El Camino Planner

*A full-stack web application developed by **András Dömötör***
as part of the **Arkance Systems** application process.

---

1. User Documentation

1.1 Overview

El Camino Planner is a web application that allows users to plan and manage their El Camino route.
Users can view, add, edit, and delete route stops directly on an interactive map.

Main Features

- Add a new stop
- Edit an existing stop
- Delete a stop
- Move a stop's position by clicking on the map

1.2 System Requirements

Required Software

- **Node.js**
- **PostgreSQL**
- **Web browser** (Google Chrome, Firefox, Edge, etc.)

1.3 Installation Guide

Step 1 — Clone the Repository

```
git clone https://github.com/domotorandras/camino-app.git
cd camino-app
```

Step 2 — Create the PostgreSQL Database

Open your PostgreSQL terminal (`psql`) and run:

```
CREATE DATABASE camino_app;
\c camino_app

CREATE TABLE coordinates (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    latitude DECIMAL(10, 6),
```

```
    longitude DECIMAL(10, 6),
    description TEXT
);
```

Step 3 — Fill the Database with Example Data

```
INSERT INTO coordinates (name, latitude, longitude, description) VALUES
('Porto', 41.1579, -8.6291, 'Starting point of the Camino Portugués'),
('Barcelos', 41.5317, -8.6184, 'Known for its iconic rooster symbol'),
('Ponte de Lima', 41.7685, -8.5839, 'Charming riverside town with Roman
bridge'),
('Valença', 42.0317, -8.6455, 'Border town with historic fortress
overlooking Spain'),
('Tui', 42.0476, -8.6444, 'First Spanish town on the route'),
('Porriño', 42.1617, -8.6197, 'Industrial town surrounded by forests'),
('Redondela', 42.2830, -8.6090, 'Town where coastal and central routes
meet'),
('Pontevedra', 42.4310, -8.6444, 'Beautiful historic center and pilgrimage
hub'),
('Caldas de Reis', 42.6043, -8.6421, 'Famous for its thermal waters and
calm atmosphere'),
('Santiago de Compostela', 42.8806, -8.5456, 'Final destination — home of
the Cathedral of Santiago');
```

Step 4 — Set Up the Environment File

Create a `.env` file inside your **server** folder and fill it with your PostgreSQL credentials:

```
PGUSER=postgres
PGPASSWORD=your_password
PGHOST=localhost
PGPORT=5432
PGDATABASE=camino_app
PORT=5050
```

Step 5 — Install Dependencies

```
cd backend
npm install
cd ../frontend
npm install
```

Step 6 — Run the Application

Start the backend:

```
cd backend
node index.js
```

Then start the frontend:

```
cd ../frontend
npm run dev
```

Step 7 — Open the App

Go to http://localhost:5173 in your browser.

1.4 User Interface Overview

Sidebar

- Displays all route stops.
- Includes buttons to **Add**, **Edit**, or **Delete** entries.

Map View

- Shows all stops on an interactive **Leaflet map**.
- In edit mode, you can click on the map to move a stop's position.

---

2. Developer Documentation

2.1 Project Overview

El Camino Planner is a full-stack JavaScript web application designed to manage and visualize route stops for the *Camino de Santiago* pilgrimage.
It follows a **client–server architecture**, where:

- The **frontend** (React + Vite) provides an interactive user interface with a map and sidebar.
- The **backend** (Node.js + Express) manages CRUD operations through REST API endpoints.
- The **database** (PostgreSQL) stores route stop data.

2.2 Technology Stack

Layer	Technology	Description
Frontend	**React (Vite)**	Modern SPA framework for responsive UI
Backend	**Node.js + Express.js**	REST API and server logic

Layer	Technology	Description
Database	**PostgreSQL**	Relational database for storing coordinates
Map	**Leaflet.js**	Interactive map rendering
Styling	**CSS / Tailwind**	Layout and responsive design
Communication	**Fetch API**	Handles requests between frontend and backend

---

2.3 Backend — Express API

GET /api/coordinates

Returns all saved coordinates.

```
GET http://127.0.0.1:5050/api/coordinates
```

Response Example:

```
[
  {
    "id": 1,
    "name": "Porto",
    "latitude": 41.1579,
    "longitude": -8.6291,
    "description": "Starting point of the Camino Portugués"
  }
]
```

POST /api/coordinates

Adds a new route stop to the database.

```
POST http://127.0.0.1:5050/api/coordinates
```

Request Body Example:

```
{
  "name": "New Stop",
  "latitude": 41.15,
  "longitude": -8.63,
  "description": "Newly added stop"
}
```

PUT /api/coordinates/:id

Updates an existing coordinate entry.

```
PUT http://127.0.0.1:5050/api/coordinates/3
```

Request Body Example:

```
{
  "name": "Updated Stop",
  "latitude": 41.20,
  "longitude": -8.65,
  "description": "Updated description"
}
```

DELETE /api/coordinates/:id

Deletes a route stop from the database.

```
DELETE http://127.0.0.1:5050/api/coordinates/3
```

Response:

```
{ "success": true }
```

2.4 Frontend — React Components

App.jsx

The main entry point that manages global state and logic.

- Loads data from backend.
- Handles `add`, `edit`, and `delete` operations.
- Passes data to `Sidebar` and `MapView`.

Key React hooks used:

```
useState, useEffect
```

Sidebar.jsx

Displays the list of route stops with:

- “Add” button: Creates a new stop at the last coordinate.
- “Edit” mode: Enables editing name, description, or position.
- “Delete” button: Removes a selected stop.

Communicates with the parent (`App.jsx`) via props:

```
handleAdd, handleSave, handleDelete, setSelected
```

MapView.jsx

Renders the **Leaflet** map and all route markers.

- When a marker is clicked: It becomes “selected.”
- In edit mode: Clicking on the map updates that marker’s coordinates.
- Uses `react-leaflet`’s `MapContainer`, `TileLayer`, and `Marker` components.

Example snippet:

```
<TileLayer
  url="https://{s}.tile.stamen.com/terrain/{z}/{x}/{y}.jpg"
  attribution='Map tiles by Stamen Design — OpenStreetMap contributors'
/>
```

2.5 Database Schema

Table: `coordinates`

Column	Type	Description
id	SERIAL PRIMARY KEY	Unique identifier
name	VARCHAR(100)	Stop name
latitude	DECIMAL(10,6)	Latitude coordinate
longitude	DECIMAL(10,6)	Longitude coordinate
description	TEXT	Stop description

2.6 Future Improvements

- Bug fixes (adding new point)
- User authentication (login & private routes)
- Route saving per user
- Image or media upload for stops
