Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: **Manel Moreno Blanco**

Titulació: Grau en Tècniques d'Interacció Digital i de Computació

Títol de Treball Final de Grau: Fleet Manager Control de flotes

Director/a: Jordi Mateo Fornés

Presentació

Mes: Juliol

Any: 2022

Abstract

The aim of this project is to bring real time information about the location of a company's vehicles in order to increase both its control capacity and efficiency.

This project's implementation is divided in three main parts. The first part corresponds to the backend, where we centralize and store all the project's data. The second part is assigned to the frontend, where we need to be able to have programming views and the results of the data processing done by the backend, using html and css. The third part of this project corresponds to Mikrotik device configuration, as we need to make them able to send their coordinates to the GPS via Curl to the API, in order to store them.

All of this will require a database where we process this objects, Google Maps to be able to show a map with the locations in our web's views, and PHP to stablish a connection with the database and further process these elements. We will we working with Larabel, as it is an easy-to-use framework for real life item location management.

Table of contents

Abstract *1*

Introduction *4*

Objectives *5*

Methodology *5*

Chosen Techniques 5

Project Planification 6

Project Monitoring 7

Solution Design *8*

Database design 8

Architecture 9

Backend 9

Frontend 11

Mikrotic Configuration 23

Deployment 25

Results *26*

Functionality and final product 26

Product's impact to the company 26

Conclusions *27*

Bibliography *28*

Introduction

Iguana SL is a telecommunication company that offers high-speed internet to Anoia's residents. It also provides technical service related to optic fiver and television installation, complemented with other services related to network usage.

This company assigns a vehicle to each one of its technicians to allow them to go to their client's address in order to provide them with the services aforementioned or to resolve network -related incidents.

One of the main troubles that this company faces is the lack of real time vehicle control, and the inability to confirm if the technicians have arrived at the correct location at the time agreed with their clients. It was also unable to know when one of their workers had finish a task and was able to move to the next one.

After analysing different ways to assess this problem, the main proposal was to add a GPS device to the vehicles with the aim of locating them in real time.
Taking advantage of this technology we should be able to know if a technician has correctly arrived at the designed place and when has him finished his job.

FleetManager was then born to solve this problem. Its purpose is to create a platform able to manage all this data and configure a GPS device capable of bringing real time information about the location of the company's vehicles.

In order to initiate this project, we carried out a study case to analyse the different models and scenarios which we would be working on. The chosen case addressed a worker that had a vehicle assigned and needed to go to a client's address to carry out his service.

One of the main setbacks that we found was to narrow down which was the concrete model that we would be using for information management. Another one of the problems that we observed was to match the coordinates obtained by the GPS with the corresponding vehicle.
Once these difficulties were considered, we decided that using a web page was the best way to address them considering the advantages that it entails, such as model data storage and the functionality that it provides.

The functionality of this web page will rest with one or more users (in charge of worker management) that will be able to manage this data.

Objectives

- Worker's data management.
- Vehicle's data management.
- Worker-vehicle matching.
- Platform users' data management.
- Linking the vehicles with their corresponding coordinates.
- Mikrotik device GPS configuration.
- Script programming to send coordinates to the API.
- Server configuration to upload our project.
- Working and incorporating into Google Maps' API.
- Displaying vehicle locations in the map.
- Displaying a vehicle localisation summary.

Methodology

Chosen Techniques

All of this project's development has been carried out using Laravel[1]. Laravel is a framework that uses PHP[2] programming language and allows us to work with database models avoiding directly using SQL language. It also provides us with a lot of functionalities that help us to program and keep a good maintenance of our application, such as new table and model creation.

Laravel uses a design pattern called **controller view model**. Designed patterns are problem resolving techniques common in both program development and in other fields related to interaction or interface design. It is often crucial to use a pattern design, as it helps us during the programming process, and will allow us to avoid poor application maintenance in the future.

This pattern is manly used to obtain a three-layer separation. The first layer takes care of the model, where the object is defined and how it will be processed. The second layer belongs to the controller, which is defined as the functions or actions that process the object. The final layer is the view, which displays an interface for the user where he will be able to see the final result of the sum of these actions conducted on the model or object.

A key part of this work is vehicle localization to manage their real time location. To solve this, we have developed an API using Laravel that allows us to handle all this data.

An API is the combination of procedures that allow two program components to communicate between them using a set of stablished definitions and protocols. It is what enables us to configure a GPS (Global Positioning System) to obtain satellite, real time, very accurate information about the geographic location of a vehicle.

Once vehicle location was solved, we programmed a GPS device equipped with an internet connection sim card. This device will send the coordinates via fetch tool. Fetch tool is a Mikrotik tool that sends POST/GET and other kind of requests to a remote server, in this case our API, in order to store the coordinates of the vehicle that we have assigned to it.

Our Mikrotik[3] was programmed using WinBox[4]. To enable the scripts that run at every moment we also needed to install two antennas to the device in order to activate the GPS function and obtain their coordinates.

We also used Tailwind css[5] to keep our data in an attractive and intuitive demeanour. Tailwind css is a style library that provides us with tools to make the visible part of our web page. Unlike Boostrap[6], Tailwind brings us a higher performance in exchange for lesser support and a smaller user community.

Lastly, one of the most important tools that we have used to manage the databases is mysql[7] as well as phpMyAdmin[8]. Using them in tandem allowed us to further facilitate database management and displaying.
We used a relational database model to keep an easier maintenance and a good model-table relation. This eases the process of keeping information about more than one model and obtaining extra information about the related model.

Project Planification

Project planification is based on narrowing down the project's duration, its budget and the amount of formation required by the worker which will program it.

In this case, formation was carried out during a month, where we realised the familiarization with Laravel framework, and we obtained basic knowledge about its functioning.

We also stablished different project stages, where the most part of time was assigned to software development, but we also had to set time aside for Mikrotik device configuration.
This time-delimited task separation has shown to optimize the efficiency and to shorten projects duration. We also agreed on biweekly meetings in order to make actualizations about the state of our project and to evaluate the obtained performance.

Project Monitoring

In Figure 1 we are able to see the duration of the different tasks that we have been working on through the project.

The first part consisted of worker formation, which contains Laravel framework familiarization and learning, which lasted for one and a half months.
Further on we started proper project developing, including both the backend and the frontend. It had a 6-month duration.
Later on, we carried out Mikrotik device configuration, which lasted for two more months.
Finally, all the functionality verification tasks were carried out during the week prior to its deployment.

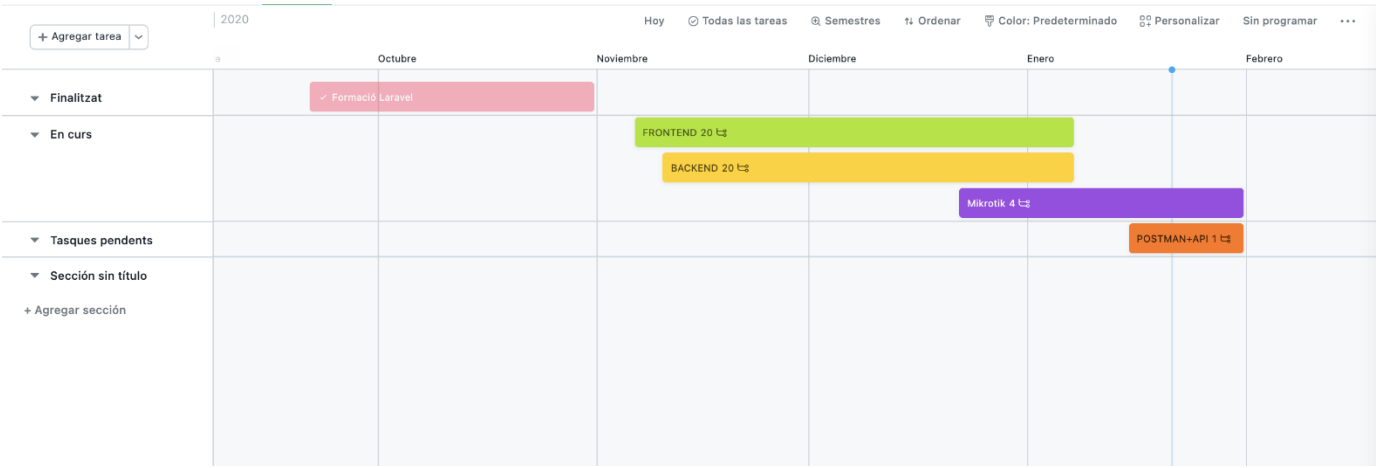


Figure 1 Gantt's Diagram

To stablish the monitoring of this application we worked using the SCRUM method. This method consists in keeping a regular task realization process with the aim of working in a collaborative manner in order to promote teamwork. Using this working method should allow us to obtain the best result possible in any determined project.
This method also relays on setting objectives during project development and to set up meetings to update everyone involved about the state of the tasks and the efficiency obtained.

Once we obtained our first goals, we configured a Git[12] to keep all the deploys of the tasks that had been carried out. For this development's maintenance we created two branches on our Git. One was named test, where we developed and verified that all of the application's functionality was working correctly, and a second branch named production to upload all of this tested functionality once we made sure that there were no mistakes.

Solution Design

Database Design

In the following diagram (*Figure 2*) we can see a table relation model. There are three models in our database. On the one side, the attributes of the workers' data that consists of a DNI, name, surname and phone number, and a vehicle identifier, which is related to our vehicle data. This allows us to obtain detailed information about the worker and the vehicle's relation.
The third model consists of the vehicle's coordinates, which are also related through the vehicle's identifier.
All of this allows us to link to a determined vehicle a worker and its coordinates.

To manage this models we are in need of another one, the user which will manage this data and will obtain the information generated by it. This information will be subsequently displayed on our webpage.

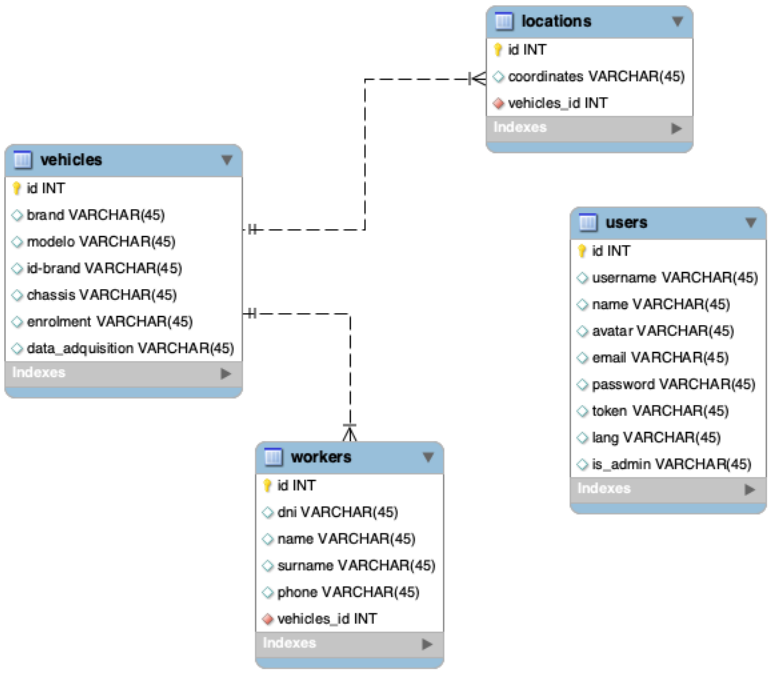


Figure 2: UML database design

Architecture

In *Figure 3* we are able to see the itinerary and functionality of this project and how we send the data to FleetManager's platform. As we can see, we start with a worker to whom a vehicle is assigned. Then we have a Mikrotik device equipped with GPS assigned to the same vehicle that stores its coordinates. All this data goes then through FleerManager's API, whit the help of Google Maps'[9] API, to display on the screen the exact ubication of this vehicle and its assigned worker.

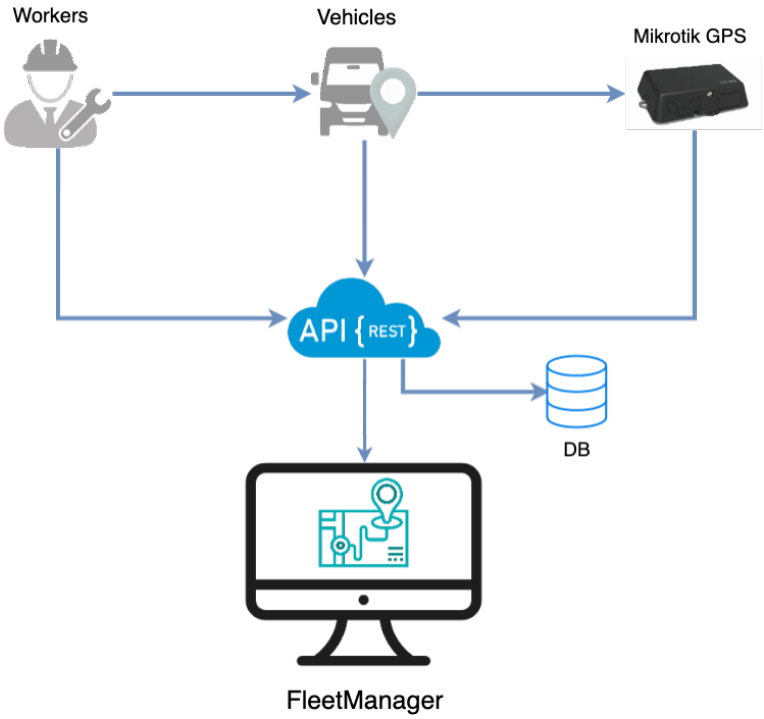


Figure 3: Functional diagram of the project

Backend

In the backend part of the project, we develop all of the necessary procedures to allow us to treat and store all of the information needed when a user makes an action, or a device sends his coordinates.

We will only be using one of the API's routes, since we expect the Mikrotik device to send its coordinates through the endpoint that we have defined.

Here we can see a list of all the requests made by the backend:

Method http	Endpoint	Response status	Funcionality
POST	api/locations/insert	Succes:200 Error:404	GPS coordinate sending
PATCH	api/profiles/{user_username}	Succes:200 Error:404	User data updating
POST	api/vehicles/insert	Succes:200 Error:404	New vehicle creation
GET	api/vehicles/index	Succes:200 Error:404	Display all vehicles
GET	api/vehicles/{vehicle}	Succes:200 Error:404	Display vehicle information
GET	api/vehicles/search	Succes:200 Error:404	Search a vehicle by its license plate
POST	api/vehicles/{vehicle}	Succes:200 Error:404	Vehicle data update
GET	api/vehicles/info/{vehicle}	Succes:200 Error:404	Display advanced information about a vehicle and its route
POST	api/workers/insert	Succes:200 Error:404	Create a new worker
GET	api/workers/index	Succes:200 Error:404	Display all workers
GET	api/workers/info/{worker}	Succes:200 Error:404	Display advanced information and the location of a worker
GET	api/workers/{worker}	Succes:200 Error:404	Display editable information about a worker
POST	api/workers/{worker}	Succes:200 Error:404	Update worker data
DELETE	api/workers/delete/{worker:id}	Succes:200 Error:404	Delete the worker specified by an identifier
DELETE	Api/vehicles/delete/{vehicles:id}	Succes:200 Error:404	Delete the vehicle specified by an identifier
GET	api/locations/maps	Succes:200 Error:404	Google Maps testing
GET	api/location/{locations:id}	Succes:200 Error:404	Display a vehicle’s coordination

In the function noted further on we are able to see how we process the data that we get from the Mikrotik device.
With these coordinates we will firstly create the object “location”, and in its attributes we will store the latitude and longitude, plus the vehicles identifier that links it to a target vehicle. If the connection has been correctly stablished, we will display a verifying message accompanied with the state 200 of the request.
If there is any problem with the request, an error message will be shown informing of the problem that our program detected.

Following this, we did some API single testing using fake data that we are able to create, update and eliminate specifically for this process. This allows us to verify if there is any creation, elimination or displaying error.

Frontend

In relation to the frontend, we assigned different tasks to develop the various views. It is in this section where the user can observe the interactions between all the functions carried out by the controller.

In the following list we show all our web platform's routes that the user can use to access the different views. We can also see the functions with which the user is able to interact with the formulary to create certain objects.
Laravel doesn't by default let a user access to Middleware if he is not registered. Middleware is a service that brings security to the platform, both to the API and to the web page, and allows us to create various restrictions in them. In this example, we have some kinds of views and functions restricted which only an administrator user can access.

Method http	Endpoint	Response status	Funcionality
GET	users/index	Succes:200 Error:404	Display all users
GET	users/insert	Succes:200 Error:404	Display the new worker creation view
POST	users/insert	Succes:200 Error:404	Worker creation function
PATCH	users/info/{user:id}	Succes:200 Error:404	User profile picture and data updating
GET	users/info/{user:id}	Succes:200 Error:404	Display user information
DELETE	users/delete/{user:id}	Succes:200 Error:404	Delete the user selected by its identifier
GET	home/	Succes:200 Error:404	Home page
GET	lang/{Lang}	Succes:200 Error:404	Choose the platform's language
GET	profiles/{user:username}	Succes:200 Error:404	Display the logged user's profile
PATCH	profiles/{user:username}	Succes:200 Error:404	Update the logged user's profile
POST	vehicles/insert	Succes:200 Error:404	Vehicle creation function
GET	vehicles/insert	Succes:200 Error:404	Vehicle creation form display
GET	vehicles/index	Succes:200 Error:404	Display a table with all the vehicles
POST	vehicles/{vehicle}	Succes:200 Error:404	Display the editable data of the vehicle
GET	vehicles/search	Succes:200 Error:404	Search a vehicle by its licence plate
POST	vehicles/{vehicle}	Succes:200 Error:404	Edit the data of target vehicle
GET	vehicles/info/{vehicles}	Succes:200 Error:404	Display detailed information about a vehicle

GET	vehicles/realtime/{vehicle}	Succes:200 Error:404	View that shows the location of a vehicle in a map
GET	vehicles/map/{vehicle}	Succes:200 Error:404	Vehicle's map view
POST	workers/insert	Succes:200 Error:404	Worker creator function
GET	worker/insert	Succes:200 Error:404	Worker creator function's view
GET	wokerk/index	Succes:200 Error:404	Display a table with all the workers
GET	worker/info/{worker}	Succes:200 Error:404	Display detailed information about a worker
POST	workers/{worker}	Succes:200 Error:404	Edit the data of target worker
DELETE	workers/delete/{worker:id}	Succes:200 Error:404	Delete a worker selected by its identifier
DELETE	vehicles/delete/{vehicle:id}	Succes:200 Error:404	Delete a vehicle selected by its identifier

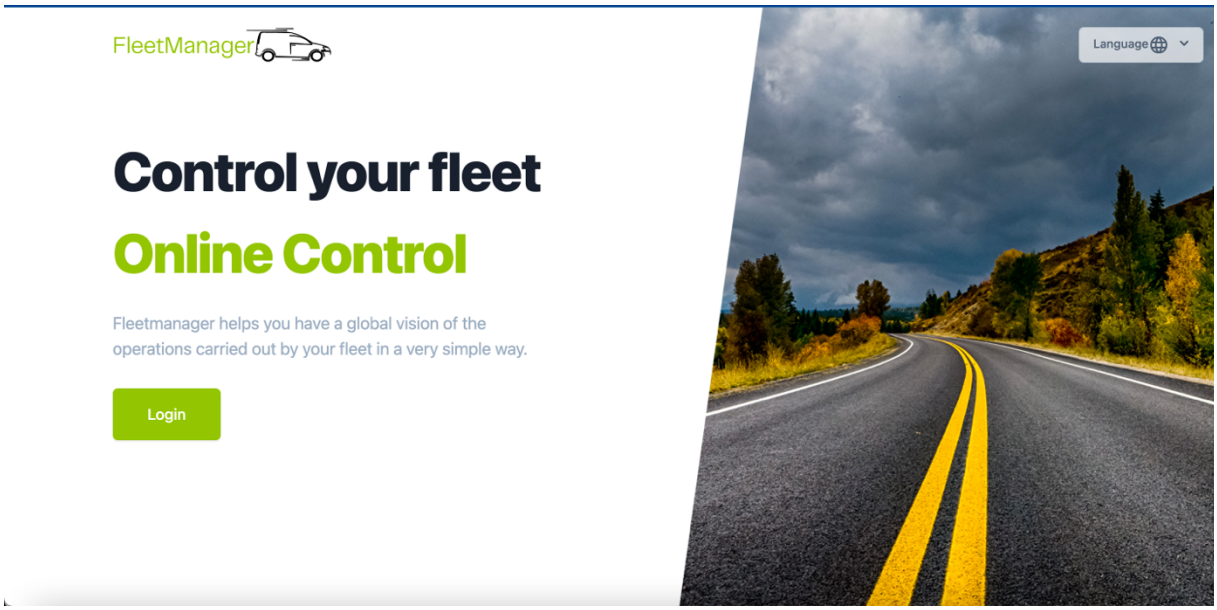


Figure 4: Main Page

In *Figure 4* we can see the web page's view where the user can select the language which he wants to work on and a button to access to our platform's login.

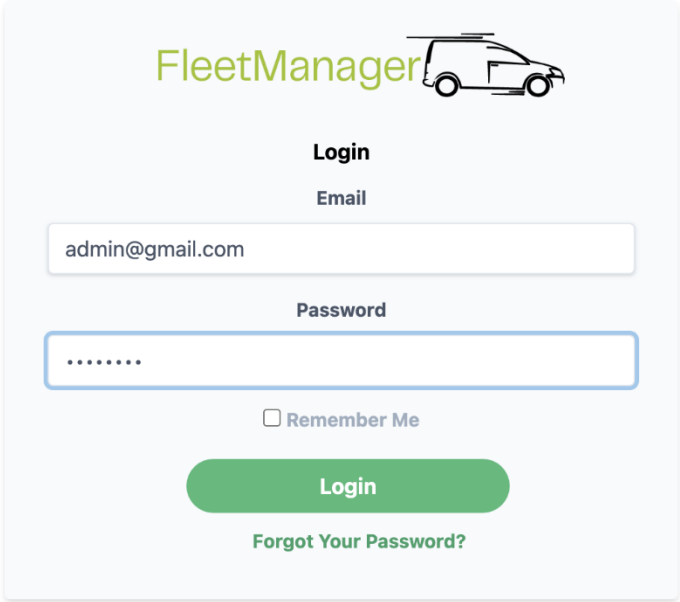


Figure 5:Login page

In *Figure 5* we can see the screen in which the user writes his credentials in order to access the platform.

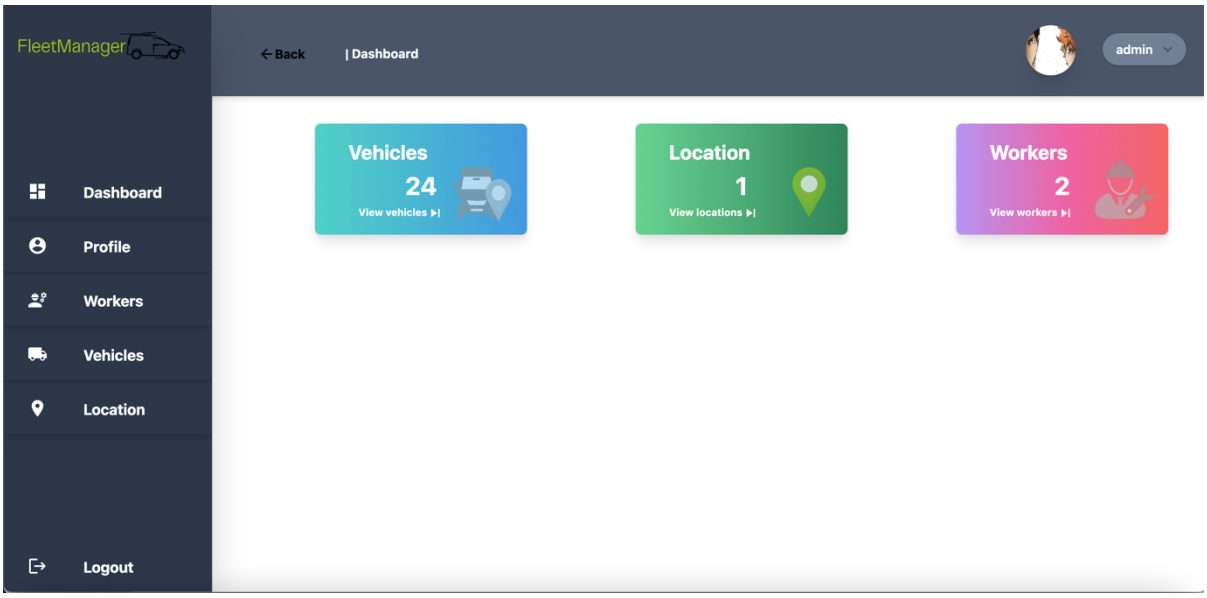


Figure 6: Dashboard page

Once the user has accessed, he is able to see a summarised information about the worker status and their locations, as shown in *Figure 6*.

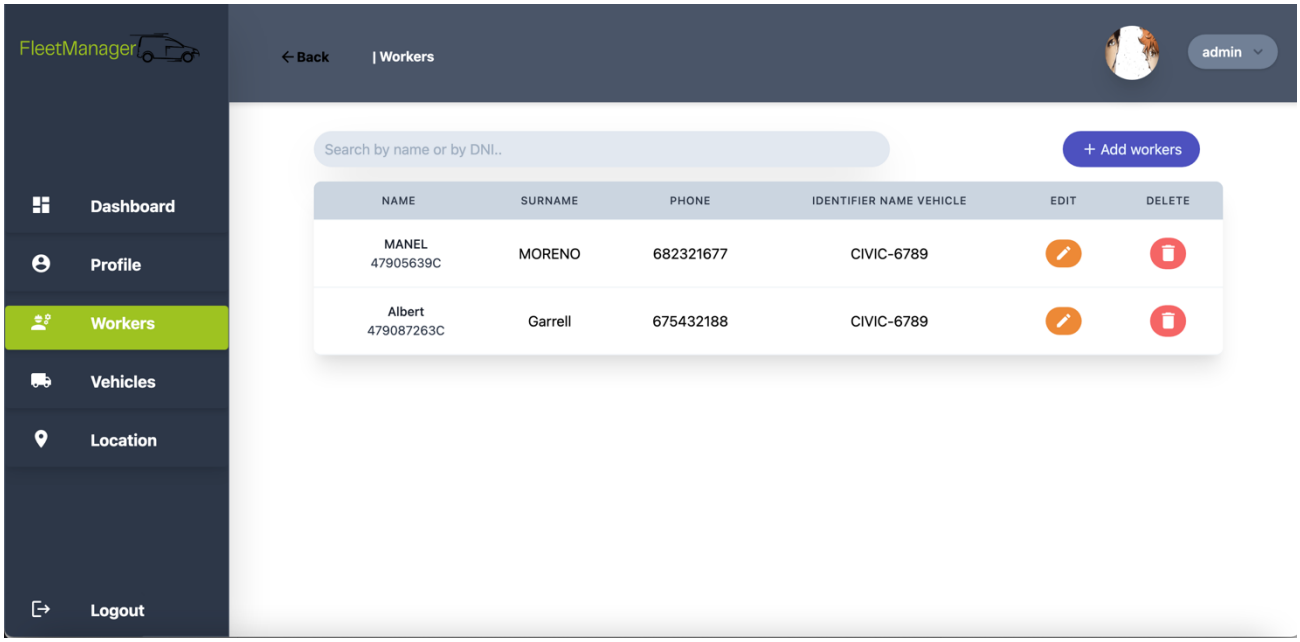


Figure 7: Worker's page

The user, once he has accessed to the worker's screen, he will see a list of the company's workers and a button to add new workers, as we can appreciate in *Figure 7*.

In the workers table we can also access to 2 other functionalities. A search engine to search by name or NIF and a button to edit a worker's information and, if it was the case, delete his data. Moreover, if the user wants to see a detailed view of the worker's file card, he can do so by clicking on the corresponding line.

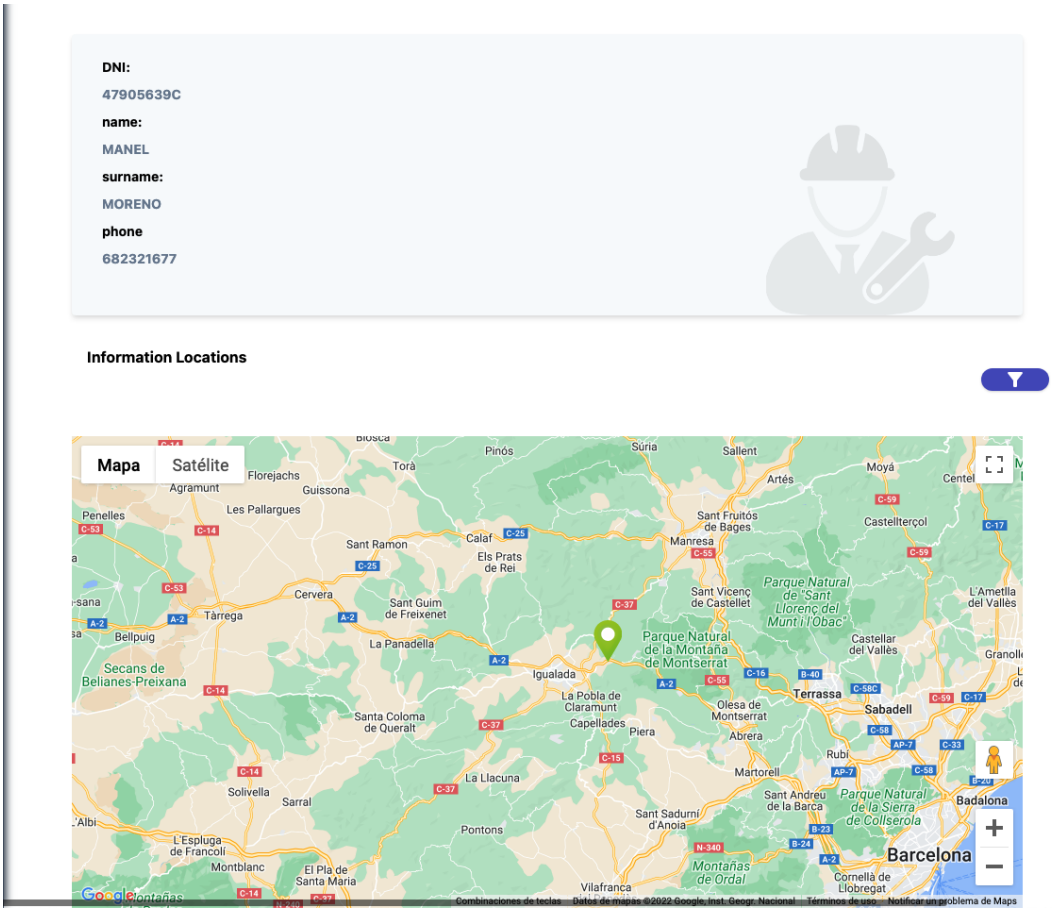


Figure 8: Worker location page

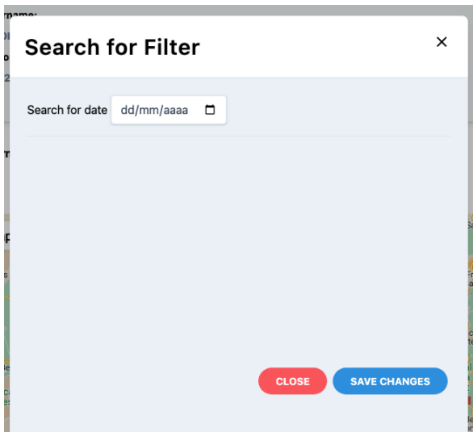


Figure 9: Worker location filter

Once the user accesses to the information about a worker he is able to see a summary with all the locations where his vehicle has travelled in a map, shown in *Figure 8*. We additionally have the functionality shown in *Figure 9* that allows us to sort by date to see the locations that the vehicle has been in during a concrete day.

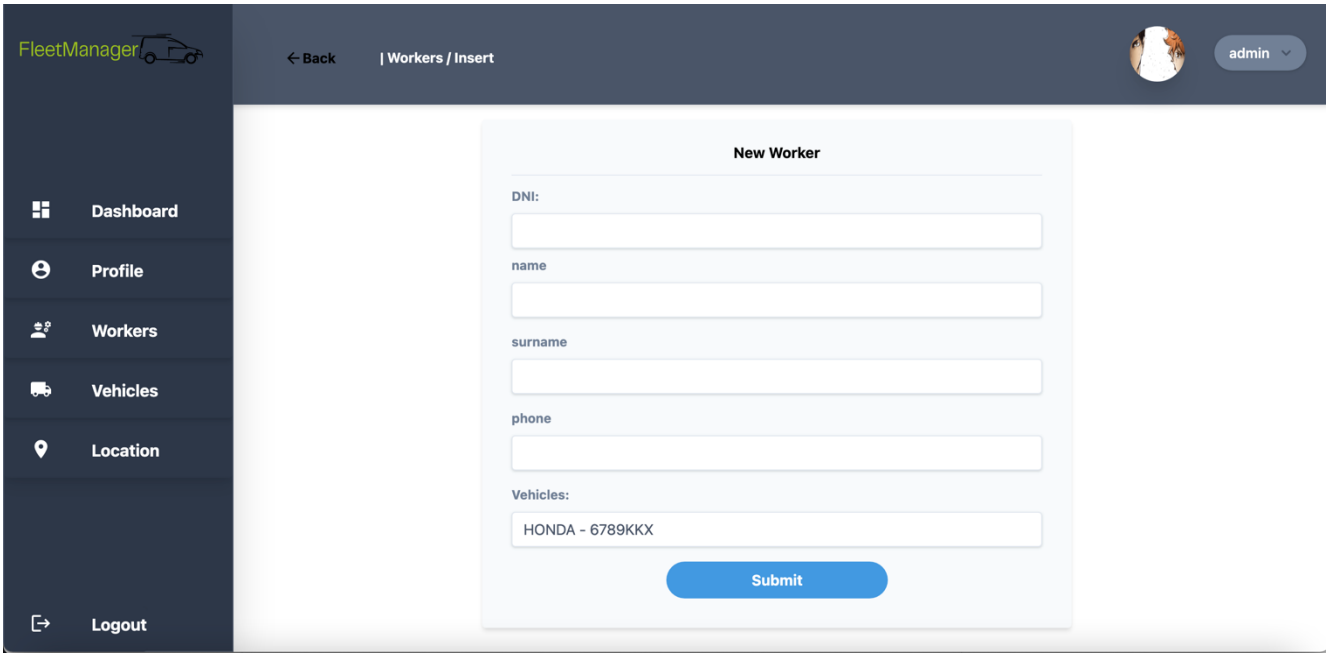


Figure 10: Worker creator questionnaire

As shown in *Figure 10*, the user must fill this questionnaire to be able to enrol a new user. DNI is a unique and specific field that prevents the creation of different workers with the same number. Vehicle assignation is optional, since it may happen that the company doesn't have an available vehicle at the moment.

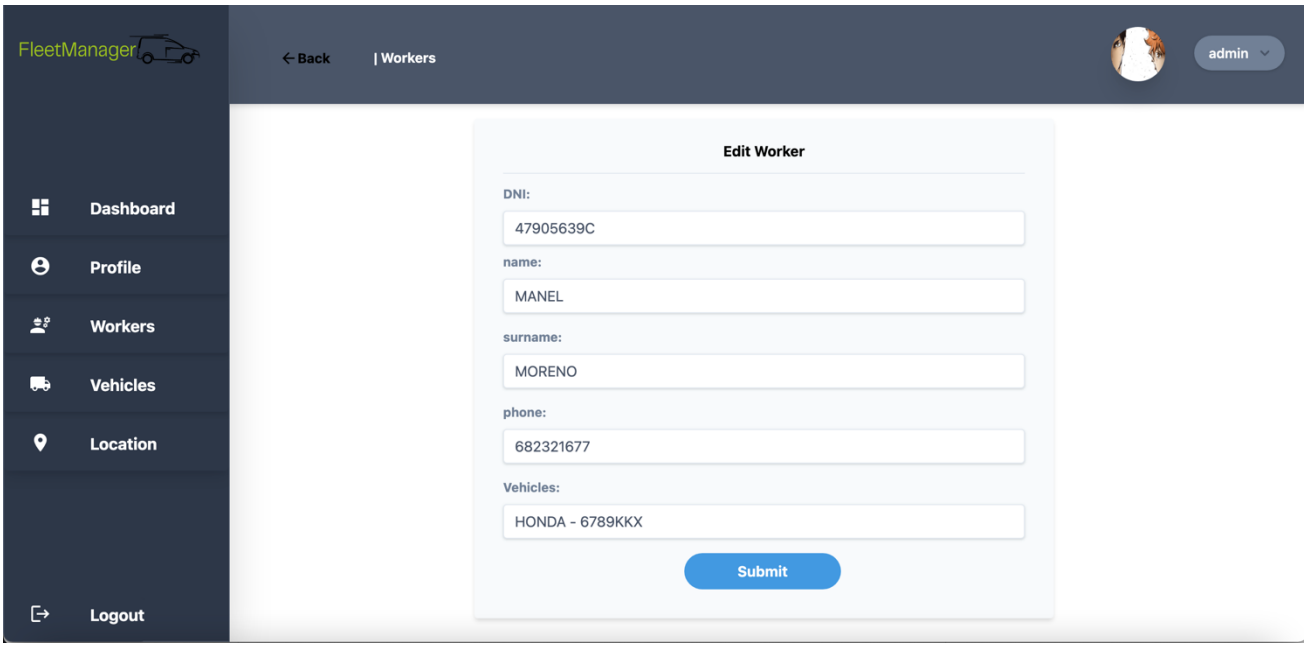


Figure 11: Worker data editing screen

If a user needs to edit a worker's information that is displayed in the table, he has to click the pencil icon, where he will access the view shown in *Figure 11* to edit it.

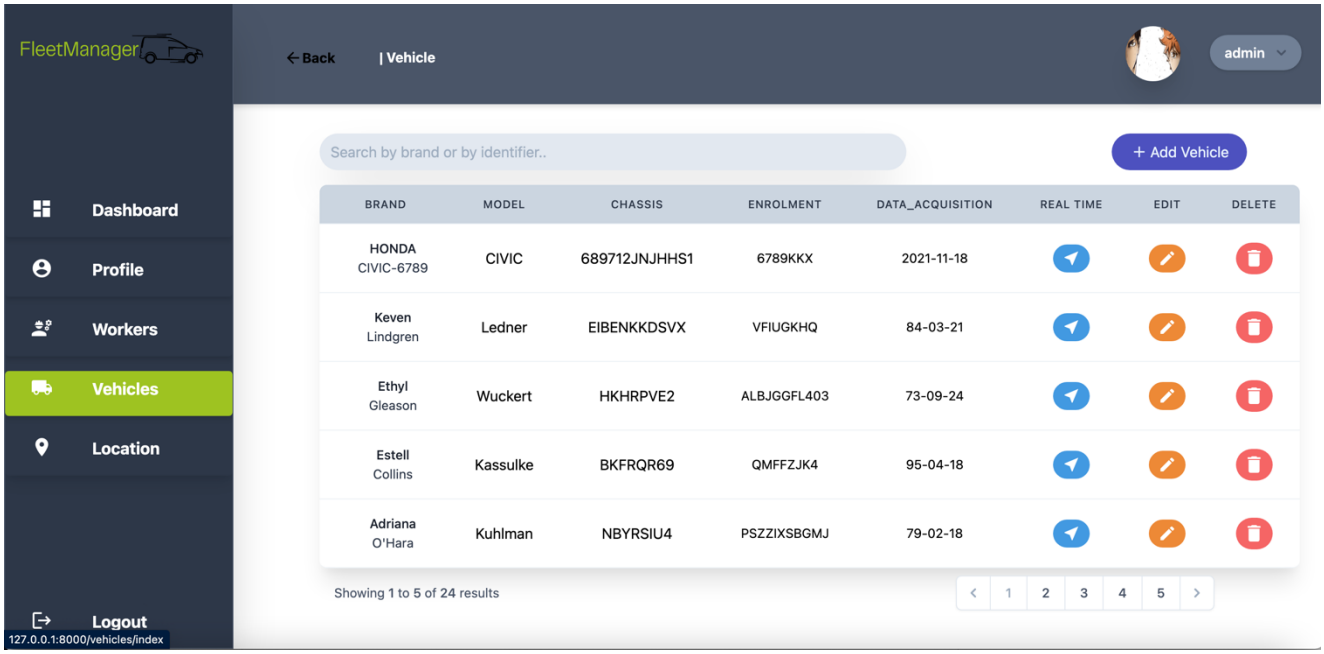


Figure 12: Vehicles Page

As shown in *Figure 12*, the user is able to access a table view with al the company's vehicles. It contains various functions such as a search engine to search by license plate or brand, a button to access its real time location, a button to edit a vehicle's data and another one to delete them.
Moreover, if the company obtains a new vehicle, it can add it to the database with the button next to the search engine

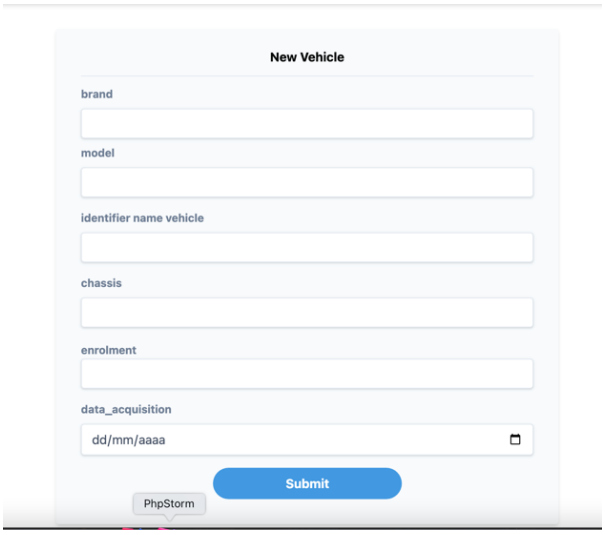


Figure 13: Vehicle creation questionnare

As we can see in *Figure 13*, the user must fill the form in order to introduce a new vehicle's data. There are compulsory fields such as the license plate number, the name of the vehicle and the chassis number.

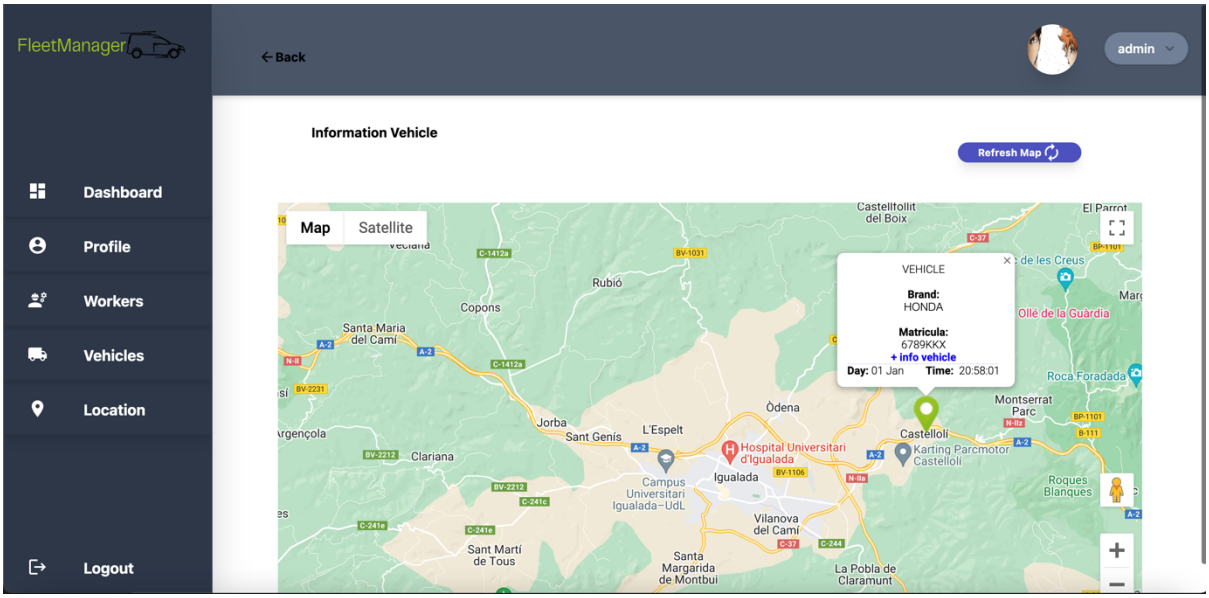


Figure 14: View of all worker's locations with their assigned vehicles

One of the functions of the table is to access a vehicle's real time location view. In the case shown in *Figure 14*, the user accesses to a worker's vehicle and shows its location being in Castellolí.

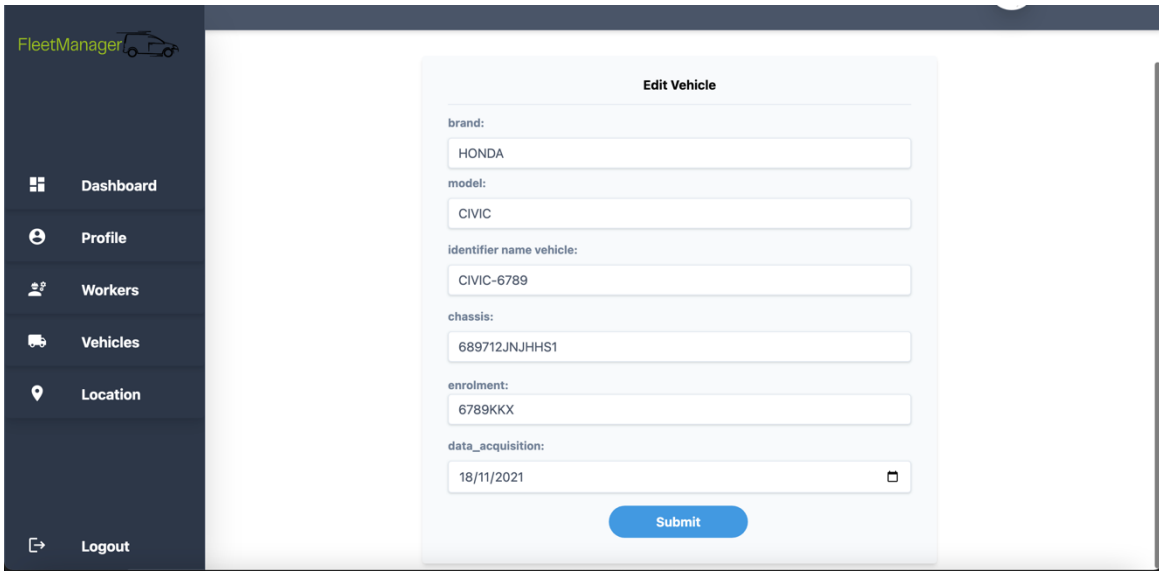
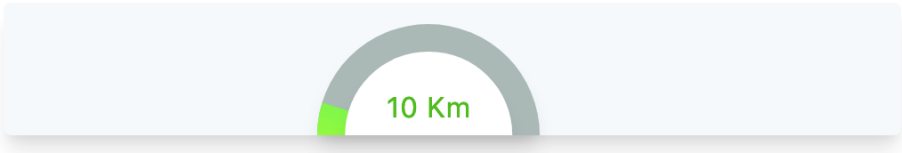


Figure 15: Vehicle data edition screen

As shown in *Figure 15*, the user will access this view trough the table specifying which vehicle he needs to edit and will enter the vehicle editing questionnaire.

Information Vehicle

The total kilometers of the vehicle



Last Locations

BRAND	MODEL	CHASSIS	LAST LOCATIONS
HONDA CIVIC-6789	CIVIC	689712JNJHHS1	2022-06-08 20:58:01
HONDA CIVIC-6789	CIVIC	689712JNJHHS1	2022-06-08 21:25:43
HONDA CIVIC-6789	CIVIC	689712JNJHHS1	2022-06-08 21:26:47

Figure 17: Detailed vehicle information screeen

All Location Vehicle for Map

Figure 16: Map with the routes of the vehicle filtered by date

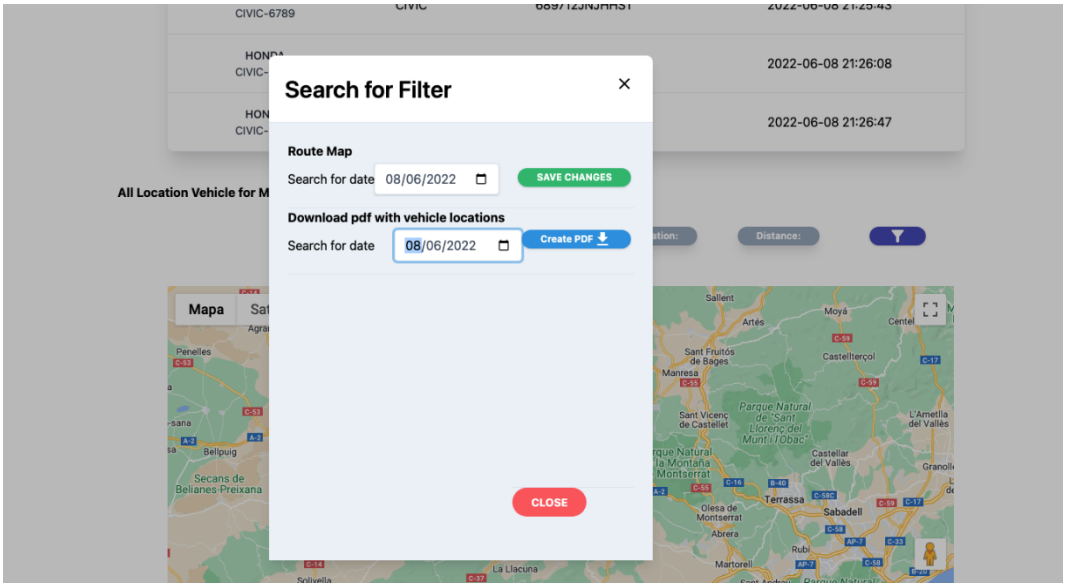


Figure 18: Modal where the user filters by date and can download the summary pdf

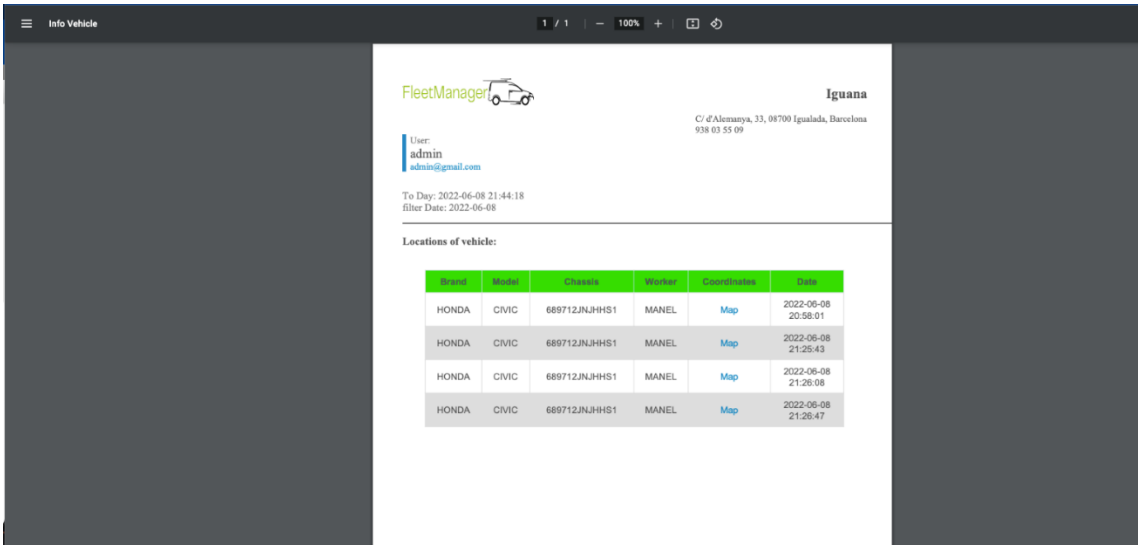


Figure 19: Pdf downloaded by the user with all the records of the coordinates of specific date

As shown in *Figure 16*, the user can access and see the information about a vehicle, including the data and its last coordinates registered, in addition to all the vehicles location in the map.

In *Figure 17*, we can see the filter button, which has two functionalities. As shown in *Figure 18*, the user is able to filter by date and see the whole vehicle's route during that day, its duration and the total number of kilometres made . Moreover, as shown in *Figure 19*, the user has access to the register of all this coordinates, being able to also filter by date and download a PDF with all this information.

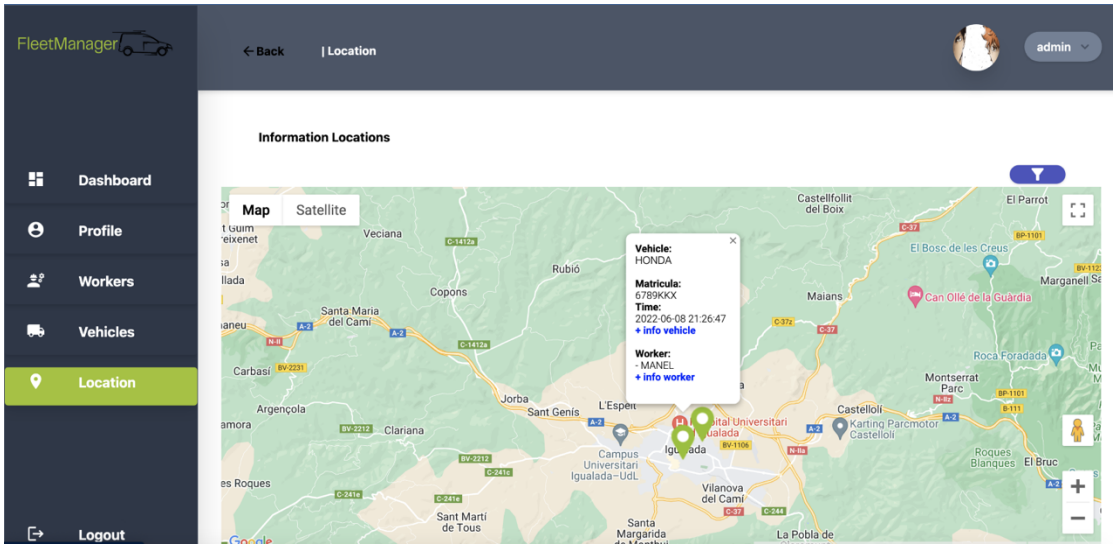


Figure 20: Locations Page

As seen on *Figure 20*, when the user accesses the locations views, he will be able to see the vehicle's last location registered on the map. It does also have a button that allows him to filter by worker and see his las location registered.
If the user clicks on the vehicle's location icon, he will be able to see a bit of information about the target vehicle, such as his licence plate number, the time that it has spent moving and the worker that is driving it. If he wants to obtain more information, the user is able to do so using any of the options previously mentioned.

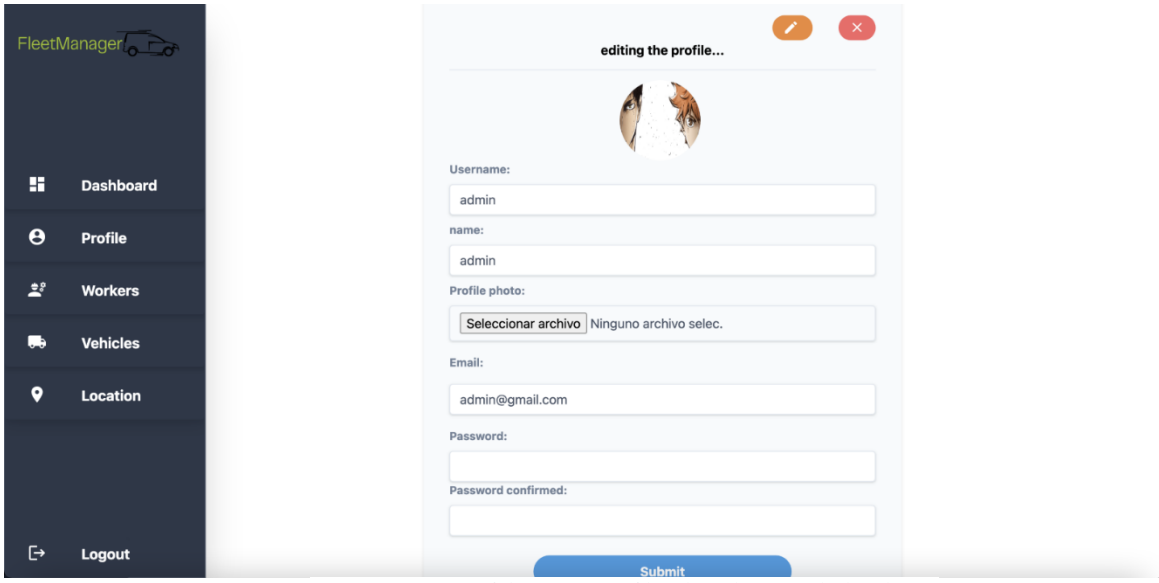


Figure 21: View of the user's profile where he can edit his data

Figure 21 shows the screen in which the user will be able to edit their own data, change his profile picture and his password.

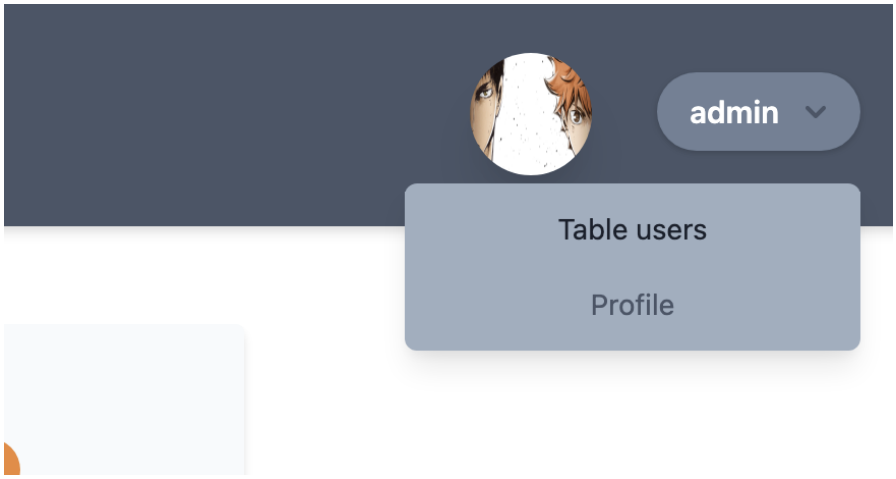


Figure 22: Administrator user menu

As shown in *Figure 22*, administrator users will also be able to access a functionality by any view that allows them to manage the users. If the user is not an administrator, he would only be able to edit the data on his own profile.

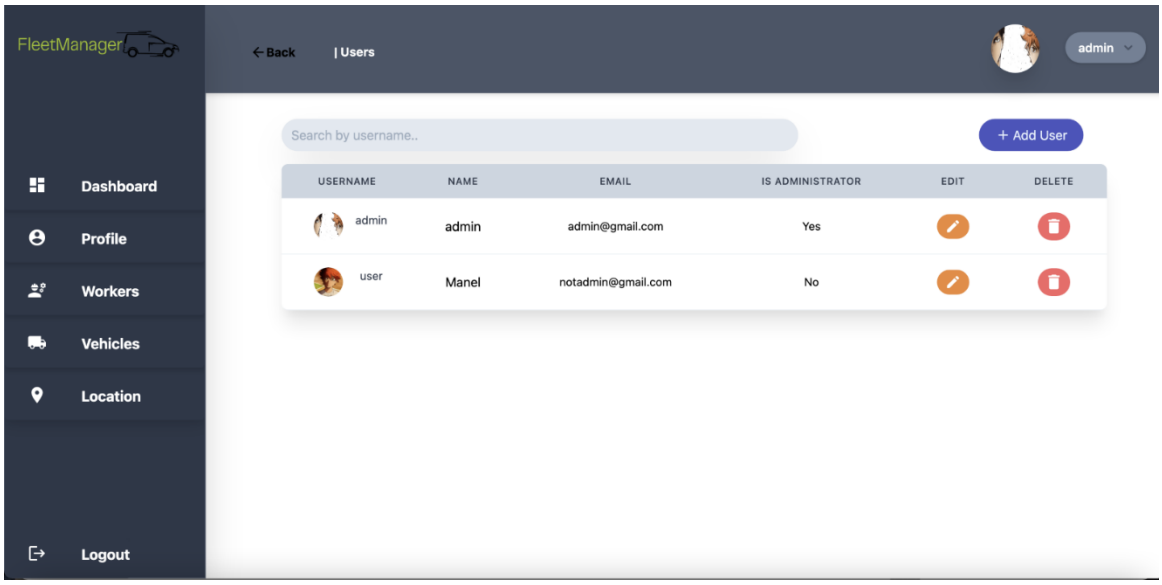


Figure 23: Administrator view where user management is located

Figure 23 shows the user management screen, where only administrators will be able to access into. Using this view, the administrator will be able to add new users and to assign roles to them. He will also be able to edit user's data and eliminate user profiles.

Mikrotic Configuration

To configure our Mikrotik devices, we assigned various tasks consisting of sim card, LTE internet and GPS configuration.
We developed a scrip programmed to run every two minutes, which sent the coordinates of a device through our API. We also installed a hardware which consisted in an external antenna that allowed us to receive the GPS coordinates more precisely.

In order to configure the Route OS Mikrotik device, we required WinBox, a tool that allowed us to configure both the GPS and the LTE of the sim card. The latter was needed in order to access the internet and make the proper requests to our API in the remote server.

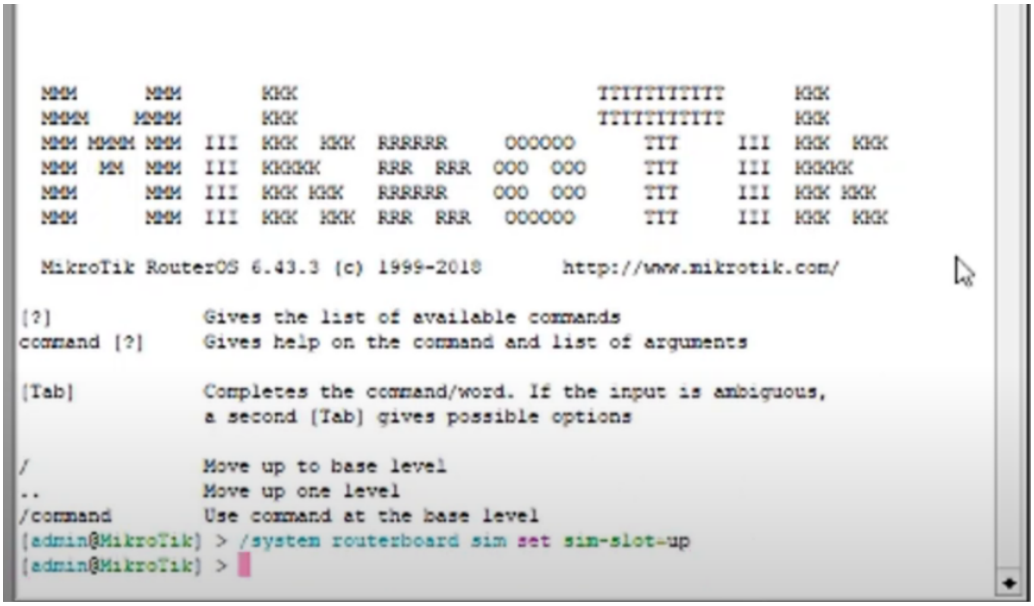


Figure 24: Winbox program terminal

In *Figure 24* we are able to see the terminal of the program where we can type the command "/system routerboard sim set sim-slot=up" to detect our sim card an configure it.

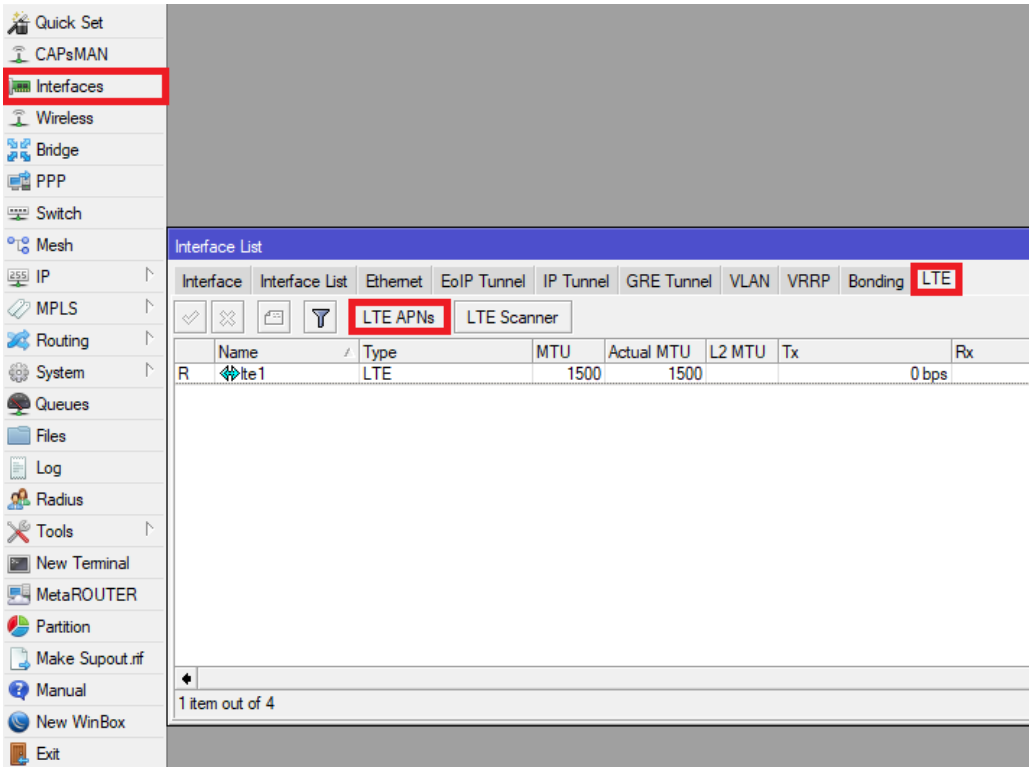


Figure 25: LTE activation showcase

In order to configure the sim card once it has been detected by the terminal, we must go to the view shown in *Figure 25*, in the subsection "interfaces", where we will configure the LTE and APNS to activate its internet.

Once we have internet access, we can do ping testing in our terminal. By doing this, we can test the sim card external requests to our server.

Following this, we configured the GPS. In order to activate this service, we needed to purchase an antenna that allowed us to obtain enough signal, since with the internal antenna we did not get enough communication with the satellite.

Finally, to conclude the device's configuration with all of its functionality, we programmed a script that sends to our API the coordinates linked with an identifier in order to identify the vehicle which they were coming from.

```
script
{
:global lat
:global lon
:global id 1;
/system gps monitor once do={
:set $lat $("latitude")
:set $lon $("longitude")

}
tool fetch mode=https url="https://fleetmanager.iguana.cat/api/locations/insert" port=443
http-method=post http-data=("{\"lat\":\"" . $lat . "\",\"lon\":\"" . $lon . "\",\"vehicle_id\":\"" .$id. "\"}")
http-header-field="content-type:application/json,Authorization: Bearer mjt7PWtWLUxljelGtPUvNsO7KSqi5e"
:put ("{\"lat\":\"" . $lat . "\",\"lon\":\"" . $lon . "\"}")
}
```

Figure 26: Mikrotik script to capture GPS coordinates able to be sent to the API

In *Figure 26* we can see the script where we store the coordinates as well as the GPS' configuration. We do also send the vehicle's identifier and a security token that allows us to send requests to our server.

Once we verified that the script was fully functional, we programmed a task that sent the vehicle's coordinates every two minutes to our API. This way, we were able to make a register with all the whereabouts of the vehicle during each day.

Deployment

Once the project's development is finished, including the software and hardware, it is necessary to upload it with a public, own domain in order to be able to make requests to the API by the Mikrotik device.

In order to obtain a public domain, there is a process that we need to consider:

1. The first step is to download Apache2[10] with the version 2.4.43 to have a web server and upload our project
2. Then we need to download MySQL with the 5.5 version, in addition to phpmyadmin with the 5.2 version in order to keep and manage the databases and tables
3. The third step is to install PHP in the 8.0 version, since it is needed to use the latest functionalities that our framework provides us with, and makes us avoid incompatibility problems
4. The next step is to download the 2.0 version of Composer[11], which is a PHP packet library that acts as a standard for managing, downloading and installing dependencies and libraries
5. Then we need to download our project FleetManager developed with Laravel
6. Configure the ".env" archive of our project so it points to our server database.
7. Execute the php command "artisan migrate" in order to migrate all the tables of our project to the server's database
8. Configure the archive in the folder sites-available to point to our project's folder, with the domain fleetManager.iguana.cat
9. Reboot Apache
10. It is advisable to install a ssl certificate. Encrypting the platform's web requests keeps the users secure when using our application.

Results

Functionality and final product

Following the SCRUM method we have continuously proposed goals, assigned weekly in the form of new tasks that took us closer to establish our objectives. This method is particularly useful for proportioning a coherent order for developing the functionalities of an application.

This project contains all the required functionalities needed to solve the problems of Iguana SL related to vehicle management. FleetManager allows us to manage the workers and to assign a vehicle to them, in addition to the management, creation and elimination of the users that can sign up in it. The most relevant functionality is the vehicle's position control using the GPS device, which allows us to locate them through our API.

Some of the other advantages that this application brings us are the various filters, which provide us with a lot of functionality, and the capacity to make daily summaries of the routes which a vehicle has taken, which increases the control and efficiency of the workers.

Product's impact to the company

Our application brings a series of advantages to the company, in relation to the information that it provides and the processing that we do on this data through our program. In the list further on we have summarised the main strengths of our application, and the problems that we seek to resolve.

FleetManager's main advantages:

- Vehicle fuel use and ubication control
- Worker control and its location in the assigned task
- Daily vehicle route summary
- Vehicle and worker's data management
- Vehicle's location display in a map
- Vehicle cost optimization
- Management of the users of the platform
- Ideal route study to increase the worker's efficacy

Main disadvantages of not using FleetManager:

- Being unable to control vehicle's fuel use and location
- Having no control over if a worker has arrived at the target location and when has he finished his service

- Not having a daily summary of the cost associated with a worker and it's vehicle
- Not having vehicle data management
- Being unable to optimize vehicle cost due to the lack of data
- Requiring additional staff to manage time of starting and finishing a worker's service
- Not knowing the location of a vehicle after a robbery
- Not knowing the location of a vehicle after an accident

Conclusions

The knowledge obtained during the development of this project consisted in the learning of a new framework such as Laravel, that enables us to manage relational database models. Combining that with the Mikrotik's configuration and Google Maps' integration, it brings a lot of advantages for companies that face the same lack of vehicle control problems.

The service presented in this article brings a lot of potential since it consists of a web application where the user can manage all the data related to the vehicle, the worker, and their locations. It is important to highlight that FleetManager is able to show real-time vehicle location.

For future projects, we must take further into account the organization and the priority level of the various tasks and emphasize the importance of defining the goals during the meetings, since it greatly effects development.

Furthermore, this project contains all the required functionalities that it has been asked for. The platform's visible part is currently designed for computer screen resolution, but it would be advisable to develop it for other kinds of resolutions.
It is also advisable to develop a continuous request signal to the database to display the information on the map in a dynamic manner, since it would positively effect user experience.

About the database, it would be wise to consider working with a non-relational model, since it offers greater potency with processing the different objects and can bring in a bigger and more dynamic storage potential.

Bibliography

1: Laravel - The PHP Framework For Web Artisans. (2022). Retrieved 30 June 2022, from https://laravel.com/

2: PHP: Hypertext Preprocessor. (2022). Retrieved 2 July 2022, from https://www.php.net/

3: MikroTik. (2022). Retrieved 30 June 2022, from https://mikrotik.com/

4: Winbox. (2022). Retrieved 30 June 2022, from https://mikrotik.com/download

5: Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (2022). Retrieved 2 July 2022, from https://tailwindcss.com/

6: Boostrap, B. (2022). Bootstrap. Retrieved 2 July 2022, from https://getbootstrap.com/

7: MySQL :: MySQL Downloads. (2022). Retrieved 2 July 2022, from https://www.mysql.com/downloads/

8: phpMyAdmin, p. (2022). phpMyAdmin. Retrieved 2 July 2022, from https://www.phpmyadmin.net/

9: Google Maps Platform | Google Developers. (2022). Retrieved 30 June 2022, from https://developers.google.com/maps

10: Apache, D. (2022). The Apache HTTP Server Project. Retrieved 30 June 2022, from https://httpd.apache.org/

11: Composer. (2022). Retrieved 2 July 2022, from https://getcomposer.org/

12: Git. (2022). Retrieved 2 July 2022, from https://git-scm.com/