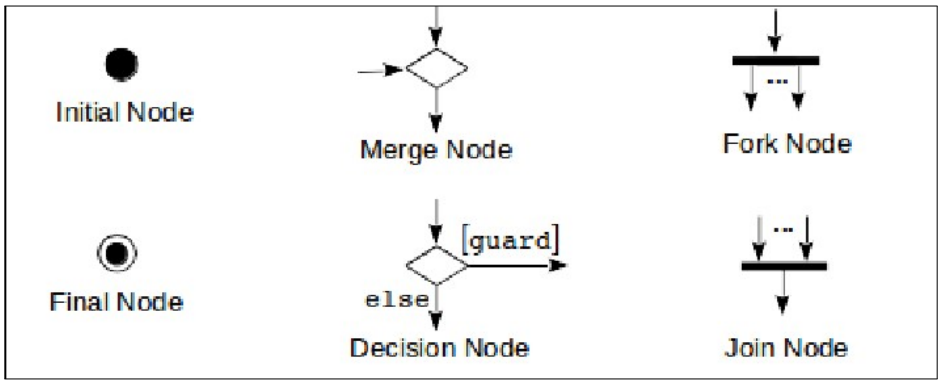


Figure C.1: Basic components of activity diagram.

C.3 Sequence diagrams

A sequence diagram consists of the following elements[4]:

(a) An actor: is a person or system that benefits from and is external to the system. Actors participate in a sequence by sending and/or receiving messages. An actor is placed at the top of the diagram and is represented as stick figure and, when non-human actors are involved, as rectangles with<<actor>>inside .

(b) An object: participates in a sequence by sending and/or receiving messages. It is placed across the top of the diagram and it is represented by a box containing the name and type of the instance.

(c) A lifeline: denotes the time the instance exists during a sequence. It is represented by a dotted line and contains an X at the point at which the class no longer interacts.

(d) An execution occurrence: is a long narrow rectangle placed atop a lifeline. it indicates when an object or an actor is sending or receiving messages.

(e) A message: conveys information from one object to another one or from an actor to an object. An *operation call* is labelled with the message being sent and a solid arrow, whereas *a return* is labelled with the value being returned and shown as a dashed arrow. An object can also send a message to itself, this is known as *self-delegation*. Messages generally refer to program methods.

(f) A guard condition: represents a test that must be met for the message to be sent.

(g) A fragment: a sequence diagram fragment is a rectangular frame drawn over a portion of the diagram. It represents conditional structures that affect the flow of messages. The frame is labelled in the upper left corner. This specification is called an interactive operator in UML [5].

The most commonly used fragments are[5]:

- **Option:** represents a simple behavioural choice between two options. The interaction operator is ”opt”. The sequence fragment modelled will either occur or not, based on a particular condition. It models the *if then* programming statement.
- **Alternative:** models the classic *if-then-else* programming statement. The interaction operator is ”alt”. In this fragment, the message flow represents a mutually exclusive choice between two paths. The process flows along one of the two paths.
- **Loop:** is used to represent a repetitive sequence. The interaction operator is ”loop”.

C.4 Class Diagram

C.4.1 Visibility

Visibility determines whether attributes and methods of specific classes can be seen and used by other classes [6].

- Private: denoted by ”-”, can be accessed only within the class.
- Protected: denoted by ”#”, can be accessed by the same class or descendent classes.
- Public: denoted by ”+”, can be accessed from within or outside the class.
- Package: denoted by ” ”, can be accessed by classes within the same package as the container class.

C.4.2 Relationships between classes

- Association: A broad term that includes almost any logical connection or relationship between classes [7].

[5]Sequence diagram with fragments — Gleek — Gleek: `https://www.gleek.io/blog/sequence-diagram-fragments`

[6]Visibility in domain modelling class diagrams - IBM Documentation: `https://www.ibm.com/docs/en/radfws/9.6.1?topic=classifiers-visibility`

[7] UML Class Diagram Relationships Explained with Examples — Creately: `https://creately.com/guides/class-diagram-relationships/#inheritancegeneralization`

- Inheritance: One associated class is a child of another class because it inherits the same functionalities of the parent class. In other words, the child class is a specific type of the parent class[7].
- Aggregation: Refers to the formation of a particular class as a result of one class being aggregated or built as a collection[7].
- Composition: The contained class depends on the life cycle of the container class. This means that the contained class will be obliterated when the container class is destroyed[7].

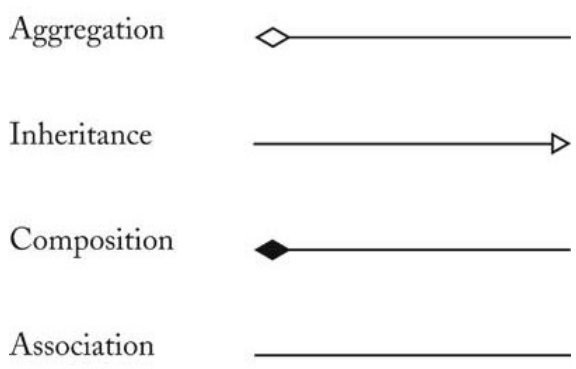


Figure C.2: Relationships in a class diagram

Appendix D: Relational database model

A relational database is a type of database where data is perceived by the user as tables. The operators available to the user for retrieval are operators that derive new tables from old ones[1].

This is done via unique identifiers called relation keys that demonstrate the different relationships which exist between tables. Each record (row) in a table is typically identified by a unique column or group of columns. This column, or columns, is called the primary key (PK) of the table and ensures the entity integrity of the table .

A foreign key is a column, or a combination of columns, that is used to establish and enforce a link between the data in two tables to control the data that can be stored in the foreign key table. In a foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table[2].

D.1 Types of relationships in a relational database

There are three main types of relationships in a relational database[3]:

- **One-to-one:** occurs when each row in a given table has only one related row in another table.
- **One-to-many:** occurs when one record in a given table A is related to one or more records in another table B.
- **Many-to-many:** occurs when multiple records in a given table are related to multiple records in another table.

[1]Book: An introduction to database systems by Date, C.J, 1986

[2]Primary and Foreign Key Constraints - SQL Server — Microsoft Learn: `https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints`

[3]Relationships in SQL - Complete Guide With Examples - Devart Blog: `https://blog.devart.com/types-of-relationships-in-sql-server-database.html`

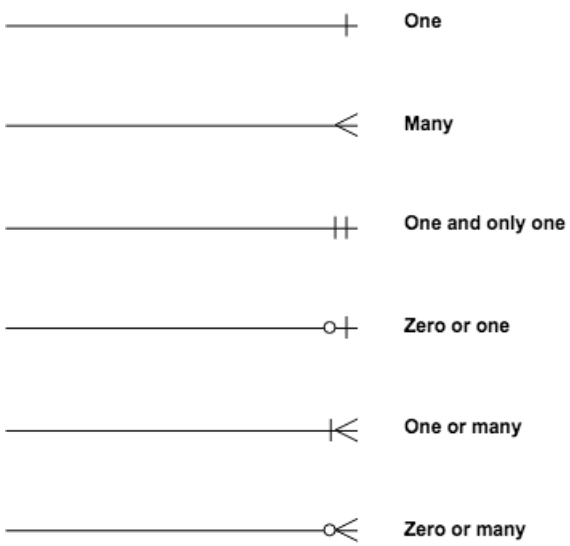


Figure D.1: Database relationships symbols in ER Diagram

D.2 Advantages of a relational database

- **Data categorization:** it is easy for database administrators to categorize and store data in a relational database, which can then be queried and filtered to extract information for reports[4].
- **Accuracy:** data is stored only once, eliminating data duplication in storage procedures[4].
- **Convenience:** users can easily execute complex queries using SQL, the dominant query language used in relational databases[4].

[4]What is a Relational Database? `https://www.techtarget.com/searchdatamanagement/definition/relational-database`

Appendix E: Listings

E.0.1

```

L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    maxZoom: 15,
    attribution: '&copy; <a href="http://www.openstreetmap.org/copyright">
   OpenStreetMap</a>'
}).addTo(map);
```

Listing E.1: Creating a map object in Leaflet

E.0.2

```

p = L.polygon([
              [27.21, 2.54],
              [27.41, 4.83],
              [26.19, 4.65],
              [26.47, 3.11]
            ],
            {
                "color": "red",
                "weight": 0.4,
                "opacity": 1,
                "fillOpacity": 0.4
            }
            ).addTo(map);
```

Listing E.2: Creating a polygon in Leaflet

E.0.3

```
$.ajax({

    url:"polygons.json",
    dataType:'json',
    type:'get',
    cache:false,

    success:function(data){

        var geojson_objects = L.geoJSON(data).addTo(map);

          geojson_objects.eachLayer(function (layer) {

            var p = layer;

            p.bindPopup(
              layer.feature.properties.name
            );

            p.on('mouseover', function (e) {
            this.openPopup();
            });
            p.on('mouseout', function (e) {
                this.closePopup();
            });

            layer.setStyle({"color": layer.feature.properties.color,"weight":
   0.6,"opacity": 1,"fillOpacity": 0.35 });


    },
    error: function(){
        console.log('error cannot get data from polygons.json')
    }
})
```

Listing E.3: Using AJAX JQuery to fetch polygons from json and add them to the map

E.0.4

```
[
     { "type": "Feature",
        "geometry": {
             "type": "Polygon",
             "coordinates":[
                 [
                   [
                       "2.54",
                       "27.21"
                   ],
                   [
                       "4.83",
                       "27.41"
                   ],
                   [
                       "4.65",
                       "26.19"
                   ],
                   [
                       "3.11",
                       "26.47"
                   ]
                 ]
             ]
             },
             "properties":{
                 "color":"red",
                 "name":"Poly 1"
             }}
]
```

Listing E.4: JSON file with only one polygon object

E.0.5

```
1
2 var map = L.map('map', {
3   crs: L.CRS.EPSG4326,
4   minZoom: -1,
5   maxZoom: 5
6 })
7 var bounds =[[16.643 , -12.230], [ 39.734 , 20.427]] // [y,x]
8 var image = L.imageOverlay('algeria.png', bounds)
9 map.setView([28.0339 , 1.6596], 5);
10
11 image.addTo(map);
```

Listing E.5: Loading a map image using L.ImageOverlay