QuakeWatch 🌍

Frontend application developed as part of the **Ubiwhere Frontend Recruitment Exercise**.

QuakeWatch is a React + TypeScript application that displays recent earthquake events on an interactive map, allowing authenticated users to explore and inspect detailed earthquake information.

Deployed version: QuakeWatch on Vercel

✨ Features

🔐 Authentication

- Login with provided credentials
- Token-based authentication (Bearer token)
- Authentication state persisted across page refresh
- Protected private routes
- Logout functionality

🗺️ Dashboard

- Interactive map built with **Leaflet**
- Earthquakes displayed as clickable markers
- Marker colors reflect earthquake magnitude
- Smooth map centering animation when selecting an earthquake
- Floating legend overlay

📊 Data Visualization

- Paginated list of earthquakes (4 earthquakes per page,**up to 5 pages**, as requested in the exercise)
- Table view with selectable rows
- Details panel with extended earthquake information
- Mobile-friendly UI inspired by Google Maps (bottom panels)

📱 Responsive Design

- Desktop and mobile layouts
- Floating panels and overlays
- Mobile bottom sheet for tables and interactions

🛠️ Tech Stack

- **React 19**
- **TypeScript**
- **Vite**
- **React Router**

- **React Query (@tanstack/react-query)**
- **Axios**
- **Leaflet / React-Leaflet**
- **Tailwind CSS + CSS Modules**

🚀 Getting Started

1. Clone the repository

```
git clone <repository-url>
cd ubiwhere-fe
```

2. Install dependencies

```
npm install
```

3. Environment variables

Create a `.env` file at the root of the project:

```
VITE_UBIWHERE_API_URL
```

> [!WARNING] This variable should point to the backend API URL provided for the exercise.

4. Run the project

```
npm run dev
```

The app will be available at:

```
http://localhost:5173
```

> [!NOTE] Your port may vary. Check the terminal output.

To access the app from another device on the same network:

```
npm run dev -- --host
```

Login

- **Email**
- **Password**

Project Structure (Simplified)

```
src/
├── api/                # Axios configuration
├── components/         # Reusable UI components
├── context/            # Authentication context
├── helpers/            # Map helpers
├── hooks/              # Custom hooks (e.g., useAuth, useEarthquakes)
├── pages/              # Application pages
├── routes/             # Private and public routes configuration
└── types/              # TypeScript types
```

Architecture Decisions

- React Context used for authentication to avoid prop drilling
- React Query used for server state (pagination, caching, retries)
- Leaflet chosen for open-source map rendering
- CSS Modules + Tailwind for scoped and maintainable styling
- Separation between MapView, DetailsPanel and Table to improve readability and reuse

---

Notes

- The project follows clean architecture and separation of responsibilities.
- Focus was placed on clarity, maintainability, and real-world usability.
- UI is functional and production-ready.

Author

José Pedro Antunes

Thank you for reviewing this project