EcoDrive Planner eMob (EDP-eMob): Software system for route optimization and sustainable driving modes

Version 1.0 - User manual

Patricia Ruiz, Bernabé Dorronsoro, Juan Carlos de la Torre, Javier Jareño, Fernando Muriano

Enero 2026

Contents

1 Description of the program, language, and operating environment **2**
1.1 Brief description of the computer program 2
1.2 Programming language 2
1.3 Operating environment 2

2 Starting and stopping services **3**

3 Repository structure **3**

4 Infrastructure **5**

5 User Manual **6**

1 Description of the program, language, and operating environment

1.1 Brief description of the computer program

EcoDrive Planner eMob (EDP-eMob) is a modular web software platform for the planning, management, and visualization of sustainable routes, oriented towards electric, hybrid, and plug-in hybrid vehicles. It allows (i) managing vehicle profiles with physical and energy parameters, (ii) defining and storing routes via an interactive map (adding points manually or by clicking on the map), and (iii) enriching routes with geographic information (e.g., elevation profiles) for analysis and reuse. Additionally, it includes a sustainable driving solution generation/consultation module that, based on a route and a vehicle, executes an optimization model to propose operation strategies (e.g., distribution of electric/combustion usage) minimizing CO_2 emissions, and presents results summarized, detailed by segment, and on the map.

1.2 Programming language

The system is implemented with a decoupled frontend–backend architecture: the backend is developed in **Python** using **FastAPI** (REST API and model execution engine) and persistence is handled in **MongoDB**; the frontend is developed in **JavaScript** with **React** (supported by Vite and map libraries like Leaflet). External services are integrated for route and elevation calculation, such as **OSRM** and **OpenTopoData**.

1.3 Operating environment

EDP-eMob operates as a web application: the frontend runs in a browser and communicates with the backend via HTTP/HTTPS requests to a REST API. The backend is deployed on a server that centralizes business logic, data access, and model execution, connecting to a MongoDB instance. As environment requirements, the backend requires **Python** $\geq$ **3.9** and **MongoDB**, and the frontend requires **Node.js** $\geq$ **18**; by its nature, it is compatible with typical deployments on server operating systems (e.g., Linux) and multi-platform access from the client via browser. To facilitate its use, the system is deployed in Docker.

2 Starting and stopping services

Start the services:

```
docker-compose up --build
```

This command builds the images and brings up the following services:

- MongoDB on `localhost:27017`
- Backend on `http://localhost:8000`
- Frontend on `http://localhost:5173`

Stop the services:

```
docker-compose down
```

3 Repository structure

```
.
|-- docker-compose.yml
|-- Dockerfile.backend
|-- Dockerfile.frontend
|-- nginx.conf
|
|-- backend/
|   |-- main.py
|
|   |-- requirements.txt
|   `-- model/
|       |-- 000_main_algoritmo.py
|       |-- CustomObservers.py
|       |-- HybridTruckProblem_Demo.py
|       |-- ReadRoute.py
|       |-- Route.py
|       |-- Section.py
|       |-- Truck.py
|       `-- jmetal/
|
|           |-- __init__.py
|           |-- config.py
|           |-- algorithm/
```

```
|           |   |-- __init__.py
|           |   |-- multiobjective/
|           |   |-- singleobjective/
|           |   `-- test/
|           |-- core/
|
|           |-- lab/
|           |-- operator/
|           |-- problem/
|           `-- util/
|
|-- frontend/
|   |-- index.html
|   |-- package.json
|   |-- vite.config.js
|   |-- README.md
|   |-- public/
|
|   `-- src/
|       |-- App.jsx
|       |-- main.jsx
|       |-- api/
|       |   `-- api.js
|       |-- components/
|       |   |-- CrearSoluciones.jsx
|       |   |-- MapaInteractivo.jsx
|       |
|       |   |-- MapaSoluciones.jsx
|       |   |-- VehicleForm.jsx
|       |   |-- VehicleList.jsx
|       |   `-- VisualizadorSoluciones.jsx
|       `-- pages/
|           |-- Home.jsx
|           `-- Navbar.jsx
|
|   `-- mongodb/
    `-- init/
        `-- 01-schema.js
```

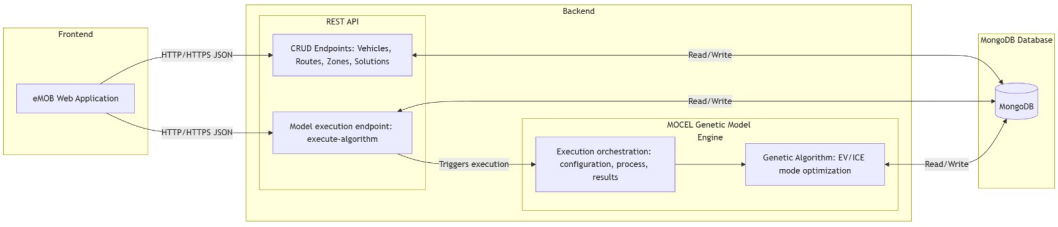

Figure 1: System Diagram

4 Infrastructure

Figure 1 shows the system interaction diagram.

Frontend (Web UI) The frontend corresponds to the eMOB web application used by the end user to manage vehicles, create/visualize routes on the map, and consult or generate solutions. Its responsibility is exclusively presentation and interaction: it collects user input and makes HTTP/HTTPS requests to the backend REST API. The frontend does not connect directly to the database nor does it execute heavy logic; even solution generation is initiated by invoking a specific backend endpoint, keeping all business and calculation logic centralized on the server.

Backend (REST API + Model Engine) The backend is divided into two functional blocks. On one hand, the REST API exposes the necessary endpoints for the application's operation (CRUD operations and queries on vehicles, routes, zones, and solutions), validating data and acting as the single access layer to persistent information. On the other hand, the backend incorporates the model execution engine, responsible for launching the optimization algorithm (in this case, a MOCell) when the user requests it via a dedicated endpoint (e.g., "Execute Algorithm"). This endpoint orchestrates the execution: it retrieves the necessary data (vehicle, route, configuration) from the

database, runs the model, and stores the results so they are available in the solutions view. Both the API and the model engine can read from and write to MongoDB, whether to obtain input data or to persist generated solutions and metrics.

Database (MongoDB) and NoSQL justification MongoDB acts as the central repository where vehicles and their characteristics, routes with their points/segments and metadata (elevation, average speed, stops, charging points, etc.), defined zones, and generated solutions with their statistics and segment details are stored. A non-relational database was chosen because the system works with flexible and semi-structured structures: a route may have a variable number of points, segments may incorporate additional attributes, and model results may grow in complexity as new metrics are added. In this context, a rigid relational schema would hinder evolution and force frequent migrations; in contrast, MongoDB allows adding or modifying parameters for vehicles, routes, or solutions agilely, facilitating adaptation to future models and new requirements without breaking the rest of the backend's functionality.

Adaptability to other models The architecture is designed to be totally adaptable: the system can replace the genetic model with any other algorithm by modifying the model component executed by the backend endpoint. To maintain compatibility with the application, the new model must respect the expected data structure (input/output) and persist results in MongoDB following the format understood by the rest of the backend and used by the frontend to show summaries, segment details, and map visualization. This allows evolving the calculation engine without needing to redesign the interface or restructure the API core.

5 User Manual

Figure 2 shows the eMOB Home screen, which is the starting point of the application and can be accessed at any time by clicking the "eMOB" logo in the top bar. In the center, a welcome message and a brief description of eMOB as a mobility management platform for managing vehicles, routes, and optimized solutions are displayed. Below appear three buttons (equivalent to the Navbar options) that give access to the main functions: Vehicle Management, where the user can register new vehicles or edit existing ones; Interactive Map, an interactive map to create or visualize routes; and Solutions Generator, which allows consulting previously calculated solutions or generating new ones

from routes, determining via models when a hybrid vehicle should operate in electric mode or combustion mode to minimize CO_2 emissions.

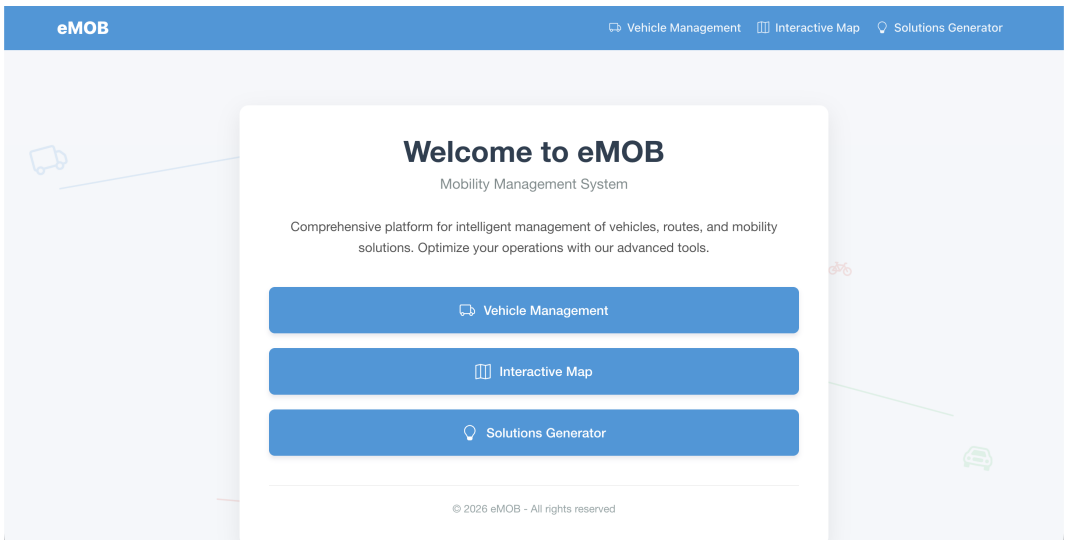

Figure 2: Home screen

Figure 3 shows the Vehicle Management section where the user can register a new vehicle by entering the necessary parameters so the system can model its on-road behavior. The screen shows an “Add New Vehicle” form where identification and typology data (Vehicle ID and Type) are completed, as well as physical and energy characteristics such as battery capacity (kWh), weight (kg), frontal area (m²), combustion engine efficiency (ICE Efficiency), electric system efficiency (EV Efficiency), and maximum powers in combustion and electric modes (Max ICE Power and Max EV Power, in kW). Additionally, extra properties can be added in Additional Characteristics by indicating an attribute name and its value and clicking Add to incorporate it. Once the fields are completed, the registration is confirmed with the Create Vehicle button, leaving the vehicle registered in the application for use in the rest of the modules.

At the bottom of the Vehicle Management module is the Vehicle List section (Figure 4), where all vehicles already registered in the system appear as cards with their ID and type (e.g., car - 1 or truck - 2) and a summary of their main characteristics (battery, weight, frontal area, efficiencies, and powers). Above the list is a search field “Search by ID or Type” that allows quickly filtering by identifier or vehicle type. On each card, the user has three actions: Edit, to modify the vehicle’s characteristics; Solutions, to access solutions associated with that vehicle; and Delete, to remove it from the system. Clicking Edit opens a popup window with the editing form, where the

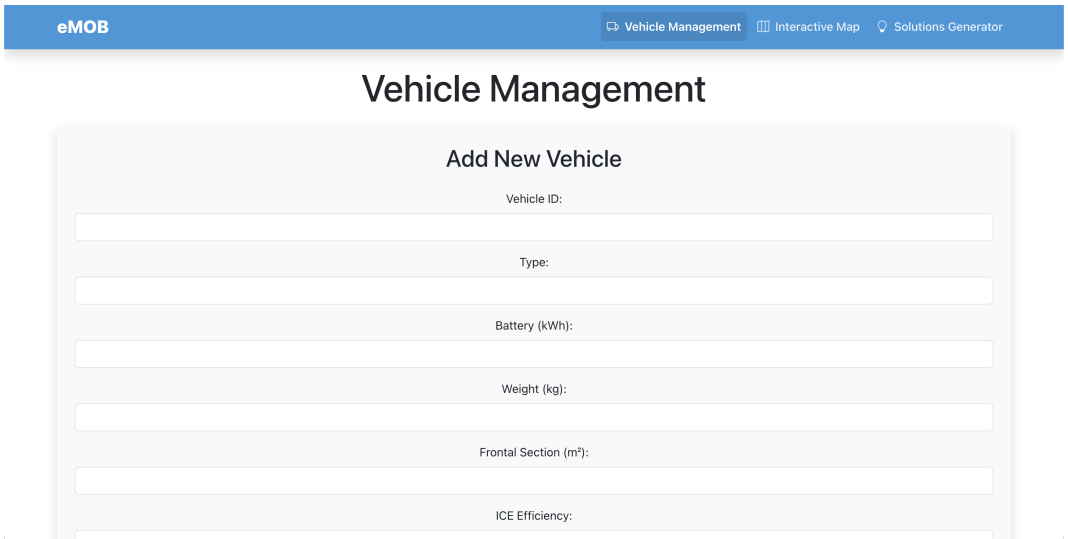

Figure 3: Vehicle management

same vehicle parameters (type, battery, weight, frontal area, efficiencies, and powers) can be updated, and the window can be closed with the X icon when finished.

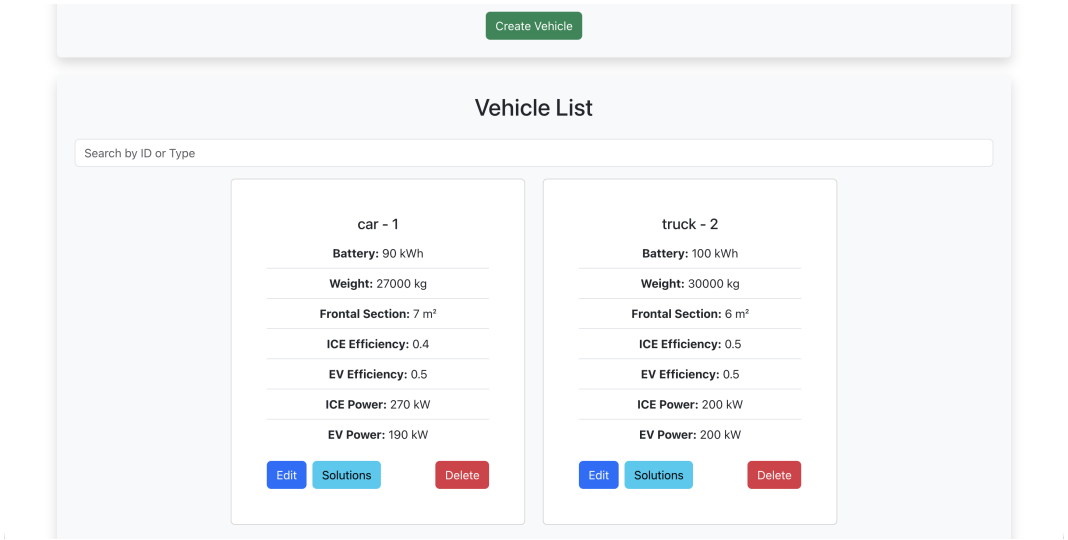

Figure 4: Vehicle list

In the Interactive Map section, the user has a map where they can create and manage routes from geographic points. The points that will make up the route can be added in two ways: via the side panel Add point, manually entering latitude and longitude and clicking Add point, or directly by clicking

on the map with the cursor to mark the location. At the top are the main actions: Calculate Route to generate the optimal route from the selected points (being able to include as many intermediate points as needed), Clear Points to clear the selection, and Stored Routes to consult saved routes; additionally, there are options to define and delete zones (buttons Define ZEZ, Define REZ, and Delete Zone). Everything described above can be observed in Figure 5.

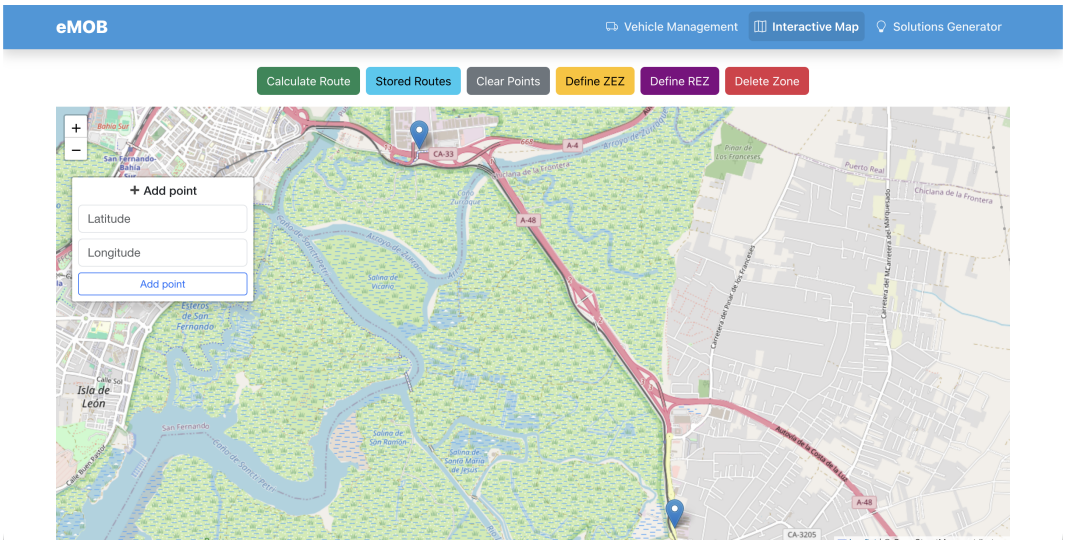

Figure 5: Establishing the route by selecting 2 points.

When calculating the route, the system traces the path and obtains relevant information for the models, such as elevation and average speed at the points/segments, showing a status notice while processing ("Calculating route…"). Once generated, the user can select points on the route to open a detail view with node data (coordinates, elevation, etc.) and provide additional context to the model by indicating if that point is a stop or a charging point (Figure 6).

Finally, with Store Route, the route is saved in the system: a name is requested to identify it and, upon confirmation, the saving status is shown ("Storing route…"), making it available for reuse later. In the Solutions Generator section, the user can generate solutions for a specific route and vehicle (Figure 7). The main screen shows a list of available routes in the system, with a search by name and a Select button to choose the route to work on (e.g., straight Cadiz), plus two access points: Generate Solutions and View Generated Solutions. Selecting a route opens a configuration window for the genetic algorithm (e.g. MOCell), where the user must choose the vehicle with which the model will be executed and adjust execution and search pa-

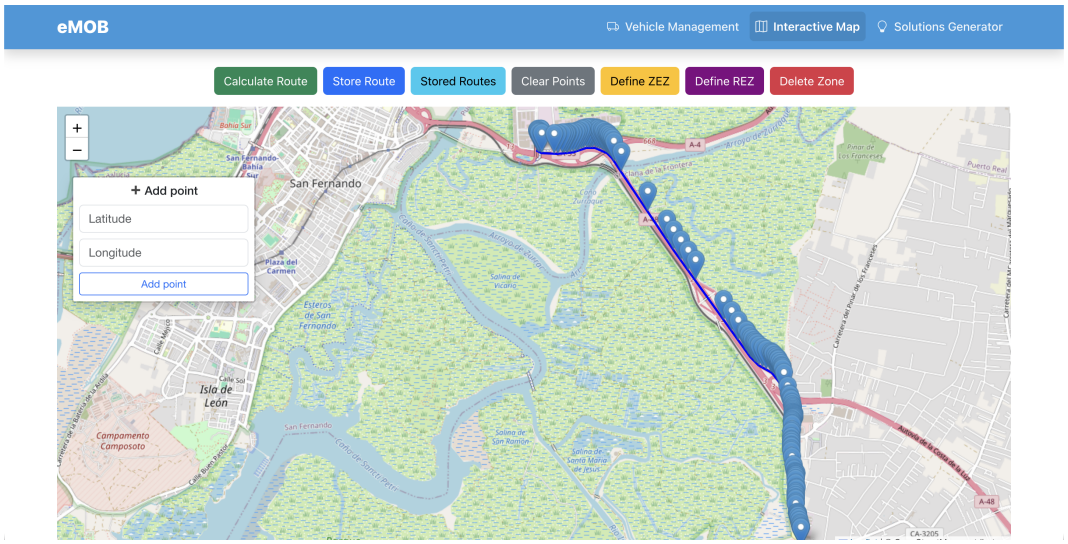

Figure 6: Map showing the calculated route.

rameters (e.g., Max Evaluations, Population Size, Offspring Size, Crossover Probability, Neighborhood Size, and the option Take stops into account). It also includes an advanced configuration with Process ID and Results Path, which indicate the process identifier and the server path where results will be stored in “RAW” format if access is available. Once the values are reviewed, the user can launch the calculation by clicking Execute Algorithm (or cancel with Cancel).

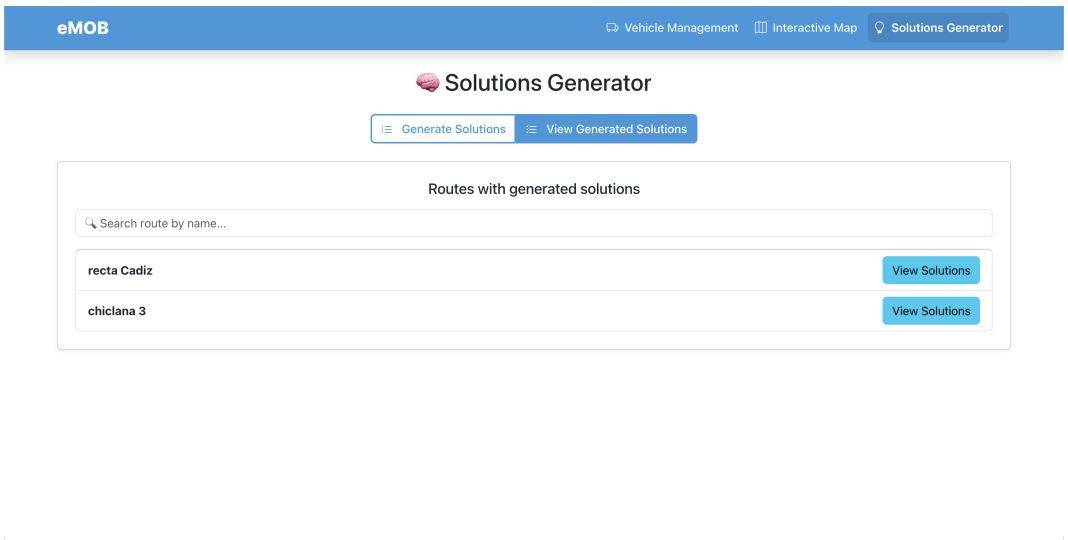

Figure 7: Solutions generator.

In the View Generated Solutions tab of the Solutions Generator module, the user can consult solutions already calculated for system routes: first, a list of “Routes with generated solutions” appears with a search by name and the View Solutions button to access the solutions for a specific route (e.g., straight Cadiz). Upon entering, the “Solutions for [route]” screen is shown (Figure 8) with a button to return (Return to solutions generator) and three views: in Summary, solutions are presented in card format with a summary of the percentage of use in electric mode and combustion mode, plus metrics like total emissions (kg CO) and energy consumed (kWh), with the ability to open details for each with View full details; in Full Detail, solutions are displayed individually and behavior can be analyzed segment by segment, indicating for each segment the mode (electric/combustion), distance, energy, emissions, and state of charge (Figure 9); and finally in Map Visualization, a map viewer is offered where the solution to show is selected and segments are visualized colored according to the driving mode, supported by a legend identifying electric, combustion, stops, and charging points (Figure 10).

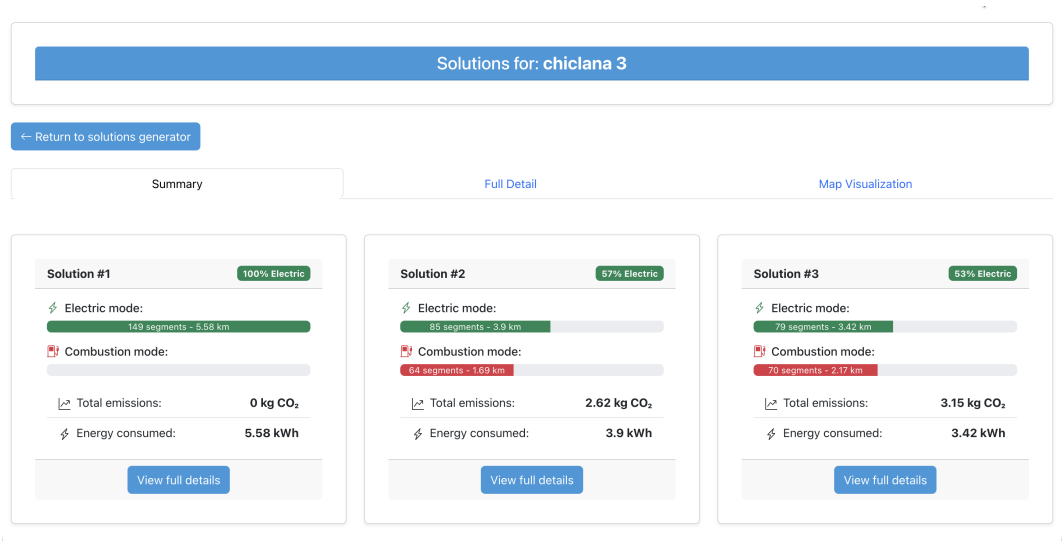

Figure 8: Summary of stored solutions.

← Return to solutions generator

Summary | Full Detail | Map Visualization

Solution #1 149 electric | 0 combustion | 0 kg CO_2

Solution #2 85 electric | 64 combustion | 2.62 kg CO_2

Segment	Mode	Distance	Energy	Emissions	Charge state
1	eléctrico	8 m	0.01 kWh	0.00 kg	90%
2	combustión	19 m	0.00 kWh	0.07 kg	90%
3	combustión	15 m	0.00 kWh	0.07 kg	90%
4	combustión	29 m	0.00 kWh	0.11 kg	90%
5	eléctrico	33 m	0.03 kWh	0.00 kg	89%
6	combustión	22 m	0.00 kWh	0.09 kg	89%
7	eléctrico	6 m	0.01 kWh	0.00 kg	89%
8	combustión	5 m	0.00 kWh	0.03 kg	89%
9	eléctrico	5 m	0.01 kWh	0.00 kg	89%
10	eléctrico	5 m	0.01 kWh	0.00 kg	89%

Figure 9: Detailed solution information.

Summary | Full Detail | Map Visualization

Select solution to display:

Solution #2

Figure 10: Solution displayed on the map.