****

****

Academic Year 2021-2022

Erasmus Project

**COVID-19 Monitoring web application**

**Isaac Sanz Ortolano**

Supervisor:

Prof. Carlos Ferreira

This project work does not include the criticisms and suggestions made by the Jury

Instituto Superior de Engenharia do Porto

INSTITUTO POLITÉCNICO DO PORTO

Porto

July 2022

**ABSTRACT**

Two years ago, COVID-19 was officially declared a pandemic, and since then the impact that it has had on society has led healthcare professionals to search for technologies which can help in containing the virus. One of the most popular technologies is web monitoring applications, as they have become one of the most effective measures in controlling the virus. This study presents the development of a web application for monitoring COVID-19 metrics and other economic, sociodemographic, and health features considered relevant for the study of the impact of the virus. For this purpose, Vue.js has been used to develop the frontend of the application and Flask for the backend. The development of this process has been done using the Scrum agile methodology. As a result, a user-friendly web application which anyone can access to get information or analyze the impacts of the pandemic has been developed**.**

**Keywords:** Web Application, COVID-19, Vue.js, Flask.

**CONTENTS**

# INTRODUCTION

This chapter introduces the context of the problem, the description of the project, the methodologies used, the objectives to be achieved and the structure of this report.

## 1.1 PROJECT CONTEXT

Coronavirus disease (COVID-19) is a highly contagious respiratory disease caused by the SARS-CoV-2 virus, which causes mild illness in most people but can lead to severe, life-threatening illness in older adults or people with severe health problems [1].

On March 11, 2020, the World Health Organization (WHO) declared a pandemic for COVID-19 [2,3]. This disease was first reported in Wuhan, China in December 2019 until it subsequently spread globally [3], with the first cases emerging from Spain in February [4]. Since then, 540,688,096 infections and 6,322,514 deaths have been reported globally and 12,613,634 infections and 107,604 deaths in Spain [5].

The significant impact of COVID-19 is not only limited to the area of health but has also conditioned different domains, for example: (a) it has tested the economic resilience of countries [6], (b) it has conditioned the educational system [7], (c) and it has had an impact on a social level and on people's lifestyle [8], among many other factors.

As a result of this impact, many professionals and institutions have worked to make technologies available which can facilitate the prevention, control, and research of this disease in order to try to put an end to the health emergency. One of these institutions is the National Institute of Statistics (INE) of Spain which has collected statistical information of interest for the analysis of the impact of the COVID-19 crisis [9] through indicators, graphs, reports that are available through its official API [10].

In regard to technologies, specifically the field of web applications, this work has mostly been translated into the development of COVID-19 monitoring applications, since their implementation is one of the most effective responses for the management, control and information on the pandemic [11], as well as the fact that its development does not involve too much complexity.

The aim of this Erasmus Project is to study and analyze the current situation of COVID-19 with the objective of creating a web application that monitors all the data related to this disease and its impact in Spain. The project will be explained in more detail below.

## ERASMUS PROJECT

This section is intended to provide a more detailed approach to the project and its development.

The application to be developed monitors all data of interest on COVID-19 in Spain and in the 19 autonomous communities that constitute the country. Following the classification made by the INE [9], three categories have been defined for the data: (1) economic (2) sociodemographic and (3) health.

The data used by the application is obtained partly through the INE. As a result, only the data taken into account was that which (1) is accessible through the INE’s API, (2) is present in the year at the beginning of the pandemic and (3) is available for each or most autonomous communities, or simply Spain. Three categories of data have been considered from the web:

* ***Sociodemographic data***: this data allows for the study of the population in terms of its demographic characteristics, and includes aspects related to the use of technologies, work activity and demographic features [9].
* ***Economic Data***: short-term indicators that allow the analysis of the evolution and general aspects of the Spanish economy. This is data considered relevant by the European Statistical Office [12].
* ***Health and mortality data***: it includes death data caused by any reason [9].

In addition, some of the metrics most used by other COVID-19 monitoring applications are used, i.e., cases, deaths, hospitalizations, and vaccinations. This data has been extracted from the Data Against COVID-19 project website [13] and manually stored into a Github repository from where the data is read.

## 1.3. APPROACH

This section describes the different methodologies used for the development of the project.

### 1.3.1 WORKING METHODOLOGY

#### 1.3.1.1 SCRUM

The Scrum methodology [14] has been used for the development of the application. Scrum is an agile development methodology that focuses on providing the best possible business value while minimizing development time.

An agile methodology is based on iterative, incremental, and adaptive development. These characteristics allow greater flexibility and customer satisfaction as they receive constant feedback on the progress of the project.

Scrum defines a set of roles for the team, artifacts, and a workflow to follow. Regarding the set of roles, three roles are detailed below:

* ***Product Owner***: this role is responsible for identifying the necessary product requirements and assigns them a priority in the Product Backlog. The Product Owner and the customer can be the same person, as is the case of the supervisor of this project.
* ***Scrum Master***: is responsible for ensuring that the team makes use of the Scrum theory, practices, rules, and values. Since the Scrum team is formed by a single developer, this figure is not necessary for this project.
* ***Scrum Team***: In each sprint, they develop and implement a certain number of user stories agreed upon with the Product Owner in the Sprint Planning. In the case of this project the team is composed of one person.

In terms of artifacts, the most prominent are:

* ***Product Backlog***: it is a dynamic list with the system requirements, which is adapted throughout the life cycle to the needs of the system. At the beginning of the project, a list containing the application requirements was elaborated, and it has been modified at each meeting with the Product Owner. For the requirements, their duration and effort were estimated based on the Product Owner's information and the Scrum team's knowledge.
* ***Sprint Backlog***: the set of tasks that the Scrum Team agrees to solve at the end of the Sprint.
* ***Burn down chart***: It is a graph that shows the evolution of the team's work during a Sprint, and in which the ideal and the real evolution are contrasted.
* ***Deliverables or increments***: These are the result of each sprint.

Finally, Scrum defines a workflow that consists of:

* ***Sprint***: these are increments (agile methodology) with a time period of two weeks in the case of the project. The Sprints begin with the planning in which it is decided which elements of the Product Backlog are going to be carried out, and end with the Sprint review, in which the deliverable is shown to the Product Owner to check that the objectives are being fulfilled.
* ***Daily meeting or daily scrum*** is a short meeting that is held daily throughout a Sprint. Its purpose is to keep all team members, Scrum Master, and Product Owner up to date on the status of the project. As with the scrum Master, this event has not been necessary for the development of the project.

#### 1.3.1.2 GANTT DIAGRAM

Figure 1.1 shows the Gantt chart of the project, showing the expected time commitment for the different tasks or activities.

The list of tasks is detailed below:

* ***Project Research***: first contact of the project, in which research has been carried out on how to approach the project and with which technologies it should be developed.
* ***Study of technologies***: time dedicated to learning how to use the technologies that have been selected for the development of the application.
* ***Initial design***: the initial architectures and diagrams of the application have been analyzed and developed. The final result is detailed in Chapter 3.
* ***Sprint 1***: Focused on frontend and backend integration, data extraction from the backend to INE and Github, and the frontend main menu.
* ***Sprint 2***: Focused on implementing all the views. As a result, the first delivery has been delivered to the Product Owner that has determined that it was necessary to add more functionalities to the system.
* ***Sprint 3***: Focused on implementing the changes proposed in the previous sprint.
* ***Sprint 4:*** In the last sprint the project presentation has been worked on and some of the changes proposed in Sprint 2 have been finalized. The result has been the final delivery, which has been validated by the customer and Product Owner.

Writing the report: although the process has been documented in parallel during the whole development, the last 15 days have been dedicated to the elaboration of the final report and its correction.

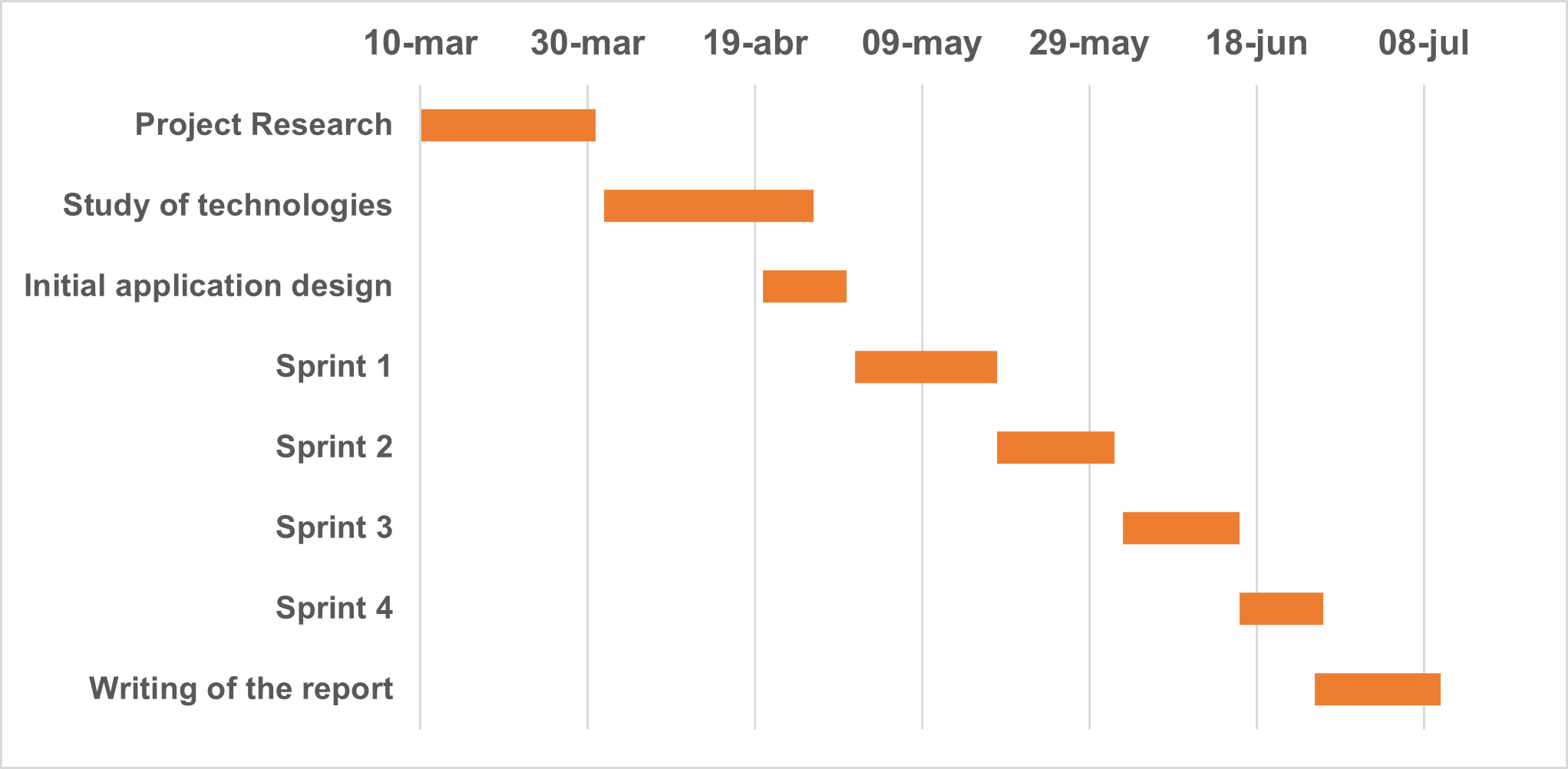


Figure 1.1 Gantt diagram of the project

### 1.3.2 TECHNOLOGY METHODOLOGY

Numerous frontend and backend technologies have been used in the development of the application.

In the frontend, the Vue.js framework has been used for the basic structure of the application and the Vue Router extension to achieve a Single Page Application (SPAs). In addition, the following libraries have been used: (1) *Chart.js* which is used to generate interactive charts, (2) the *chartjs-plugin-zoom* plugin which extends the functionality of *Chart. js* to allow for zooming in the charts, (3) Bootstrap (5th version) to make a responsive application and work with the presentation of the application, (4) Leaflet for the creation of the interactive map of Spain and (5) the jstoxml package to transform JSON objects to XML and allow one of the downloads that can be made of the data to be in XML.

To download these software’s, a package installer has been used since it facilitates and simplifies the installation of the package and its dependencies, in the case of the frontend npm (Node Project Manager) has been used.

On the other hand, for the backend structure the Flask framework has been used, and for the application logic the following libraries have been used: (1) Pandas, for the analysis of all the data and for the extraction of the data from the Github repository, (2) SciPy to calculate the Pearson and Spearman correlation of the data, (3) requests to manage the requests and (4) datetime to work with the date formats. In the case of the backend, since the language used is Python, the Pip package management system is used to install the packages.

## 1.4. PROJECT OBJECTIVES

This section specifies the general objective together with a list of specific objectives to be achieved with the development of this project.

The overall objective is to develop from scratch, based on the relevant and reliable metrics and information on COVID-19, a monitoring application that any user can use to visualize and analyze the evolution of the pandemic in a simple and fast way.

The specific objectives are as follows:

* Acquire knowledge and make use of popular frameworks to develop the application.
* Acquire knowledge about the operation and use of APIs.
* Make use of an agile methodology for project development
* Make use of Python and JavaScript libraries to use them in the development of the application.

## 1.5. DOCUMENT STRUCTURE

This work is divided as follows: Chapter 2 addresses the state of the art, in which Covid-19 Technology Trends regarding COVID dashboard, the technologies chosen for the project and some basic concepts of web development are presented and explained. Chapter 3 includes the design and analysis of the application. It is divided into a section in which the domain model and the application requirements are exposed, and another section in which the architecture is detailed following the 4+1 architectural view model. Chapter 4 details the implementation of the project. This chapter is composed of: (1) a section dedicated to the explanation of how the code has been organized and the main parts of the code, (2) another section in which the result is shown through screenshots and (3) a section dedicated to the testing of the application. Finally, Chapter 4 concludes the report with a conclusion. The chapter also includes a section on the limitations and future work of the project, and a reflection on the Erasmus+ mobility.

# STATE OF THE ART

## 2.1 COVID-19 TECHNOLOGY TRENDS REGARDING COVID-19 DASHBOARD

Since the beginning of the pandemic, COVID-19 dashboards have been developed continuously. These dynamic applications are characterized by displaying, in a visual form that facilitates analysis, information related to the disease in real time, or in a date range.

Although there is no particular approach when developing these applications, there are some factors that improve their usefulness, such as (1) the orientation of the content to the target audience [15], (2) that the information is not reduced only to show the most typical indicators about COVID-19, such as those described in the article of The COVID Tracking Project [17], but rather that the aim is to provide a complete context of the pandemic with indicators such as those proposed by the WHO [18], (3) the veracity and accuracy of the data, whether obtained from the competent authorities of the country concerned or from recognized scientific institutions [19].

In the following sections, we will analyze in detail the applications that are currently in use. Two of them are some of the best-known ones, and others that are less known, are similar to this project since they are focused on Spain. From there we will also explain the differences of this project with respect to the rest.

### 2.1.1. JOHNS HOPKINS UNIVERSITY DASHBOARD

The Johns Hopkins University (JHU) dashboard [16,21], originally developed by associate professor Lauren Gardner and her team at the Center for Systems Science Engineering at JHU, was launched in late January 2020 with the intention of developing a system that would display confirmed cases of COVID-19 in real time, so that researchers, authorities, and citizens could keep abreast of the outbreak.



Figure 2.1 JHU Covid Dashboard

As COVID-19 gained more visibility, the application incorporated data on more countries, and more metrics, forcing them to automate the data extraction process, which they obtain from their public Github repository.

The design and development have been carried out through the ArcGIS platform. This software manages to generate a dashboard that allows the user to visualize all the data in an interactive way, based on a location and on a single screen [20], creating a user-friendly interface (see Figure 2.1).

In the interface, data can be displayed individually or with graphs on confirmed cases, deaths, and vaccinations against COVID-19, globally and regionally in some countries. In addition, there is the possibility to look in detail at one of the available regions.

As a result, JHU dashboard has become one of the most popular tools and has become a reference for projects of this style.

### 2.1.2 WHO CORONAVIRUS (COVID-19) DASHBOARD

The WHO COVID-19 dashboard [22], is a dashboard developed by the WHO itself that collects official data from each country on cases, deaths and vaccination and displays them as shown in Figure 2.2.

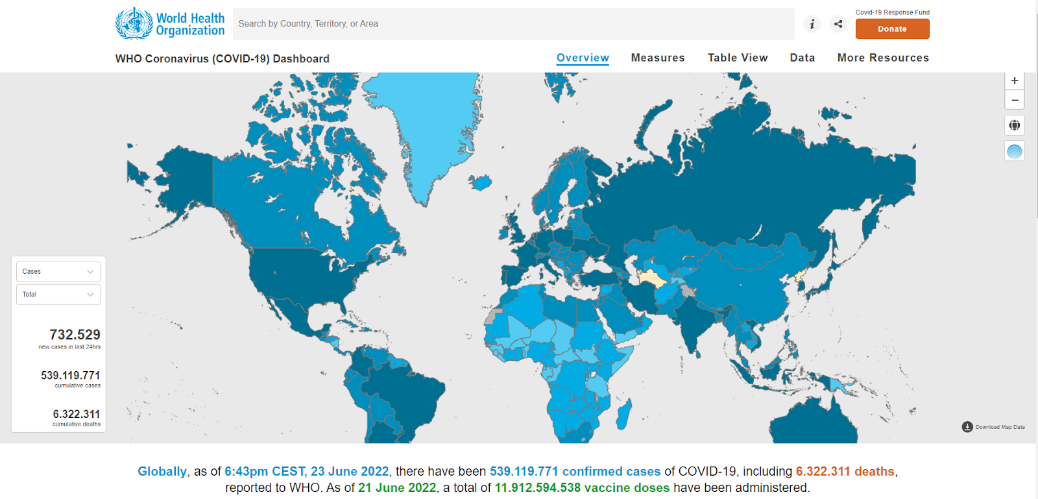


Figure 2.2 WHO Coronavirus (COVID-19) Dashboard

The design is intended to provide the information in a very simple, interactive, and accessible way to any user. In addition, it makes use of a choropleth map, whereby the different regions are shaded according to a value, such as confirmed cases for example.

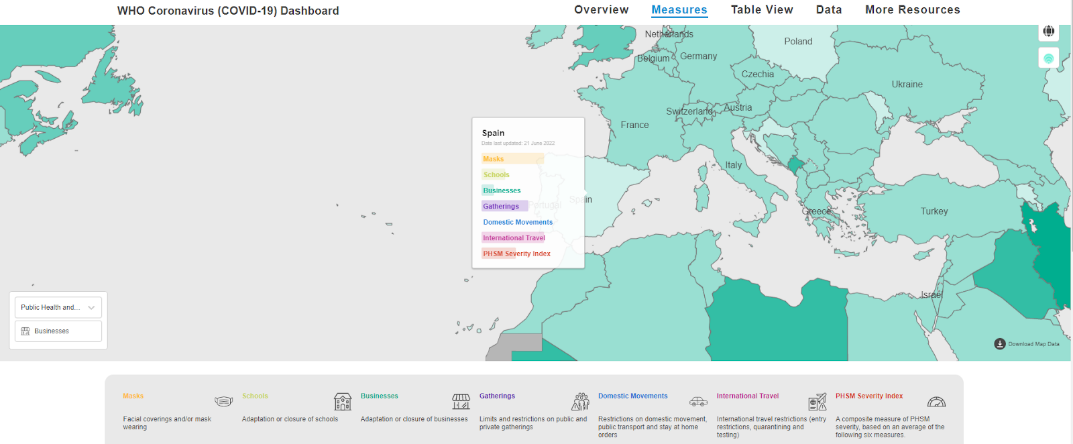


Figure 2.3 WHO Coronavirus (COVID-19) Dashboard. Measures view for Spain

A feature added by this dashboard, as shown in Figure 2.3, is the possibility of visualizing public health and epidemiological considerations for each region, as described in one of their articles [18].

### 2.1.3 RTVE GRAPHS, MAPS AND DATA

The "Corporación de Radio y Televisión Española" RTVE [23], through its Audiovisual Innovation Department, has developed one of the most effective tools for dissemination of information on the coronavirus in Spain [24].

Unlike other dashboards, RTVE provides a series of interactive maps, graphs, tables, analysis, and data, showing information on cases, deaths, care capacity (as shown in Figure 5), vaccinations and tests performed in Spain at national level, by autonomous communities and by provinces, which is obtained from the Spanish Ministry of Health. It also includes a map of the different restrictions that have been implemented in the different municipalities and autonomous communities (Figure 2.4 & 2.5).

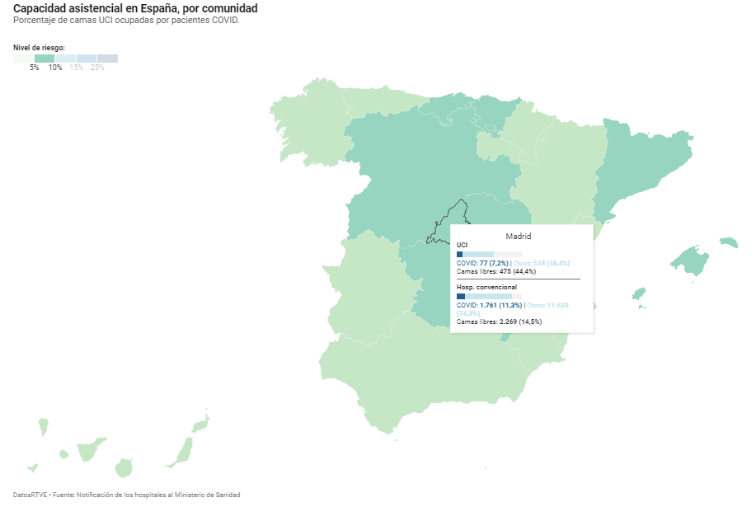


Figure 2.4 RTVE COVID-19. Health care capacity in Spain, by community

Although it may differ in the dashboard concept from the other applications presented in this chapter, the tool offers a detailed analysis that allows monitoring the COVID-19 situation in Spain, and even includes a simple dashboard of the world.

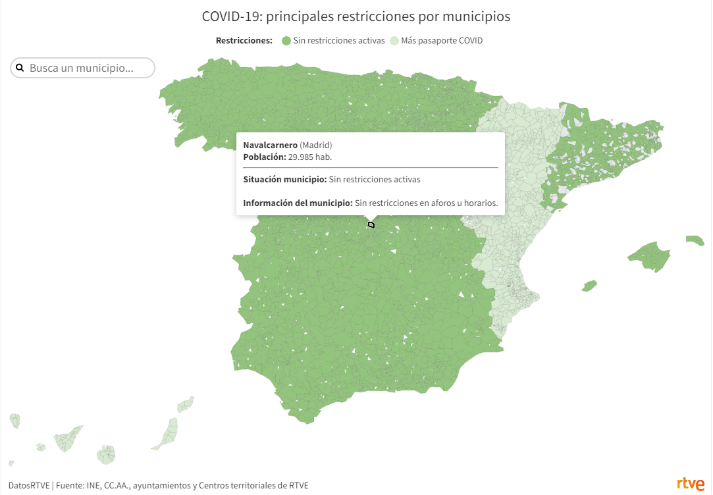


Figure 2.5 RTVE COVID-19: Main restrictions by Spanish municipality

### 2.1.4 DATAC: DATA AGAINST COVID-19

The Data Against COVID-19 (DaTAC) dashboard [13], is a collaborative project between the Pfizer Center, the University of Granada and the Junta de Andalucía aimed at scientific research.

The main aim of this project is to show the impact that COVID-19 has had on climate and air quality in Spain only. It includes all the COVID-19 metrics that the other dashboards defined in this chapter had for each Autonomous Community of the country. In addition, it includes data related to weather, air quality and demographics. These data are available for a certain range of dates and can be viewed from its interface, shown in Figure 2.6.

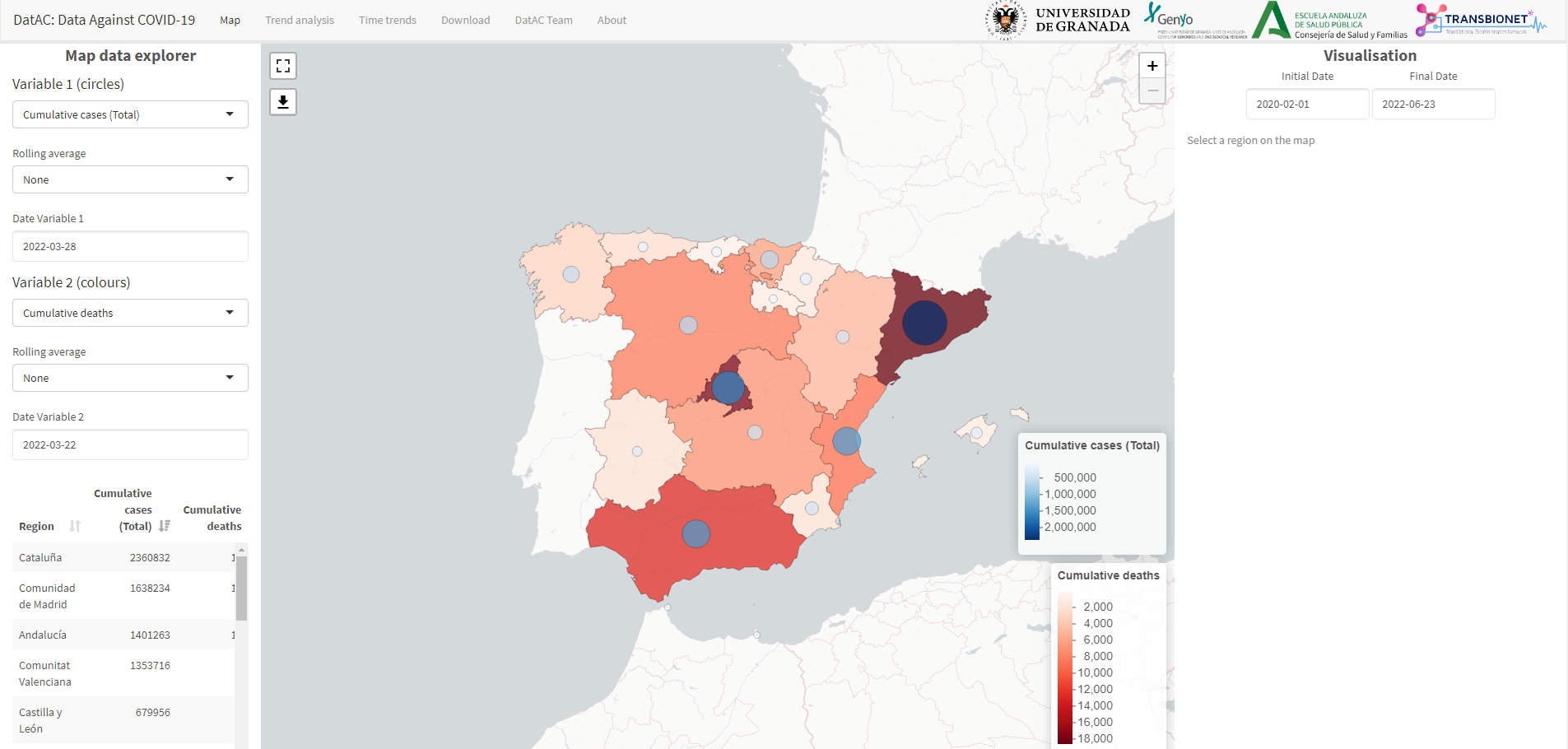


Figure 2.6 DatAc dashboard

The importance of this dashboard lies in the possibility of downloading its data, since most of the data related to the coronavirus has been downloaded and used in the Erasmus Project.

### 2.1.5 Erasmus Project against other dashboards

Nowadays, by surfing the Internet, you can find numerous dashboards with information not only worldwide, but focused only on Spain, that includes COVID-19 metrics at the level of autonomous communities and provinces, even at the level of cities and municipalities.

With this Erasmus project, it is not intended to develop another dashboard that contains the same information as another app, but to give a different approach by taking advantage of the data provided by the INE. Thus, this would give a greater context to the pandemic, providing the user with the tools to visualize and analyze the evolution of the pandemic, through all the relevant data of the coronavirus and its impact.

## 2.1 WEB APPLICATIONS

The world of web development has evolved exponentially giving rise to new technologies which allow the field of web applications to grow. One of the many definitions that exist for web application is: “Web applications refer to software programs accessed via Web browser over a network and developed using browser-supported languages (e.g., HTML, JavaScript, CSS). “[25].

The development of these web applications is divided into (1) frontend and (2) backend. The frontend is the presentation layer in charge of working the visual interface (GUI) that the user interacts with. It is oriented to markup language such as HTML, styling language such as CSS and programming language such as JavaScript (JS). The backend, on the other hand, is in charge of manipulating data and providing a way to communicate that data to the frontend [29,30].

In this section we will explore some of the technologies or tools that are part of the web development world and are used in this project.

### 2.2.1 FRAMEWORK & LIBRARIES

A framework is a set of tools, libraries and best practices that provides a blueprint for software development [26]. Using frameworks allows (1) speeding up the development process, (2) avoids duplicate code, (3) ensures good practices, (4) facilitates testing and debugging, and (5) ensures a more secure code [26,27]. As in application development, a differentiation is made between frameworks used for frontend and backend.

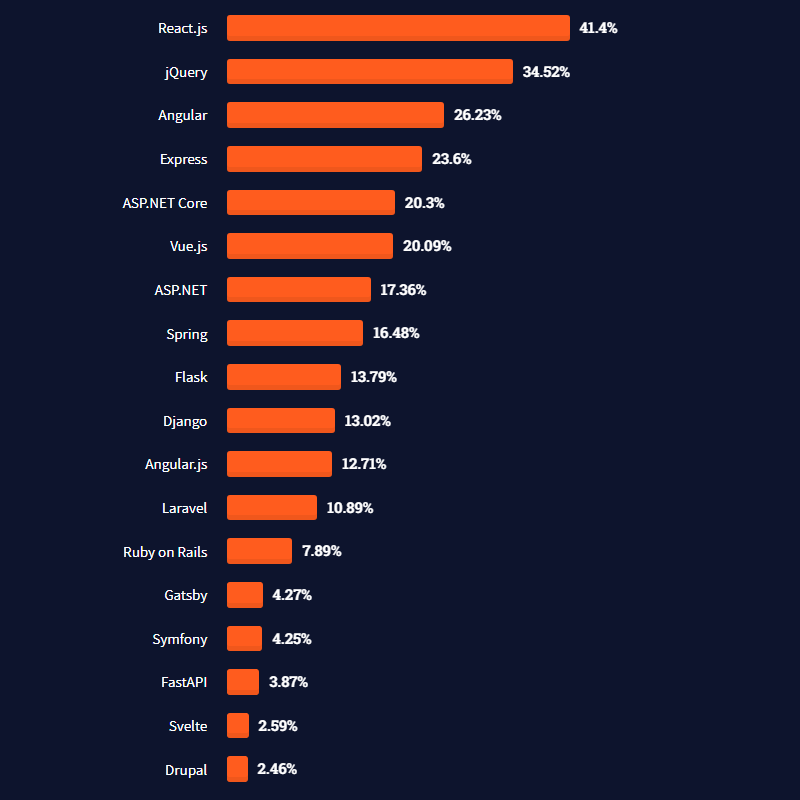


Figure 2.7. Stack Overflow Survey 2021. Most used popular frameworks and libraries amongst professional developers

On the other hand, libraries are external code fragments that extend the developer's code and are focused on solving a specific problem [27]. Their predefined functions and classes allow accelerating and simplifying software development. The main difference with respect to frameworks is that the library is oriented to a specific problem while the framework defines the structure and framework on which the application is developed [27,28].

Figure 2.7 shows the results of a survey conducted annually by the Q&A website Stack Overflow. This survey shows a list of the most popular frameworks and libraries used by professional developers [29].

#### 2.2.1.1 VUE.JS

Vue.js is a popular open-source JavaScript (JS) ES6 framework developed by Evan You, which allows the programmer to develop user interfaces in a simple way [32].

Figure 2.8 shows the architectural pattern of Vue.js; the Model-View-ViewModel (MVVM). The View displays the website content, the View Model, contains DOM listeners, data bindings and directives. Through the DOM listeners the state of the Model is updated and through the data-bindings and directives the View is updated. The Model contains the plain JS objects [36].



Figure 2.8. Vue.js MVVM Architectural pattern

This technology is composed of several fundamental features. First, Vue.js allows splitting the UI into independent and reusable pieces, called components. Generally, these components are built following the Single File Component (SCF) format that allows encapsulating the logic (JavaScript), the style (CSS) and the structure through templates (HTML) [32] in the same file. Each component instance goes through a lifecycle shown in Figure 2.9. This lifecycle consists of a series of methods accessible from the components, which are executed from the creation to the destruction of the component, and which allow to know when the component was created, added to the DOM, modified, or destroyed.

To access and make use of the components, since from version 3 of Vue.js, there are two ways: (1) Options API and (2) Composition API. With the Options API, the logic of the component is defined through options such as data, methods, or the different lifecycle hooks, while in the Composition API the logic is defined by importing API functions. [32]

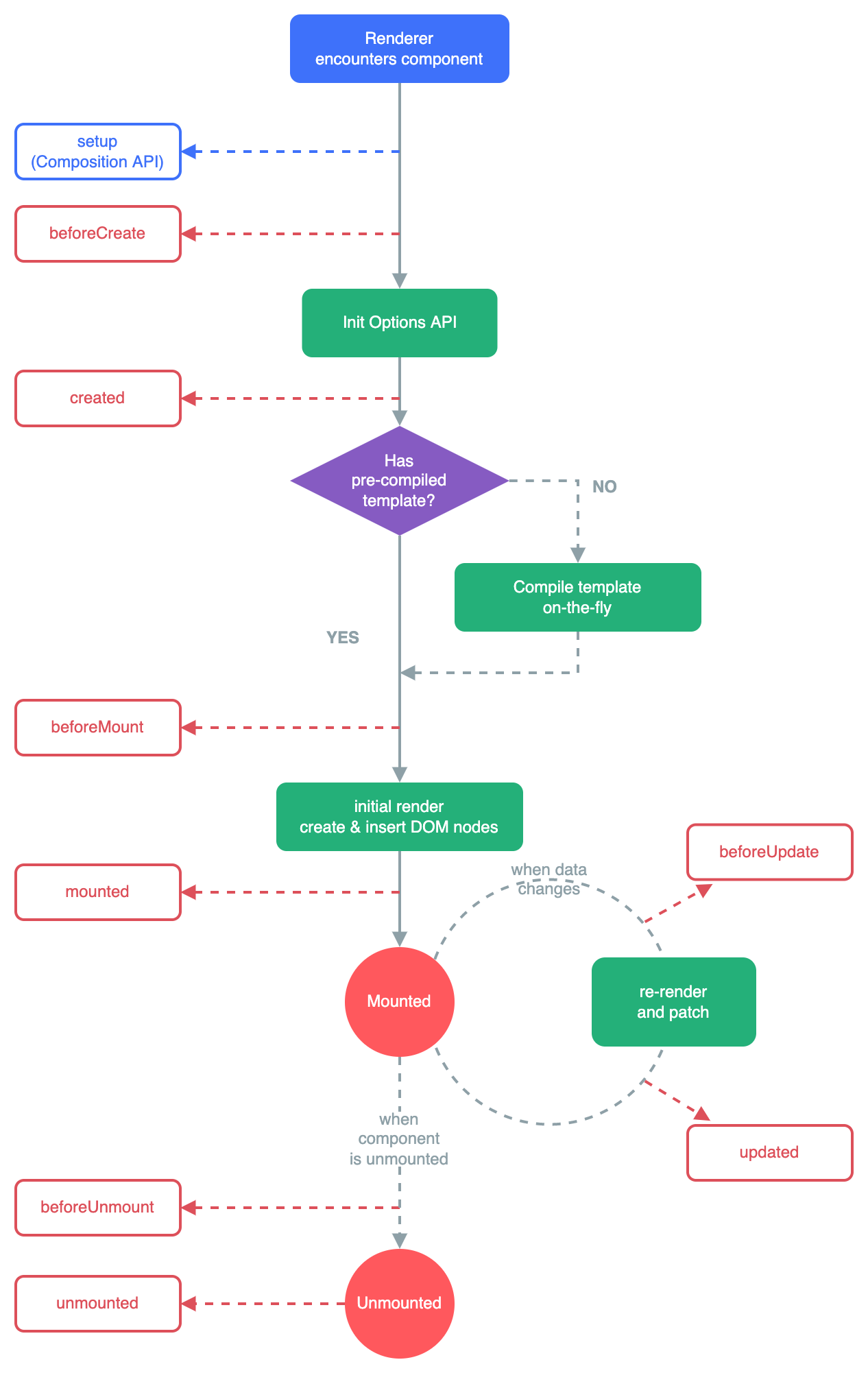


Figure 2.9 Vue.js lifecycle hooks

Secondly, it is a reactive framework. Reactivity is a programming paradigm oriented to the propagation of change, in varying values [33]. Applied to Vue.js it means that the Document Object Model (DOM) is updated based on every change in the component's JavaScript.

Thirdly, it makes use of declarative rendering, which means that it is result-oriented, because the user specifies what he/she wants to render from HTML, without having to specify how to do it [34].

Finally, Vue.js is a progressive framework, which means that it is designed in a flexible and incrementally adaptable way [32]. This flexibility has facilitated the community to develop numerous plugins to extend or create new features for this framework. One of the most important plugins is Vue-router that allows the development of Single Page Application (SPA). An SPA is a web application that fully loads all of the resources in initial request and then the components are updated depending on user interaction [35]. This approach prioritizes the user experience (UX) and a quick loading time.

The features described help Vue to become a (1) flexible, (2) simple, (3) reactive, (4) well-documented and (5) lightweight framework. On the other hand, since it is a recent framework, it does not have as much documentation and support as other frameworks like React or Angular. Furthermore, it is not recommended for complex or large-scale projects, due to the limited resources it offers. [37]

Vue.js was chosen as the frontend framework for the project, mainly because the application to be developed does not require much complexity and because of the good quality of the documentation of the framework that facilitates learning. In addition, the Options API has been used, since both APIS cover the fundamental concepts and functionalities, and also because the Options API has been used in previous versions, therefore there is more documentation about it.

#### 2.2.1.2 FLASK

Flask is a popular Python micro-framework developed by Armin Ronacher. It is considered "micro" because it aims to keep the core simple but extensible [38], which allows the developer to make use of extensions.

The main advantages of using Flask are its (1) scalability, the few dependencies it has and its simplicity offers the possibility of growing an application, (2) the flexibility and ease due to its simplicity and minimalism, (3) lightweight, (4) the documentation available on the framework [39] and (5) it is compatibility with the Web Server Gateway Interface (WSGI) , which makes it compatible with a large amount of servers [39].

However, it has a few disadvantages. Some of them include: (1) the absence of tools, it is very limited in simplicity, it requires adding extensions or external libraries, (2) there is no database support [40].

Flask has been used for the development of the project mainly for three reasons: (1) it is easy to get started because its simplicity, since the project does not require a complex backend, (2) it is developed in Python so it can make use of libraries such as Pandas for data analysis, and (3) it does not include a database abstraction layer, which makes it a lighter framework and more appropriate for this project that does not have a database implemented.

### 2.2.2 BROWSER SUPPORTED LANGUAGES

Within web applications, three categories of languages can be distinguished:

First, programming languages. Within this category, JavaScript stands out; it is an interpreted programming language mainly oriented towards the development of web applications and conforming to the ECMAScript (ES) specification. The standard ensures the interoperability of web pages in different web browsers [41].

Secondly, markup languages, which establish the structure of a text through tags or markup. Since the beginning of the World Wide Web, the most widespread is HTML (HyperText Markup Language), although there are also other languages such as XML or XHTML. HTML is currently in its fifth version (HTML5). With each version, new tags are added that define the structure and even the styles of documents [42].

Finally, there are style sheet languages, which define the presentation of a structured document that has been generated by a markup language. The most used language for HTML documents is Cascading Style Sheets (CSS). CSS describes how the structured element should be rendered on the screen [43]. CSS is currently in its version 3 (CSS3).

### 2.2.6 APPLICATION PROGRAM INTERFACE

An Application Program Interface (API) is a set of definitions and protocols used to allow two or more computer programs to communicate with each other without needing to know how each is implemented. APIS can operate in four different ways: (1) SOAP API, (2) RPC API, (3) WebSocket API and (4) Representational state transfer (REST) API, the latter being the most popular and flexible on the web [44].

Since only the REST API is used in the project, this concept will be explored in detail. A REST API defines a set of functions such as GET, POST, PUT, DELETE that the client uses to access data from the server. The communication between client and server is done through Hypertext Text Transfer Protocol (HTTP) request and responses [44].

In cases where this HTTP communication requires data transfer, two types of notations are mostly used: (1) JavaScript Object Notation and (2) eXtensible Markup Language (XML). Figure 2.10 shows an example of how the same object is represented using the different notations [45].

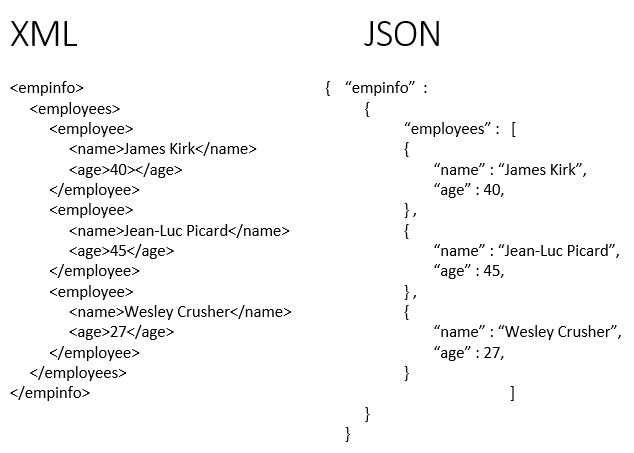


Figure 2.10 XML (left) vs JSON (right)

### 2.2.3 OTHER TECHNOLOGIES

This section describes other technologies that will be used: (1) integrated development environment (IDE) and (2) source control tool, (3) database. First, for the IDE we have used Visual Studio Code, an editor developed by Microsoft, focused on code development and allowing the use of plugins that facilitate the reading and development of the code.

Git and Github have been used as source control tools. Git is a distributed version control system, which means that all developers working on the same code have a local version of the complete history of changes made to each file. On the other hand, GitHub is a cloud repository storage platform that allows you to manage Git repositories.

Finally, the application does not make use of any database integrated to the backend, because this functionality is simulated by making use of Github. The online repository allows to visualize all unprocessed versions of stored values from a domain (*raw.githubusercontent.com*) [46], so files such as comma separated values (csv) files can be read by libraries such as Pandas. This greatly simplifies the process since many dashboards update and publish COVID-19 data into their Github repository. Additionally, in the case of the project, the data needed is a historical COVID-19 metrics for Spain, which do not occupy a considerable volume of memory (approximately less than 500KB), and do not need to be modified by the user, so it is not necessary to implement a database.

# ANALYSIS & DESIGN

Once the project has been defined and the state of the art has been studied, the analysis and design of the application will be carried out.

## 3.1 DOMAIN MODEL & REQUIREMENTS

The domain model describes the most important concepts of the system from a business perspective to check its structure and dynamics, and to extract the requirements. Figure 3.1 shows the domain model defined for the application.

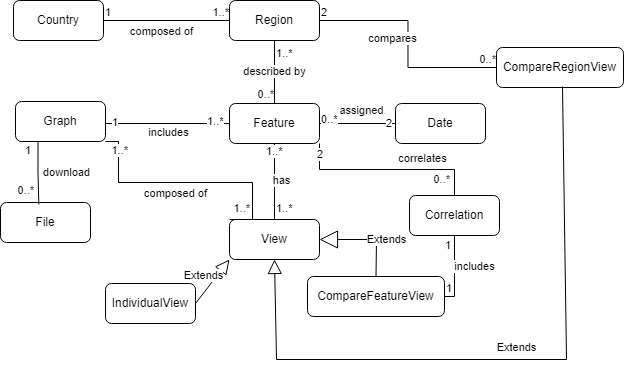


Figure 3.1 Domain model of the application

A country divided into regions, in the case of the project it is Spain, which is formed by 19 autonomous communities. For each community, characteristics will be extracted that can be classified as being directly related to COVID-19 or not. The extraction of these characteristics will be determined by two dates, one at the start and the other at the end, which will form the range in which the information will be extracted for the different views.

In the application, three views have been determined; one view (***IndividualView***) which is composed of all the features, one (***CompareFeatureView***) to compare the different features from a single graph and which makes it possible to calculate the correlation between two features, and the third view (***CompareRegionView***) to observe a feature in two regions.

All the display of characteristics is done through the graphics. These graphics allow ~~to~~ download them as image or data files.

Before defining the requirements, the actors must be identified, in this case there is only one, which is the user. The user is any person who uses the application through a web browser, and there is only one type of user.

The requirements that have been defined for the system are detailed below:

* **R1. It should work on different screen sizes.**

The application must be able to display the different views and functionalities for the different screen sizes.

* **R2. It must have a user-friendly interface.**

The application shall have a user interface that is intuitive and easy to understand.

* **R3. Download the displayed graphic.**

The user will be able to download each graphic displayed on the screen in *png* or *jpeg* format.

* **R4. Download chart data.**

The user will be able to download the data used in the chart in *csv*, *json* and *xml* format.

* **R5. View data for a region.**

The user will be able to view individually all COVID-19 metrics, sociodemographic, health and economic data for a region of Spain.

* **R6. Compare the data of a region**

The user will be able to compare COVID-19 metrics, sociodemographic, health and economic data for a region of Spain.

* **R7. Calculate the correlation of data from the same region.**

The user will be able to calculate the correlation of two different features for the same region.

* **R8. Compare data in different regions.**

The user will be able to compare the same feature between two different regions.

* **R9. Select the region using a dynamic map.**

The user can dynamically select from a map the region from which he wants to get data.

* **R10. Dynamic elements can be zoomed in and out.**

The user can zoom in and out the size of the dynamic map and the different graphs, to have a better visualization of the elements.

* **R11. Display the main COVID-19 metrics for a region.**

The user will be able to see a summary of the main metrics for a region. This summary is composed of daily and cumulative deaths, daily and cumulative confirmed cases, the number of complete vaccination schedules, the number of vaccines administered in the region and the percentage of the population who have been fully vaccinated (2 doses).

* **R12. Specify a date range in which the data is delimited.**

The user can specify a date range in which he/she wants the data to be displayed. In case no date is selected, there will always be two default dates chosen (the start date is the beginning of the COVID-19 pandemic, and the end date is the current day).

## 3.2 DESIGN

For the design of the application architecture, the Kruchten view model has been used as a guideline therefore, a Logic, Implementation, Process, Deployment and Scenario view have been created. To represent the views, the Unified Modeling Language (UML) has been used, which is a standardized modeling language used, among other things, for the design of the different diagrams that make up the views.

### 3.2.1 LOGICAL VIEW

The logical view captures the logic of the entities and what the system should provide in terms of services to its end users [47]. The class diagram has been used to represent this view.

Since Vue.js is used to develop the frontend logic, classes have been replaced by components. In addition, props, which are the variables that allow passing values from the parent component to the child component, are included as an attribute of the child class. In the case of emit, which, unlike props, are variables used to pass values from the child component to the parent component, are also included as an attribute of the child class. In addition, the computed variables, are included as an attribute of the component.

In the operations section of the class, the methods that are called in the different life hooks of Vue.js, and the methods used in the Vue.js watchers, as well as the rest of the functions are included.

To describe all these behaviors, the UML stereotype is used to extend the functionality of this language by adding it as a prefix to the attributes and operations.

Regarding associations, in cases where the component is included within the template of another, the composition association is used. The inheritance relationship is not used, because, although they are considered parent and child components, no method is inherited and when deleting the component that includes the other, its instance is deleted. The inheritance relationship is only used in cases where there is mixing with another component, since Vue mixing simulates this behavior, otherwise the simple association is used.

Figure 3.2 shows the class diagram of the system, with the attributes and methods necessary for a basic understanding of the logical operation of the system.

The root of the whole system is the App component, but as it does not include any method or attributes, it has not been included in the diagram. From there the ***MainMenu*** component is accessed, which displays the menu that will allow the user to access all the functionalities. It includes the selection of dates (***MenuDates***), the map (***SpainMap***) and the buttons (***MenuButtons***).

The ***SpainMap*** component is in charge of the logic of the leaflet map, where the geoJSON layer for Spain and the autonomous communities is loaded to allow the selection of the regions. In addition, in the component template, the most recent COVID-19 statistics are displayed in the ***CovidStats*** component, which is responsible for retrieving and updating the data

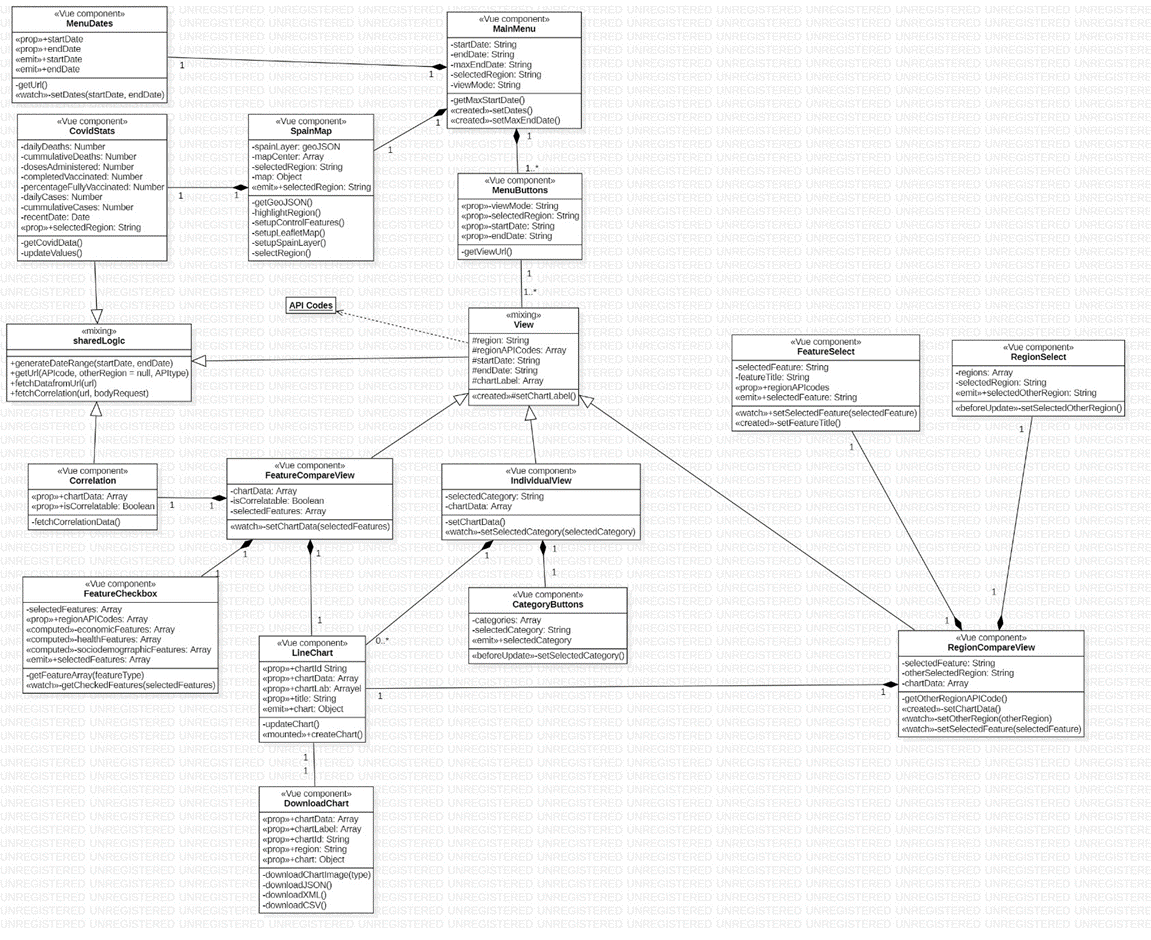


Figure 3.2 System class diagram

The menu buttons (***MenuButtons***) serve as a link to access the different views. The main purpose of this component is to generate the url so that the views can access date and region information from the url.

For the views, there is a parent component (***View***) that contains the common logic of the three views. It provides the logic to calculate the labels of the graphs, in which a label is generated with all the days within the date range. In addition, it makes use of the ***API\_Codes*** object that contains all the necessary information to make a request to the backend with the codes needed by the INE API and the GitHub data.

Three view components are inherited from the parent component. First, the ***IndividualView*** component, which displays all line charts classified by categories (***CategoryButtons***).

Secondly, there is the ***FeatureCompareView*** component which displays a line chart showing all the features that have been selected from the ***FeatureCheckboxView***. It is also associated to the Correlation component to be able to calculate the correlation between features that fulfill the conditions (when *isCorrelatable* is true).

Third, the ***RegionCompareView*** component implements the logic to compare the selected feature (***FeatureSelect***) between the region selected in the menu and the one selected from in ***RegionSelect*** component.

The display of the characteristics of the different views is performed in the ***LineChart*** component, which receives the data from the views to create the dynamic chart and keep it updated in case of changes. Each chart has associated download options (***DownloadChart***), which allows downloading the chart itself or its data.

Finally, a class has been included for the shared logic (***sharedLogic***) that some of the components have. It contains the methods for requests to the backend, and for the generation of the date range used in the chart labels.

### 3.2.2 DEVELOPMENT VIEW

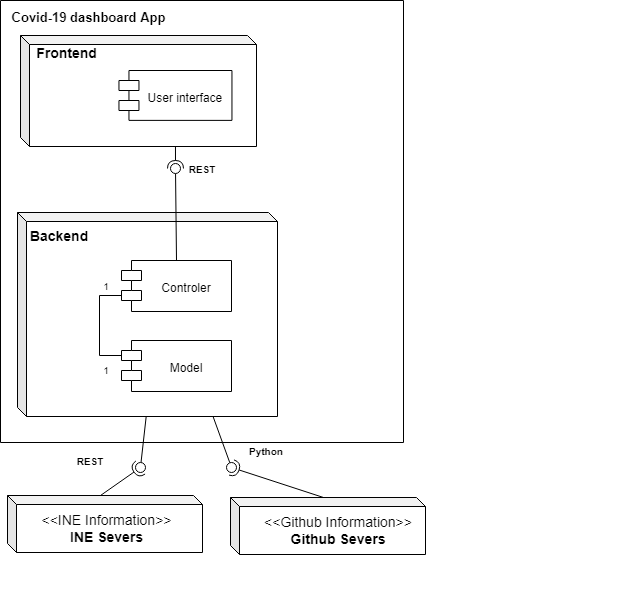


Figure 3.3 System component diagram

The development viewpoint focuses on the organization of the software modules in the software-development environment where the software is packaged in subsystems that can be developed by one or more developers [47]. For this view, the Component diagram has been used as shown in Figure 3.3.

Within the application there is a module for the frontend and another for the backend. The frontend contains the package associated with the logic of the user interface components and their design. This module communicates with the backend through the REST protocol.

In the backend these requests are received by the Controller, which is composed of several endpoints that leave the logic up to the Model. The Model performs three tasks: (1) it obtains data from external services to the application, communicating with the INE interface through REST and with Github using the Pandas Python library, (2) it transforms the data to a format suitable for use in the frontend and (3) it calculates the correlation from the data. Each time any of the operations is performed, the Controller is notified to update the application frontend.

### 3.2.3 PROCESS VIEW

The process view focuses on the software processes and how they communicate with each other [47]. The sequence diagram has been used for this view, which will be described module by module to facilitate the understanding.

Figure 3.4. Frontend sequence diagram

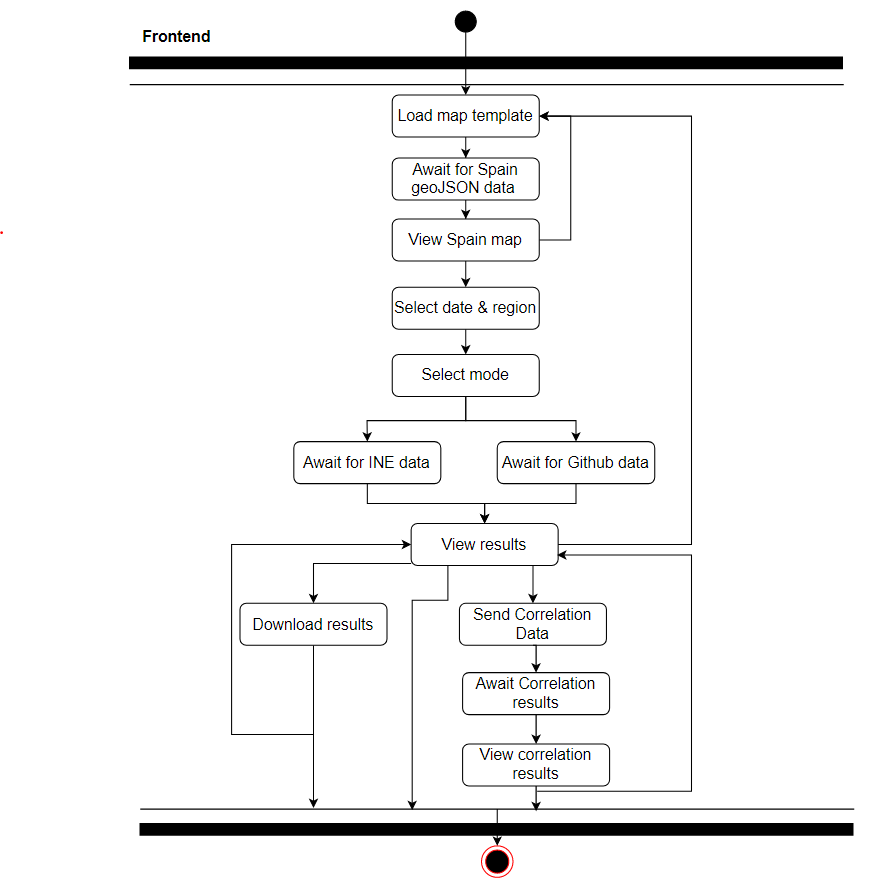


Figure 3.4. shows the frontend sequence diagram. There is only one main process, that starts with the loading of the Leaftlet map template. Before selecting a region, the app must wait for the perimeter data of the regions to arrive from the backend to be loaded on the map. Then, the mode can be selected (the view that the user wants to see), and depending on the features that have been requested, a request will be made to the backend to get the data from the INE and the Github repository, in order to visualize them.

The process can end there, or in the case that the correlation is to be calculated, it sends the data of the two features to be analyzed to the backend and waits for the backend to respond with the result.

There is also the possibility of downloading the results. This functionality is done from the frontend and not from the backend because the data exchange sent from the backend is done with JSON, so no file can be sent directly. Therefore, the transformation of the JSON to the different types of files takes place in the frontend.

On the backend side, as shown in Figure 3.5, there are four parallel processes. There is one that waits for the geoJSON data to be requested, extracts it from the file system, and sends it to the frontend. This data does not require any kind of transformation.

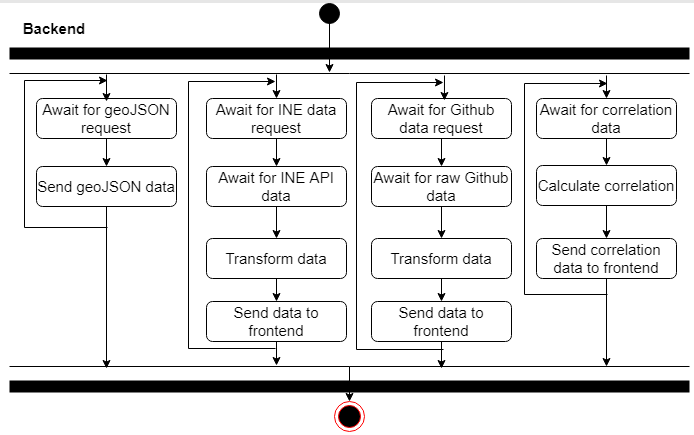


Figure 3.5 Backend Sequence diagram

There are two other similar processes dedicated to the extraction of data from the INE and the Github repository. These processes request the data from the servers and once they receive them, they are modified so that they do not have to be transformed in the frontend.

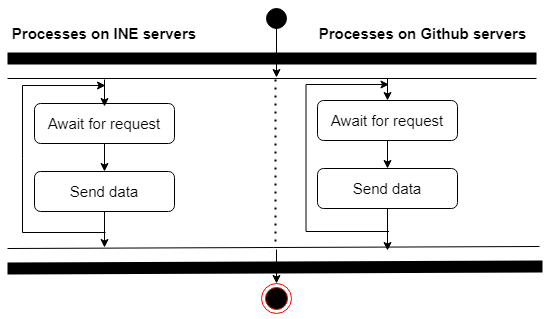


Figure 3.6. INE & Github servers sequence diagram

Finally, there is a process for correlation, which receives the data, performs the calculations to obtain the results and sends them to the frontend modified so that they do not have to be transformed.

The last processes described are those that occur on the external servers. Their operation is reduced to waiting and sending the data requested by the backend. The functioning of these processes can be seen in figure 3.6.

### 3.2.4 PHYSICAL VIEW

The physical view focuses on the topology and the connection between the elements of the other views at a physical layer level. The diagram shown in Figure 3.7 has been used to represent this view.

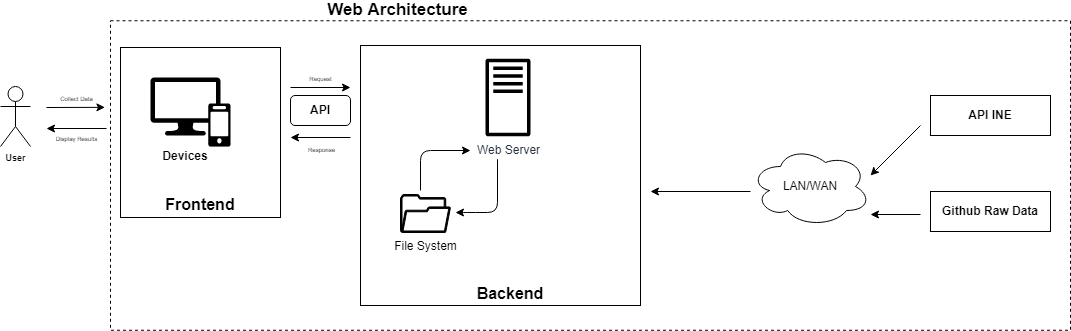


Figure 3.7 Physical view system

The user accesses the frontend through an electronic device with a browser. The results requested by the user are displayed on the interface. Through the REST API the frontend accesses the backend, and from the backend the INE API and Github are accessed through the network.

### 3.2.5 SCENARIOS

This view makes use of use cases to describe the interactions between the elements of the four previous views. It is considered the fifth view due to its redundancy regarding the other views. The use case diagram is used for this view.

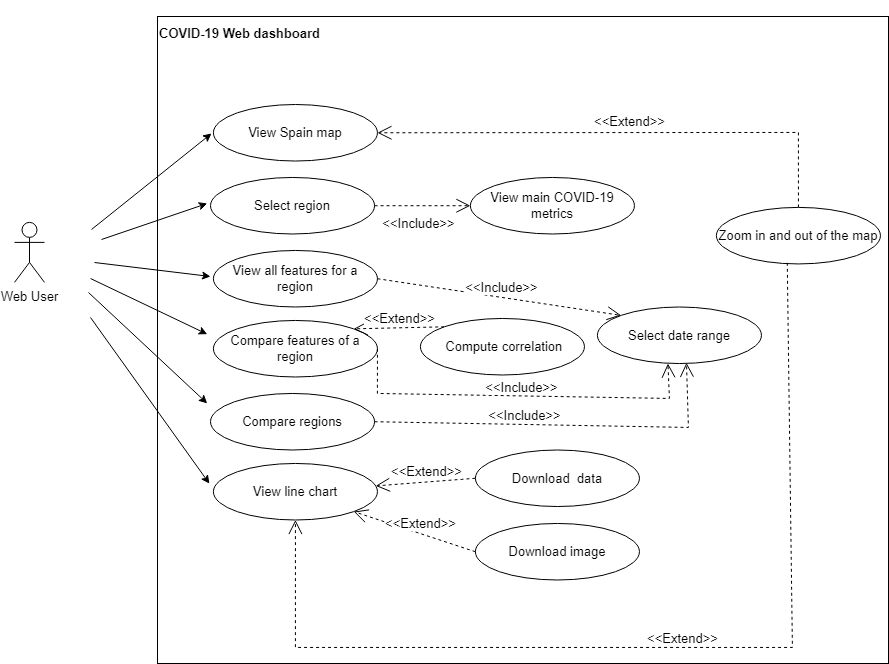


Figure 3.8 Use case diagram of the system

Figure 3.8 shows the diagram which depicts all the functional requirements (from R3 to R12) elicited in section 3.1 for the actor User, and the interaction between them.

# IMPLEMENTATION

Once the application has been analyzed and designed, its implementation is carried out.

## 4.1 CODE STRUCTURE

For the development of the application, several approaches can be used, which are detailed in an article [48]. For the project, it has been decided to employ the SPA approach, because this is more conducive for compliance with requirement R2 of section 3.1. Therefore, the frontend development with Vue.js will be completely independent from the backend [48].

Throughout the section, the structuring of the code by folders and the main parts of the code will be explained.

### 4.1.1 FRONTEND

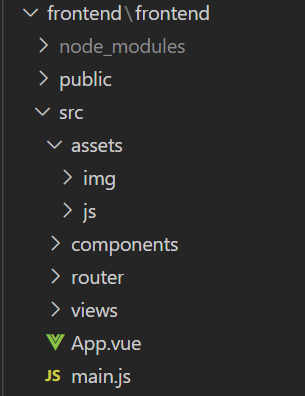


Figure 4.1 Frontend file structure

As shown in Figure 4.1 the frontend of the application is organized as follows:

* **Directory *node\_modules***:place where all npm packages and dependencies are installed.
* **Directory *public****:* contains the index.html file in which the root component App.vue is included.
* **Directory src:** is the main directory where the frontend logic is coded. It is divided in several subdirectories.
* **Subdirectory *assets*:** Figure 4.2 displays the breakdown of this folder. The *img* subfolder contains the images used in the application, and the *js* folder contains the JavaScript files common to the components. The *global.js* file includes the global variables for the whole project, in this case, the object that contains all the API's codes, and the *sharedLogic.js* file that includes the shared logic of the components.

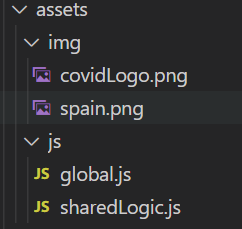


Figure 4.2 Assest folder structure

* **Directory *components:*** includes all Vue components that do not represent any view. Figure 4.3 shows the decomposition of this folder.

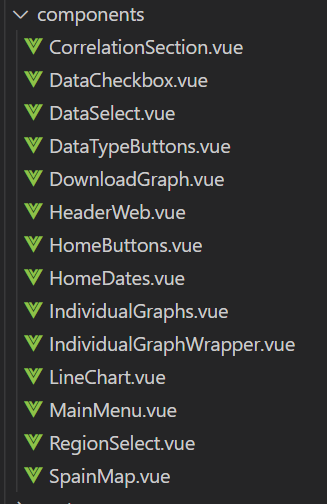


Figure 4.3 Components folder structure

* **Directory *router:***is the folder generated as a result of using the Vue router plugin. It only includes the index.js file that contains the routes of the application and its configuration.
* **Directory *views****:* as shown in figure 4.4, the folder contains all the Vue components that make up the application's views.

Text

Description automatically generated

* **Files *App.vue y main.js*:** the *App.vue* component is the root of the project that is included inside the main html template, and the *main.js* file includes the *App.vue* logic and the Vue plugins dependencies that have been used for the project (Bootstrap and Vue router).

Regarding code development, the main parts are detailed below.

For the downloading of data files, the functions to generate the *csv* and *json* have been done without using libraries because the data to be sent to the chart are not in a format that can be worked with. From these objects all types of files can be generated, and even more types of extensions could be added using other libraries.

The *dataJSON()* function of the DownloadGraph.vue component is used to generate the JSON object with a format as detailed in Figure 4.5.

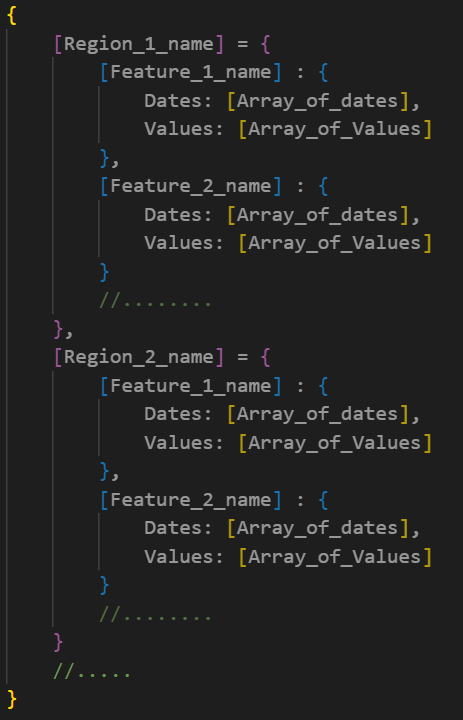


Figure 4.5 Intended JSON format format

The arrays generated in Values and Dates have the same length, so that each date has an item assigned by index within the other array. In addition, from this object the xml file can be generated.

To generate the csv, the *dataCsv()* function of the same component is used to generate a String with the format shown in figure 4.6.

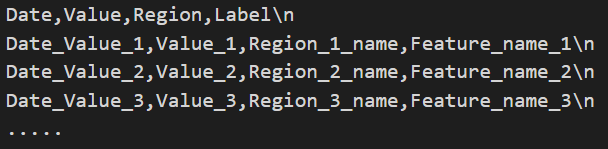


Figure 4.6 Intended csv format

It is intended to simulate a table, with the first row for the names of the columns, and the rest, the values taken by the columns.

For file downloads, the process is detailed in Figure 4.7. A link is created, in which the href attribute includes the type of file and the object to be downloaded encoded. For images, the href is replaced by the code shown in figure 4.8, by assigning the name of the image format to the *type* variable.

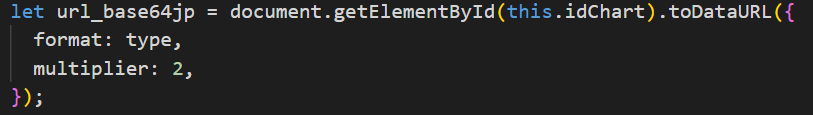


Figure 4.8 Capture of code to download an image

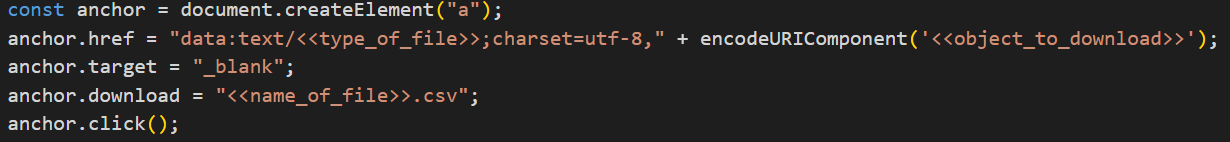


Figure 4.7 Capture of code used for downloading files

Leaflet has been used to generate the dynamic map; the main operation of the map is shown in figure 4.9.

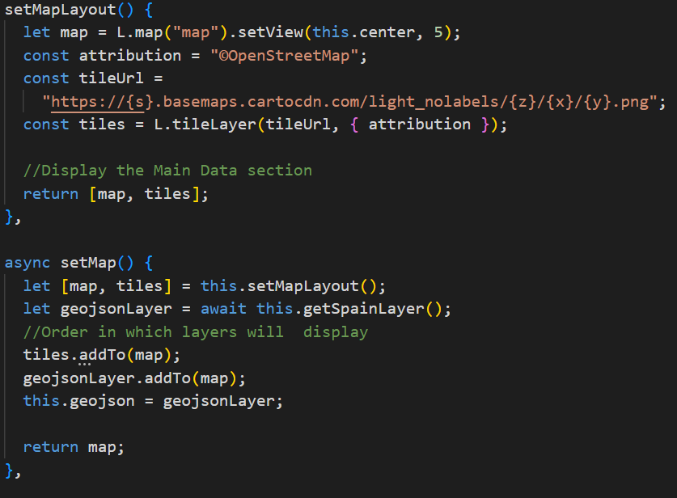


Figure 4.9 Capture of code to display Leaflet map

From the *setMap()* function, *setMapLayout()* is called to load the world map and focus the map around Spain. Within the method, the tile is assigned to a mute map template, to which a tile with the regions of Spain is added. To add dynamism to these regions so that the regions respond to mouse events, the piece of code shown in Figure 4.10 is used.

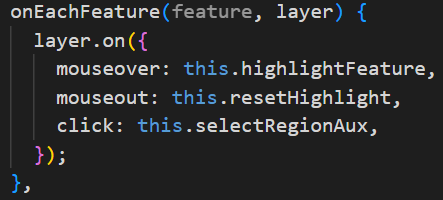


Figure 4.10 Capture of code to generate interactive layers

As for the charts, the Chart.js library is used to represent the data within the ***LineChart*** component. The component receives the data and labels and assigns them to the data section of the code used to create the chart. In addition, because the zoom extension is used, an object within the chart instance is included with the zoom settings.

The creation of the line graph, shown in figure 4.11, is done under the *shallowRef* method of Vue.js because in the case that the parent component wants to update or modify the graph, it would not be able to because what it would be modifying the proxy object of the graph, and not the graph itself. To do this, through *shallowRef* the reactivity of the graph object is removed, allowing that what is emitted to the parent component is the real object and not a copy. Because of this approach, the events of the graph (not of the graph data) must be traced manually since the object is no longer deeply reactive.

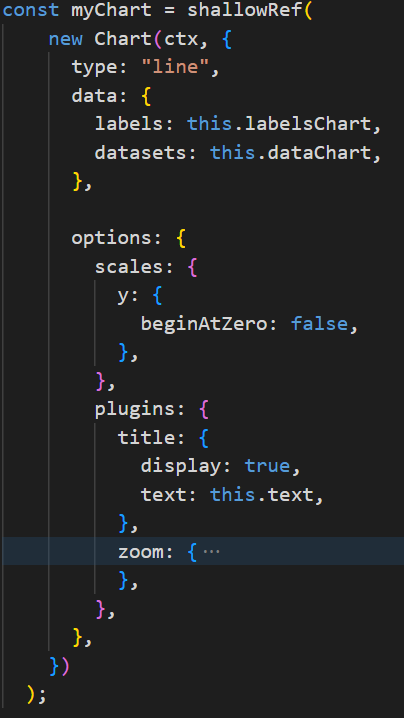


Figure 4.11 Capture of the code used to generate a line chart

Despite this, an auxiliary component is necessary for the generation of graphics of the individual view. This is due to the fact that the component in charge of this logic (***IndividualGraph***) needs to use a loop in the component's template to create several graphs. This loop generates as many auxiliary components (***IndividualGraphWrapper***) as graphics are needed. ***IndividualGraphWrapper*** acts as an intermediary and contains the creation of one ***LineChart*** component along with the data retrieval logic for the graph.

Finally, one of the most important sections to be detailed are the calls to the backend. In the project we have used mainly two: (1) one only to receive data and (2) another one to send content to the server and wait for a response.

For all the calls of the first group, the API\_Codes object of the *global.js* file is used. This object is not only used for the requests, but also to obtain all the graphics of the views, classify them by categories and give them titles. To better understand how it works, image 4.12 shows an example for the Andalusia region. Each region has an object with the necessary codes to call the INE API or Github from the backend. These codes have an object assigned to them with the following attributes: (1) the *type* attribute, which indicates to which data category it belongs, (2) *title*, which indicates the description of the code, (3) *isCovid*, which in case of having true value indicates that this code is used to obtain the values from the Github repository and false to obtain them from the INE API, and (4) *category*, which indicates the periodicity of this data (“D” for Daily, “W” for Weekly “S” for Semiannual, “M” for Monthly, “Q“ for Quaterly A for Annual).

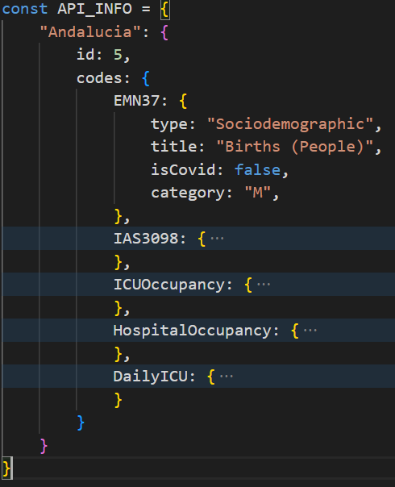


Figure 4.12 Example of the API\_Codes objects for Andalucía

When requesting data to the server, it is necessary to indicate in the link which feature we want to obtain with the code obtained from this object.

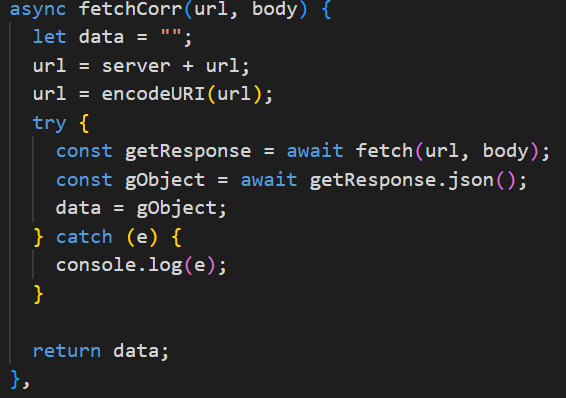


Figure 4.13 Capture of the code used for requests

For the calls, the JS fetch API is used because it offers a simple interface that only needs the server endpoint to make the request. The development of this functionality is shown in figure 4.13. The request URL is made up of a URL that has been passed from the component with information about the dates and the feature, and a prefix with the address of the backend server. In the case of requests that only fetch data, the body element is not necessary, but in the case of correlation, that needs to send the data to the backend, it is necessary to send a body with the format shown in figure 4.14.

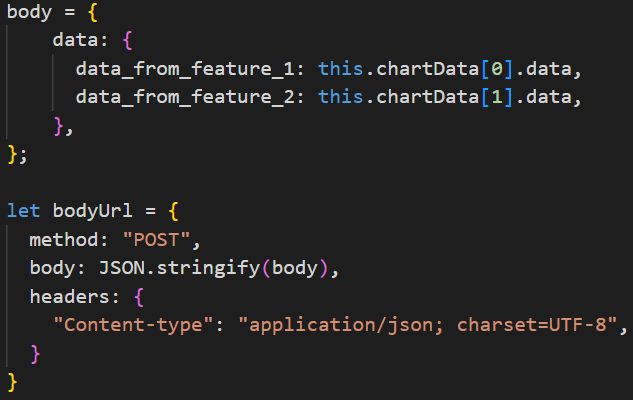


Figure 4.14 Code of the request body

Before working with the data received from the backend it is necessary to transform the response to *json*, so that it can be correctly manipulated as an object with JS.

### 4.1.2 BACKEND

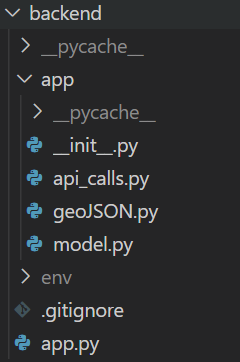


Figure 4.15 Backend file structure

Figure 4.15 shows the structure of the backend, organized as follows:

* **File *app.py****:* is the file with which the backend is loaded. It imports the app module.
* **Directory app**: is the folder where the backend logic is located. It is composed of several files.
* **File *\_init\_.py*** : contains the Flask application instance and some settings.
* **File *api\_calls.py***: it is the backend controller, it contains all the endpoints that the frontend communicates with.
* **File *geoJSON.py****:* contains the variable with the perimeter of all the Spanish regions in geoJSON format. It is stored in the backend because initially this file was too heavy for the frontend.
* **File *model.py*:** contains all the logic related to data extraction, data transformation and correlation calculation.

As for code development, the main parts are detailed below.

First, since our front- and back-end will be on separate ports due to the approach we have chosen, Cross-Origin Resource Sharing (CORS) in Flask must be enabled to allow the front-end JS to make use of the back-end resources. For this purpose, the *Flask\_CORS* library has been used by implementing the code fragment shown in Figure 4.16.



Figure 4.16 Capture of Code fragment to enable CORS

The calls that the backend receives from the frontend have a url very similar to: *"/flask\_api/<string:cod>/<string:start\_date>/<string:end\_date>/<string:region>"*. The first part of the call (e.g. *flask\_api*) allows the controller to identify which method of the model to call. The rest of the URL consists of parameters that are passed to the method.

The geoJSON was obtained from downloading in shapefile format the Municipal, Provincial and Autonomous Community Boundaries section from the Spanish National Geographic Institute download center. The shapefile was then converted into geoJSON using an online converter. Because the resulting file was very heavy and slowed down the map loading, the mapshaper tool [49] was used to reduce by 99% (simplifying the perimeter of the regions) the size it occupied in memory.

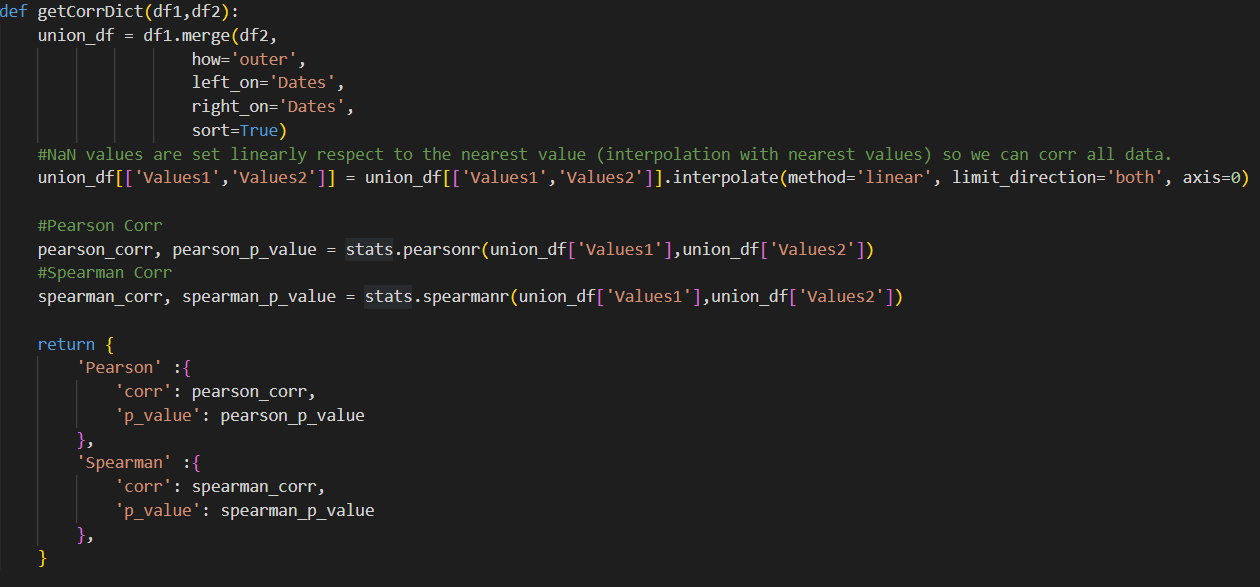
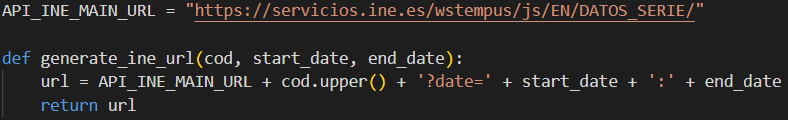


Figure 4.17 Capture of Code used to calculate correlation

For the correlation issue, there was a problem; most of the features have different periodicities, or in the case of having same periodicities, some features are reported for different dates. To solve this, using the Pandas library, two datasets are generated from the data of the two features received, and a *union all* is performed with the date as the key. For the empty values or non-number values it has been decided to interpolate linearly, so that there would be enough data to calculate with the SciPy library the Pearson and Spearman correlations with their respective p-values. The code in Figure 4.17 shows the whole creation process.

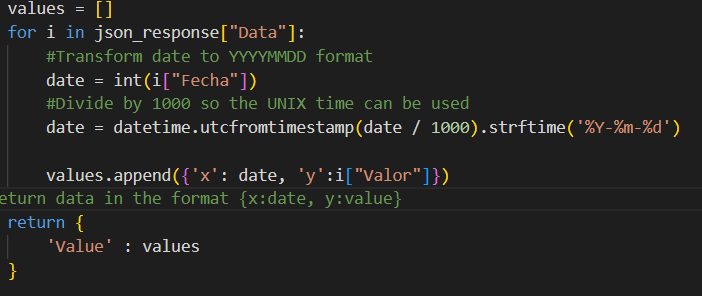
One of the main parts of the project has been the extraction of data from external sites. As mentioned in the project description there are two locations. For the INE API a URL must be generated as shown in Figure 4.18.

Figure 4.18 Capture of Code use to generate url for INE API



The *API\_INE\_MAIN\_URL* constant includes the fixed part of the INE URL. In addition, at the end of this constant the string *DATOS\_SERIES* is added to obtain the time series of values associated with a characteristic of a region. The cod that is included afterwards is the code that has been passed from the frontend thanks to the API\_Codes object and is used to specify which feature I want to request for a region. In addition, this data can be bounded between two dates if a start and end date is specified.

Figure 4.19 Capture of Code to generate Chart.js data



Once the data is received from the server it is modified to have the format required by any Chart.js chart. The format is the one obtained in the variable *values* of the code shown in figure 4.19. It is worth mentioning that the date must be transformed because it is in UNIX format.

On the other hand, there is the extraction of data from the Github repository. This repository has been manually created from the DatAc open data [13]. Once downloaded, the header of the data is changed so that the names of the regions match those used in this project and then a column with data for Spain is added. The file is saved as a csv and stored in the repository, so that any new records can be added directly into the repository. This process is repeated for all COVID-19 metrics that can be obtained from DatAc.

To extract the data from the backend, simply make use of Pandas, as shown in Figure 4.20.

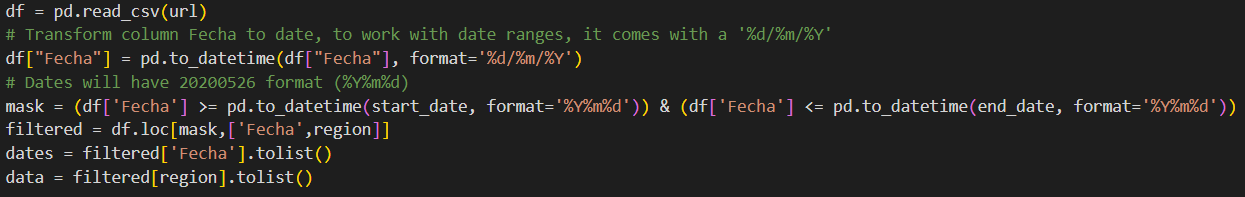


Figure 4.20: Capture. Code uses to read Github repository

Once the file has been read, it is necessary to format the dates to be able to work with them. Using Pandas syntax, it is possible to obtain the data for a given date range. Finally, the format that is sent to the frontend is the same as the one generated with the INE data.

## 4.2 RESULTS

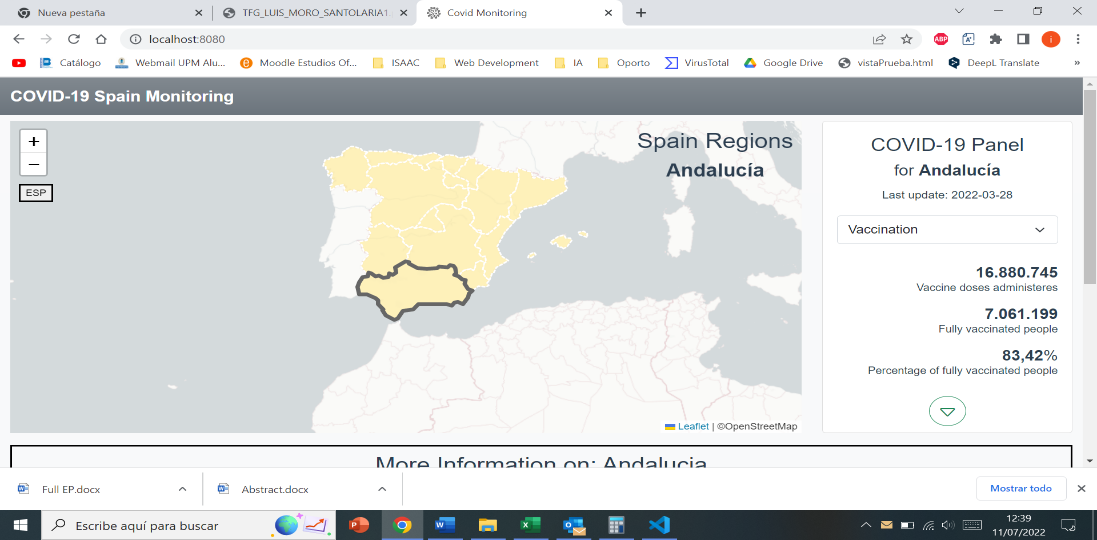
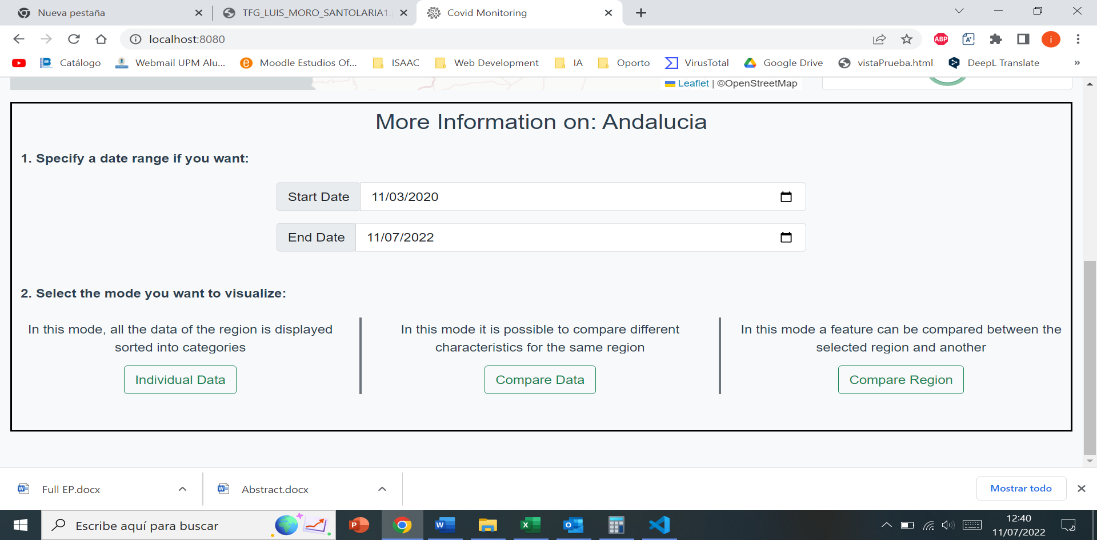


Figure 4.21 Main menu with selected Andalucía region

This section will show the results of the implementation of the code classified by views.

Figure 4.21 shows the main menu once a region is selected. From the right panel of the map you can select between Vaccination, Cases or Death to view the latest information on those topics.

In addition, from the More Information panel, the selection of dates and the buttons that take you to the different views for the region are shown. n.

The individual view is shown in Figure 4.22 with some graphs displayed for the economic category. Obviously not all graphs are shown as they do not fit in a single snapshot. In case of wanting to change the category another one of the buttons shown in the header would have to be selected. The characteristics shown in the graphs include in parentheses the unit in which they are measured.

In the case of the view to compare features, two captures are shown. Figure 2.23 shows the view with several selected features, and figure 2.24 a snapshot having performed the correlation of two features.

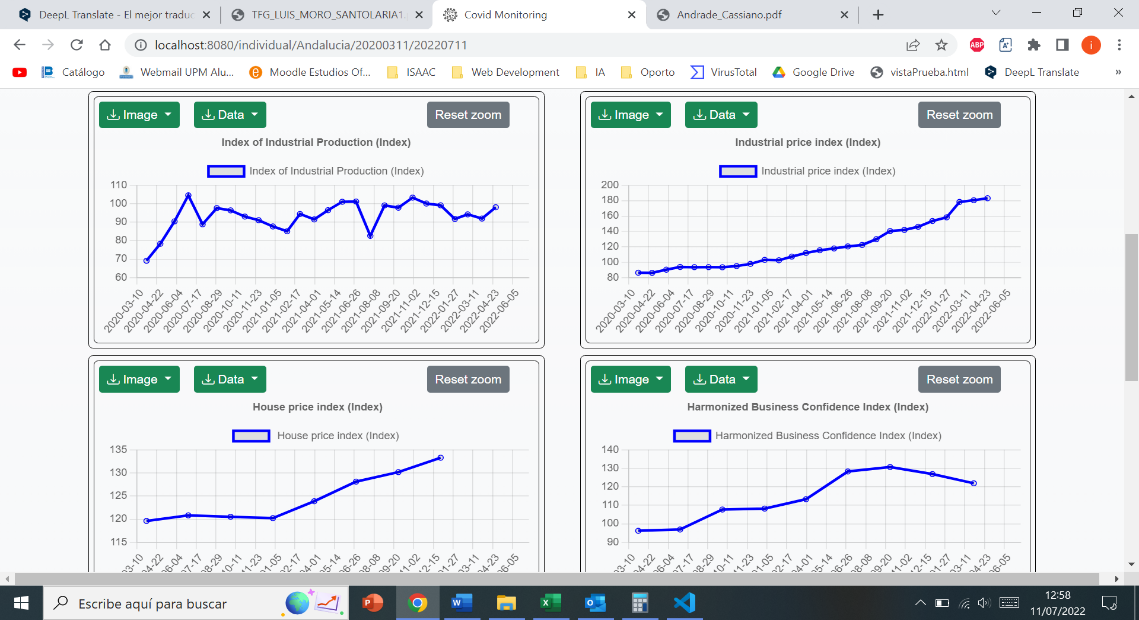
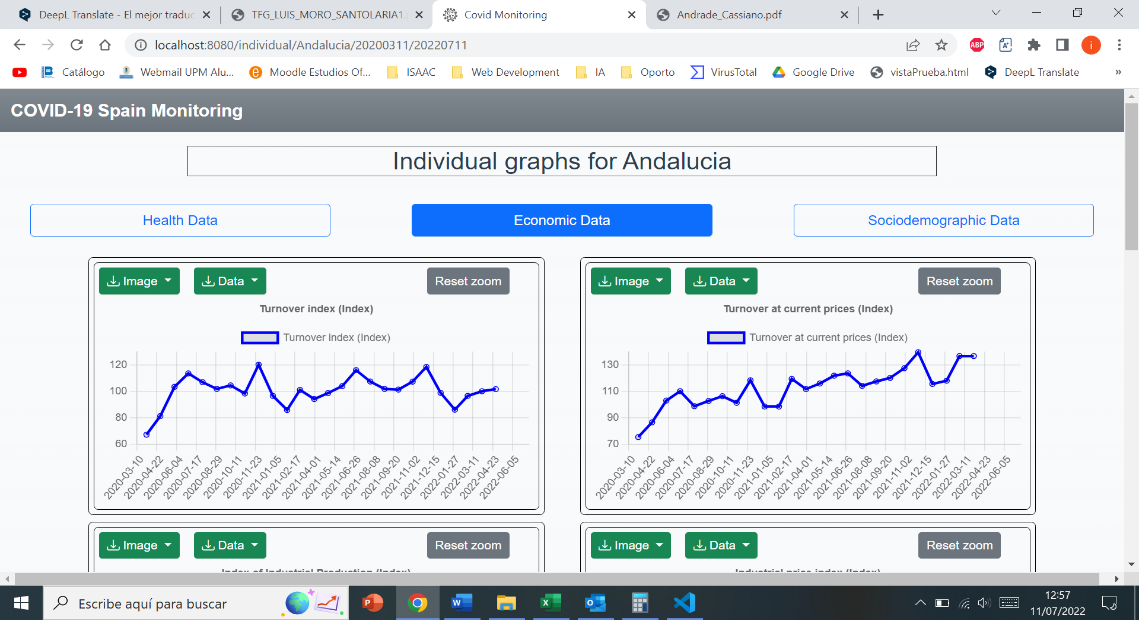


Figure 4.22 Individual view for Andalucia with the Economic data selected

The view for comparing regions is shown in Figure 4.25. Select the feature and the region to be compared from the corresponding dropdowns, regardless of the order.

It should be noted that these views already consider the range of dates selected in the main menu.

Figure 4.26 shows the graphs used. In its own box it is possible to download its image and the data used for the elaboration of the graph.

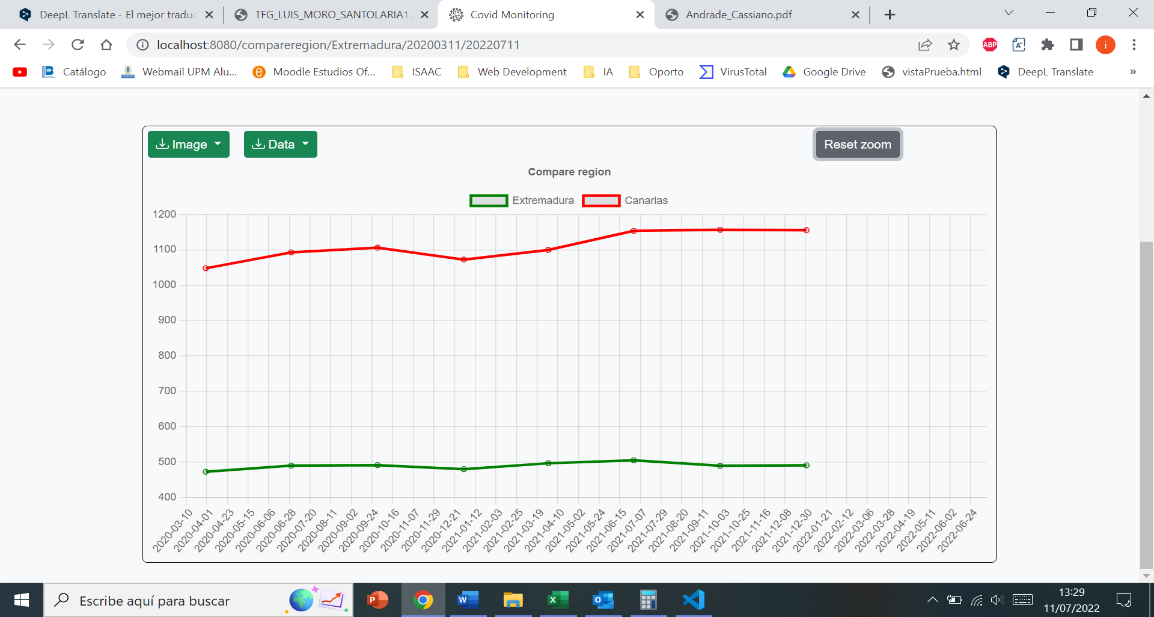
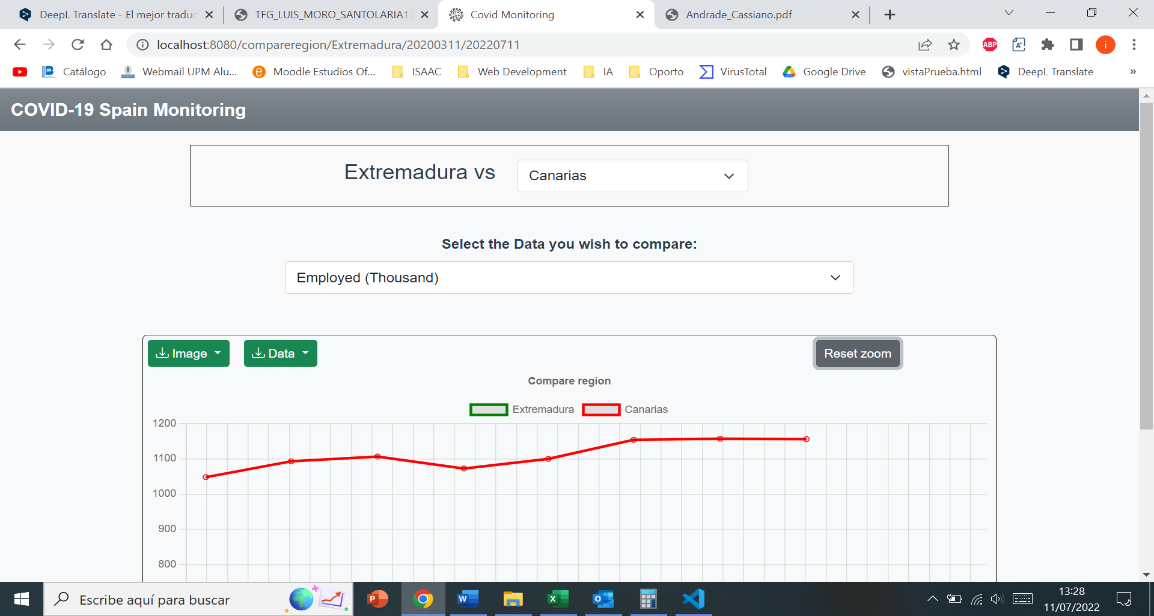


Figure 4.25 Compare region view. Number of employed people in Extremadura and Canarias.

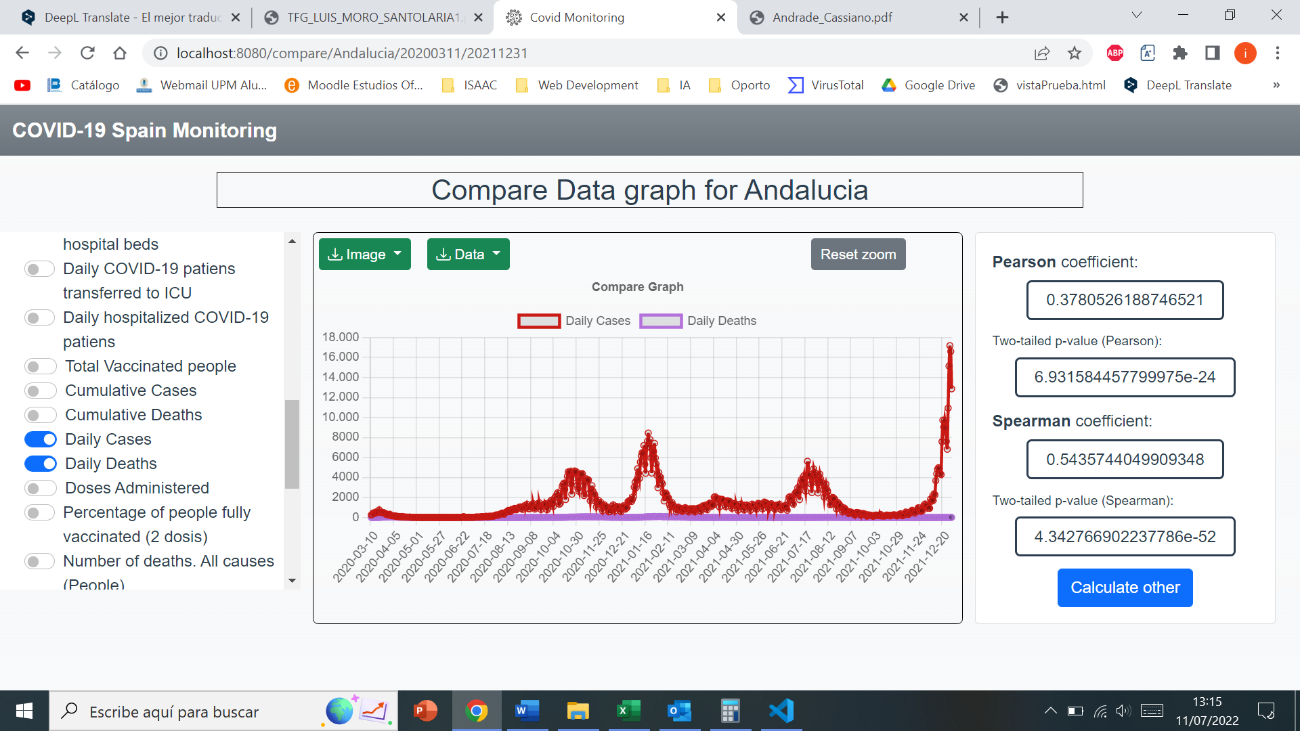


Figure 4.24. Correlation calculated for two features of Andalucia

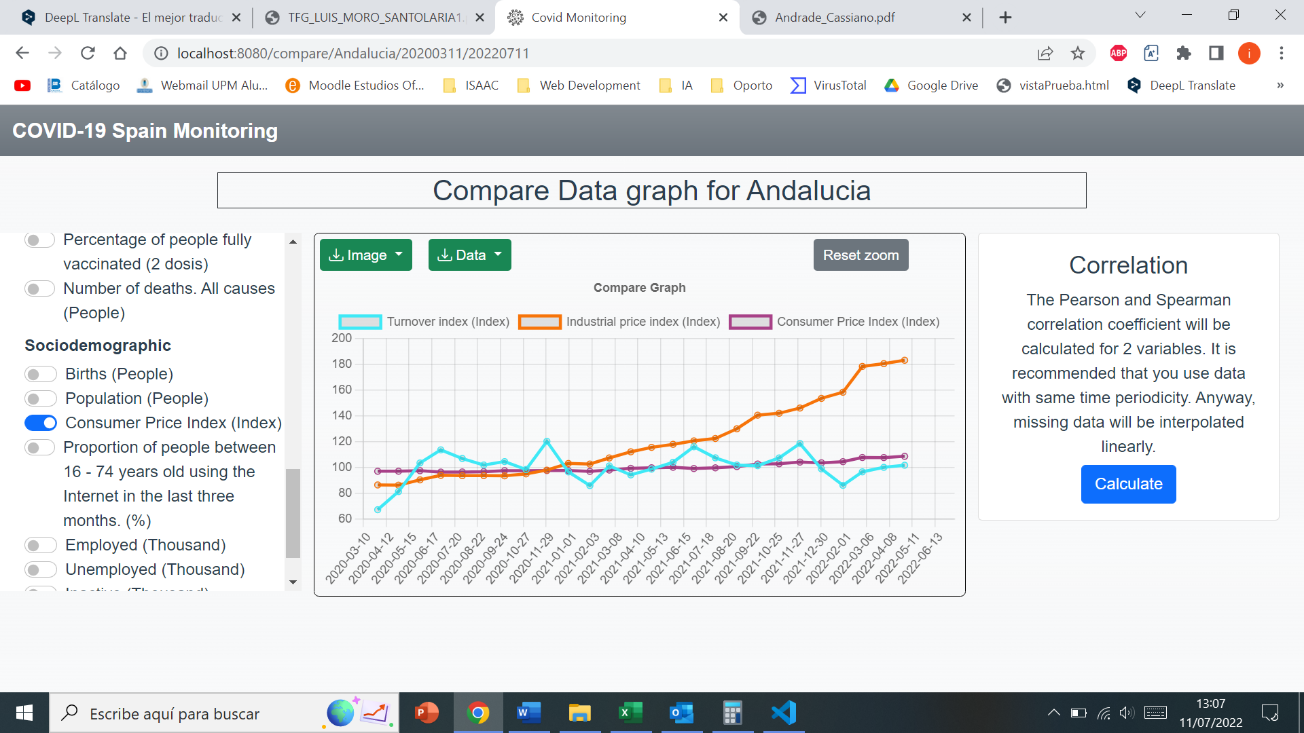


Figure 4.23 Compare Feature view for Andalucia. Display of several features

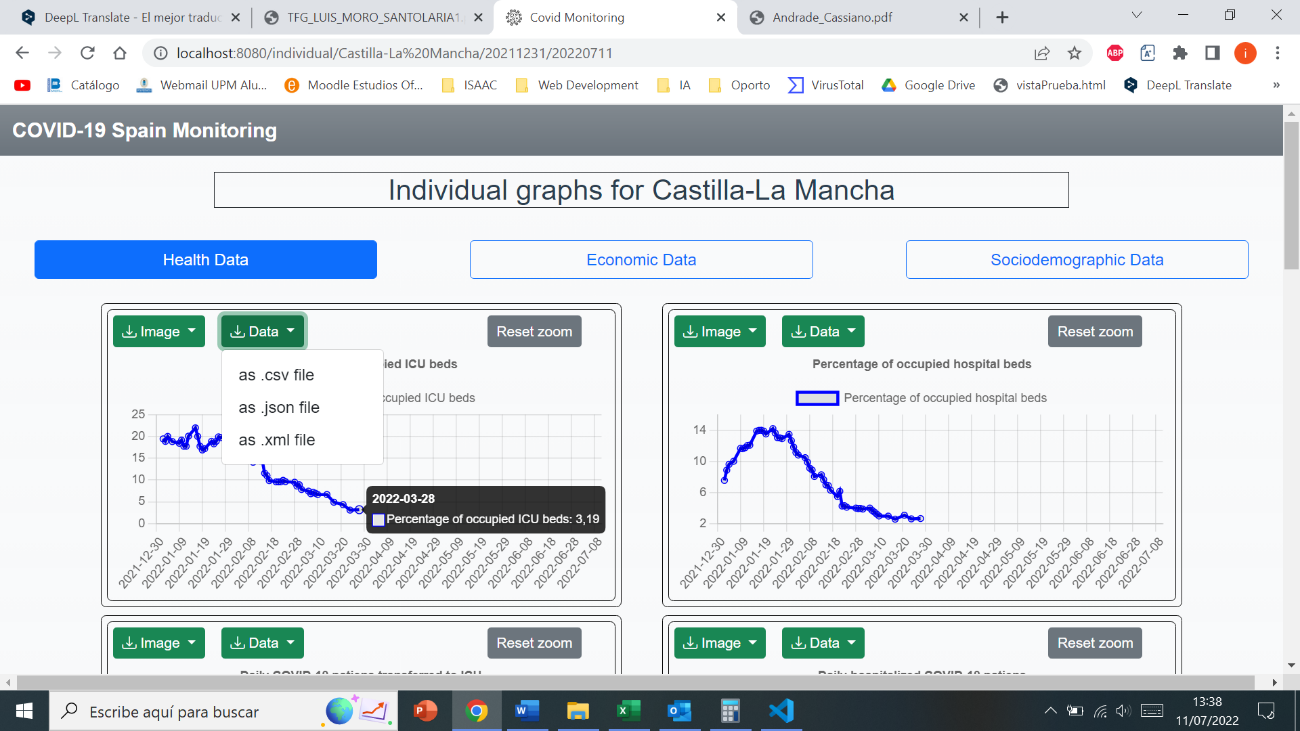


Figure 4.26. Line graph

To conclude the section, Figure 4.27 to 4.29 illustrates the result of downloading the graph in various formats.

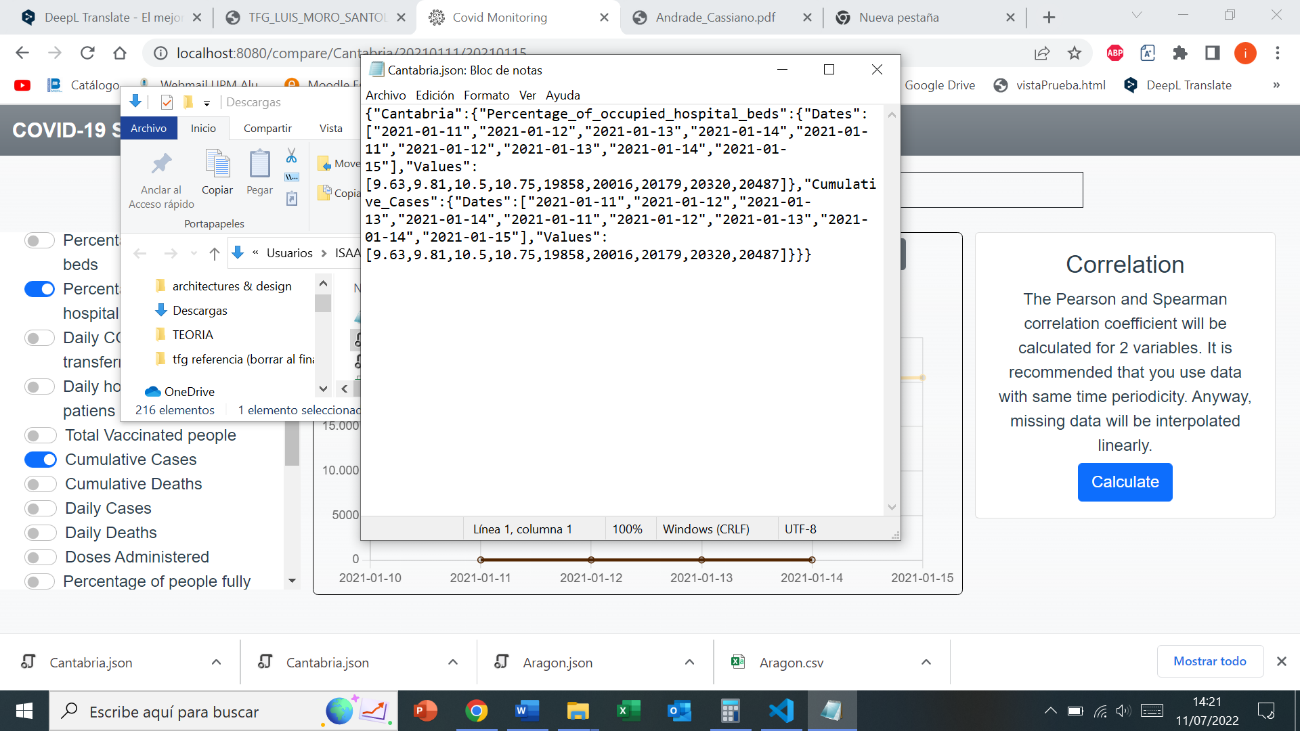


Figure 4.28 JSON file for two features of Cantabria.

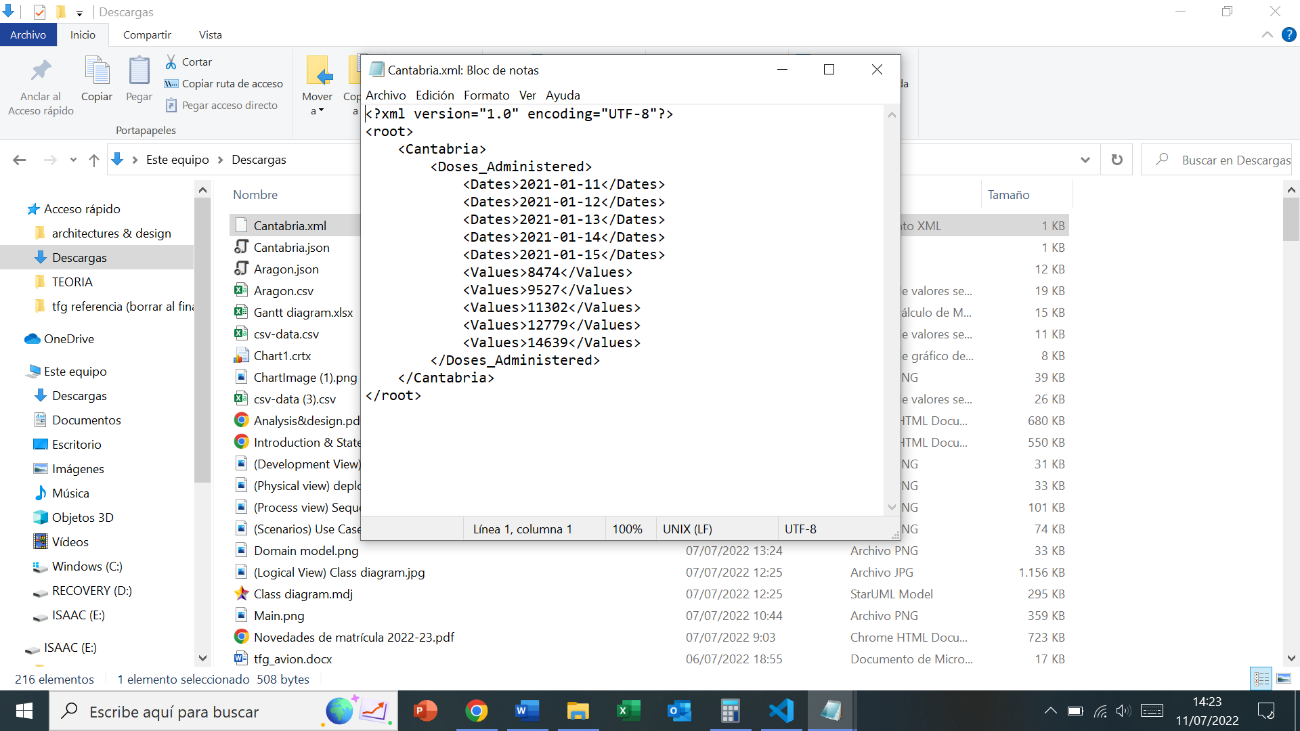


Figure 4.29 XML file for a feature of Cantabria.

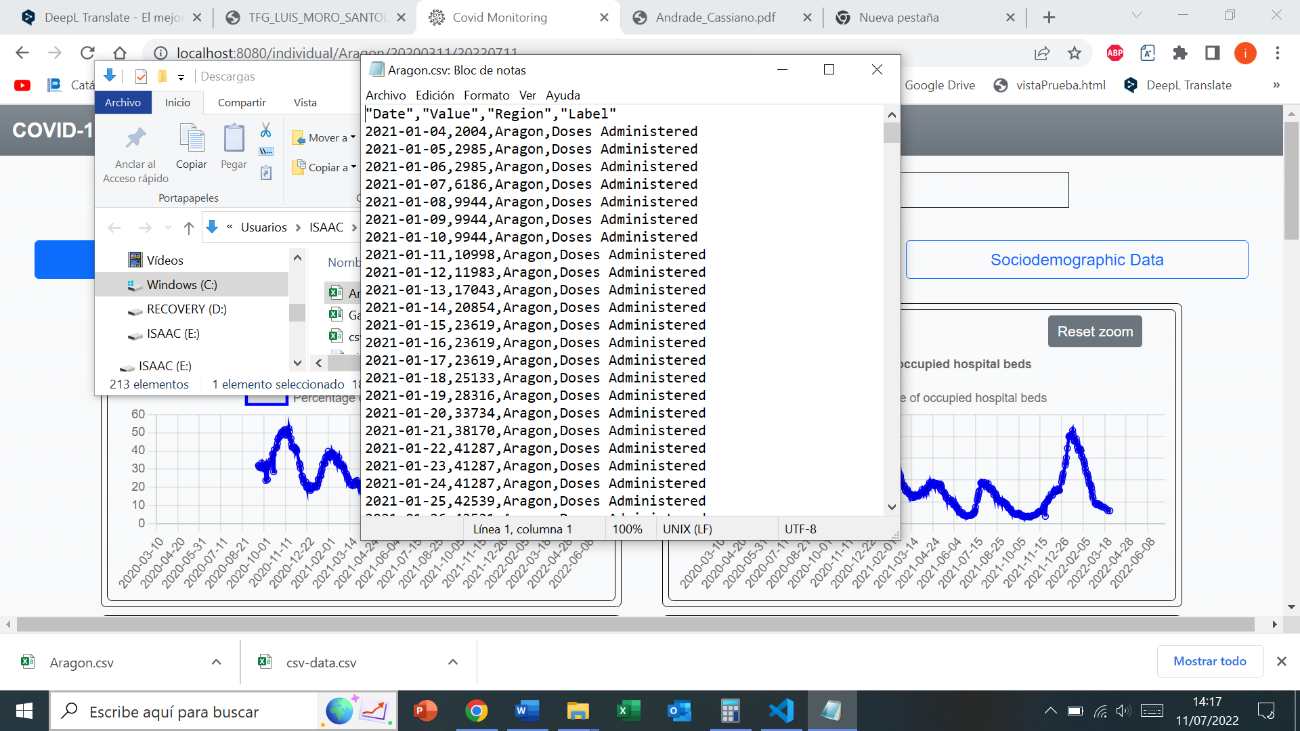


Figure 4.27 Csv file for an individual feature of Aragón.

## 4.3 TESTING

Tests are used to find bugs and demonstrate the correct functioning of the application. Usability tests have been used for the project, which aim to check how easy the design makes the use of the application.

To achieve this, a participant with a minimum of analytical knowledge was selected, since the application includes analytical features, and was asked to perform a couple of tasks so his behavior could be analyzed. This testing was carried out in person to closely observe his behavior when using the application.

Some of the tasks requested along with the behavior and response obtained are detailed below:

**Task**: *How do you view the most recent Vaccination records for Galicia?*

**Resolution:** The participant has easily solved this operation.

**Task:** *Access the screen that displays all the socio-demographic category data for a region, with the date range selected by default.*

***Resolution****:* The participant had a hard time figuring out which mode to enter, but once in the mode it was easy to find the category. For this reason, a brief description has been added to the modes, which has made the task easier.

**Task**: *Displays in a single graph the Consumer Price Index feature together with the number of daily cases and the number of births for Spain for any date range.*

**Resolution**: The participant has quickly solved this task. He had some difficulty in selecting the button that selects the Spain region, but finally he found it.

**Task**: *Calculate the correlation between characteristics with a date range of December 2019 and May 2020.*

**Resolution**: The participant has encountered an error when trying to correlate 3 features. Therefore, an alert has been implemented so that it can only calculate correlation with two features.

**Task**: *Download the data of any chart in csv format.*

**Resolution**: The participant has solved this task easily.

**Task**: *Compare the percentage of vaccinated people between the Community of Madrid and Andalusia in the last year.*

**Resolution**: Due to the added description, he solves the task. It can be observed that the participant already has some fluency using the web application.

**Task**: *How would you describe your overall experience with the product?*

**Resolution**: The participant is very satisfied with the interface design, and with the functionalities offered by the application.

**Task**: *What do you think of the loading times in the display of the different elements?*

**Resolution**: He is also very satisfied with the loading times and considers that there would be no need to optimize them.

# REFERENCES

1. World Health Organization. (n.d.). *Coronavirus*. World Health Organization. Retrieved from https://www.who.int/health-topics/coronavirus
2. World Health Organization. (2020). WHO Director-General’s opening remarks at the media briefing on COVID-19-11 March 2020.
3. Liu, Y. C., Kuo, R. L., & Shih, S. R. (2020). COVID-19: The first documented coronavirus pandemic in history. *Biomedical journal*, *43*(4), 328-333.
4. García, J. M. M., Izquierdo, J. A., & Pérez, M. I. G. (2020). COVID-19 en España, ¿cómo hemos llegado hasta aquí? *Atención Primaria*, *52*(10), 676.
5. Johns Hopkins Coronavirus Research Center (2021). Covid-19 Dashboard [Online]. Retrieved from *https://coronavirus.jhu.edu/map.html*
6. Maital, S., & Barzani, E. (2020). The global economic impact of COVID-19: A summary of research. *Samuel Neaman Institute for National Policy Research*, 2020, 1-12.
7. Tadesse, S., & Muluye, W. (2020). The impact of COVID-19 pandemic on education system in developing countries: a review. *Open Journal of Social Sciences*, 8(10), 159-170.
8. Mofijur, M., Fattah, I. R., Alam, M. A., Islam, A. S., Ong, H. C., Rahman, S. A., ... & Mahlia, T. M. I. (2021). Impact of COVID-19 on the social, economic, environmental and energy domains: Lessons learnt from a global pandemic. *Sustainable production and consumption*, 26, 343-359.
9. INE - Instituto Nacional de Estadística. (n.d.). Información estadística para el análisis del impacto de la crisis COVID-19. INE. Retrieved from *https://www.ine.es/covid/covid\_inicio.htm*
10. INE - Instituto Nacional de Estadística. (n.d.). Productos y Servicios / Datos abiertos / API JSON / Inicio. INE. Retrieved from *https://www.ine.es/dyngs/DataLab/manual.html?cid=45*
11. Vahedi, A., Moghaddasi, H., Asadi, F., Hosseini, A., & Nazemi, E. (2022). Applications, features and key indicators for the development of Covid-19 dashboards: A systematic review study. *Informatics in Medicine Unlocked*, 100910.
12. Eurostat. (2021). *European Statistical Recovery Dashboard*.
13. Martorell-Marugán, J., Villatoro-García, J. A., García-Moreno, A., López-Domínguez, R., Requena, F., Merelo, J. J., ... & Carmona-Sáez, P. (2021). DatAC: A visual analytics platform to explore climate and air quality indicators associated with the COVID-19 pandemic in Spain. *Science of the Total Environment*, *750*, 141424.
14. Schwaber, K., & Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, *21*(19), 1.
15. Ivanković, D., Barbazza, E., Bos, V., Fernandes, Ó. B., Gilmore, K. J., Jansen, T., ... & Kringos, D. (2021). Features constituting actionable COVID-19 dashboards: descriptive assessment and expert appraisal of 158 public web-based COVID-19 dashboards. *Journal of medical Internet research*, *23*(2), e25682.
16. Dong, E., Du, H., & Gardner, L. (2020). An interactive web-based dashboard to track COVID-19 in real time. *The Lancet infectious diseases*, *20*(5), 533-534.
17. Schechtman, K. W. (2021, May). Analysis & updates | A Wrap-Up: The Five Major Metrics of COVID-19 Data. The COVID Tracking Project. Retrieved from [*https://covidtracking.com/analysis-updates/five-major-metrics-covid-19-data/*](https://covidtracking.com/analysis-updates/five-major-metrics-covid-19-data/)
18. World Health Organization. (2020). Strengthening and adjusting public health measures throughout the COVID-19 transition phases: policy considerations for the WHO European Region, 24 April 2020 (No. WHO/EURO: 2020-690-40425-54211). World Health Organization. Regional Office for Europe.
19. Naeem, S. B., & Bhatti, R. (2020). The Covid‐19 ‘infodemic’: a new front for information professionals. *Health Information & Libraries Journal*, *37*(3), 233-239.
20. Sutherland, L. (2021). *Workshop: Introduction to GIS Dashboards*.
21. *Pratt, M. (*2020, September 9). Creating the Dashboard for the Pandemic. Esri. Retrieved from [*https://www.esri.com/about/newsroom/arcuser/johns-hopkins-covid-19-dashboard/*](https://www.esri.com/about/newsroom/arcuser/johns-hopkins-covid-19-dashboard/)
22. WHO COVID-19 Dashboard. Geneva: World Health Organization, 2020. Retrieved from: *https://covid19.who.int/*
23. RTVE.es. (n.d.). El Coronavirus: Gráficos, Mapas y Datos del COVID-19. Retrieved from *https://www.rtve.es/noticias/coronavirus-graficos-mapas-datos-covid-19-espana-mundo/*
24. Fuster, M. T. Z., & ZARGOZA, C. G. (2021). La innovación al servicio de la información sobre la crisis del Covid-19: el caso de RTVE Lab. *Plataformas, consumo mediático y nuevas realidades digitales. Hacia una perspectiva integradora.*, *17*, 2.
25. Al-Fedaghi, S. (2011). Developing web applications. *International journal of software engineering and its applications*, *5*(2), 57-68.
26. What Is a Framework? (2022, March 12). Codecademy News. Retrieved from *https://www.codecademy.com/resources/blog/what-is-a-framework/*
27. U. (2018, December 18). Top frameworks para desarrollo web. Platzi. Retrieved *from https://platzi.com/blog/lenguajes-frameworks-librerias-backend/*
28. Framework vs Library: Full Comparison. (2021, October 23). InterviewBit. Retrieved from [*https://www.interviewbit.com/blog/framework-vs-library/#difference-between-library-and-framework*](https://www.interviewbit.com/blog/framework-vs-library/#difference-between-library-and-framework)
29. Stack Overflow.(2021). Most popular technologies [Data set]. Retrieved from *https://insights.stackoverflow.com/survey/2021*
30. Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (2021). Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd. In *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*.
31. GeeksforGeeks. (2022, June 14). Frontend vs Backend. Retrieved from [*https://www.geeksforgeeks.org/frontend-vs-backend*](https://www.geeksforgeeks.org/frontend-vs-backend/)*/*
32. Vue.js - The Progressive JavaScript Framework | Vue.js. (n.d.). Vue.Js. Retrieved from [*https://vuejs.org*](https://vuejs.org/)*/*
33. Bainomugisha, E., Carreton, A. L., Cutsem, T. V., Mostinckx, S., & Meuter, W. D. (2013). A survey on reactive programming. *ACM Computing Surveys (CSUR)*, *45*(4), 1-34.
34. Imperative vs. declarative rendering of user interfaces. (n.d.). Explain Programming. Retrieved from [*https://www.explainprogramming.com/theory/imperative-declarative-react/#definitions*](https://www.explainprogramming.com/theory/imperative-declarative-react/#definitions)
35. Jadhav, M. A., Sawant, B. R., & Deshmukh, A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, *6*(3), 2876-2879.
36. Xing, Y., Huang, J., & Lai, Y. (2019, February). Research and analysis of the front-end frameworks and libraries in e-business development. *In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering* (pp. 68-72).
37. Patel, R. (2022, June 15). What is Vue.js? Vue.js Pros and Cons (A Complete Guide). Space-O Technologies (Canada). Retrieved from[*https://www.spaceo.ca/blog/vue-js-pros-and-cons/*](https://www.spaceo.ca/blog/vue-js-pros-and-cons/)
38. Welcome to Flask — Flask Documentation (2.1.x). (n.d.). Flask. Retrieved from[*https://flask.palletsprojects.com/en/2.1.x/*](https://flask.palletsprojects.com/en/2.1.x/)
39. Deery, M. (2021, September 6). What Is Flask and How Do Developers Use It? A Quick Guide. CareerFoundry. Retrieved from *https://careerfoundry.com/en/blog/web-development/what-is-flask/#what-is-the-flask-framework*
40. Walter, N. (2019, December 5). Flask Essential Training Online Class | LinkedIn Learning, formerly Lynda.com [Video]. LinkedIn. Retrieved from [*https://www.linkedin.com/learning/flask-essential-training*](https://www.linkedin.com/learning/flask-essential-training)
41. JavaScript | MDN. (2022, May 30). JavaScript. Retrieved from *https://developer.mozilla.org/es/docs/Web/JavaScript*
42. Eguiluz, J. (n.d.). Introducción a XHTML. uniwebsidad. Retrieved from [*https://uniwebsidad.com/libros/xhtml*](https://uniwebsidad.com/libros/xhtml)
43. CSS | MDN. (2022, May 30). CSS. Retrieved from [*https://developer.mozilla.org/es/docs/Web/CSS*](https://developer.mozilla.org/es/docs/Web/CSS)
44. Cómo crear una API pública con AWS. (n.d.). Amazon Web Services, Inc. Retrieved from *https://aws.amazon.com/es/what-is/api/*
45. Berga, M. (2022, June 21). JSON vs XML: Which one is better? Blog | Imaginary Cloud. Retrieved from *https://www.imaginarycloud.com/blog/json-vs-xml/*
46. C. (2016, August 21). What do raw.githubusercontent.com URLs represent? [Stack Overflow post]. Stack Overflow. Retrieved from *https://stackoverflow.com/questions/39065921/what-do-raw-githubusercontent-com-urls-represent*
47. Kruchten, P. B. (1995). The 4+ 1 view model of architecture. IEEE software, 12(6), 42-50.
48. Medlin, J. (2021, November 3). Combining Flask and Vue. TestDriven.Io. *https://testdriven.io/blog/combine-flask-vue/*
49. mapshaper. (n.d.). Mapshaper. *https://mapshaper.org/*