EcoRoute - v0.5.1

Andrea Valentini

Last update: 13th April 2024

Contents

1 The project and project goals

The following is a description of the project problem and the goals to be achieved to complete the assignment. We have divided this section into three groups:

- The **preface** (or scenario) helps understand the environment to develop a sound software system.
- The **problem posed** section includes lists to emphasize the critical points.
- The **goals to achieve** by the assignment.

Note: We analyzed the **citizens' stakeholders** in this project.

Preface

Two urgent global concerns are environmental sustainability and climate change; because of air pollution and greenhouse gas emissions, transportation - especially urban commuting - contributes to worsening those issues.

Even today, urban areas are characterized by a heavy reliance on personal vehicles, which are seen as the most comfortable and efficient way of commuting, despite several studies showing that better alternatives exist in most cases.

Improving public transportation systems' efficiency can make them more appealing to daily commuters and is, therefore, a promising way to lessen environmental impact and, at the same time, to increase the overall quality of citizens' life (article).

Problem posed

The project "Eco-City Commute" (ECC) aims to create a comprehensive software system that makes public transportation within an urban area as easy and efficient as possible, promoting its adoption.

ECC receives data from sensors, deployed on public transport means, that provide:

- Information about their respective occupancy rates.
- Real-time information about public transit timetables.
- Information about bike and ride sharing, from specific services (think at ATM in Milano, BikeMi, Mobike, BlaBlaCar, ...).

Based on these pieces of information, ECC offers services to two types of stakeholders:

- **Citizens**: ECC offers a *mobile app* that allows citizens to input:
  - The origin (within the urban area);
  - The destination (within the urban area);
  - Eventually constraint, for example: they do not want to use a bike; they must arrive at destination within a certain timeframe.

The application takes the input and displays (output):

  - Environmentally friendly routes possibly combining different transportation means.

- **Urban area managers**: ECC offers to managers a dashboard through which they can visualize reports concerning the daily usage of the various available transportation means, their occupation rates and delays (if any).

Goals to achieve

We analyzed the **citizens' stakeholders**. So, the main goal was to develop a software system that offers citizens a mobile app to make public transportation within an urban area as easy and efficient as possible. Therefore, the document seeks to meet two objectives:

- Analyze the requirement aspects.
- Make a well-architecture design.

2 Requirement analysis

2.1 Relevant human and non-human actors

The only *human actor* is the citizen:

- **Citizen**. A user who uses the mobile application and enters the origin and destination of his journey. As the problem says, it can also add some constraints to the research.

Instead, there are several *non-human actors* that allow the user to search for some specific services or even a public transport timetable:

- **Sensor**. An electronic device installed on public transport vehicles that provides information about their occupancy.
- **PTT Server**. A Public Transit Timetable Server identifies the public transport company's server. The application makes some queries to this server to find out which public transport line is available at the time specified by the citizen actor.
- **BRS Server**. A Bike-Ride-Sharing Server identifies the server that the mobile application can query to get the information requested by the citizen actor. The BRS Server actor is as abstract as possible, because we want to be able to add as many services as we want.

  For example, a BRS server could be the BlaBlaCar site that the application queries to know if there is a car available for rent.

The Citizen is the primary actor because he is the stakeholder and the main user of the mobile application.

Instead, the non-human actors are all supporting actors because they provide a service to the application (e.g. the sensor offers the occupancy rate of public transport).

2.2 Use cases

The following use case diagram represents the EcoRoute application with a high level visualization. In the diagram it is possible to see the primary actor (citizen) that interacts (initiate) with the application and the supporting actors (Sensor, PTT Server, BRS Server) that support (participate) the application with the collection of the data from different services.

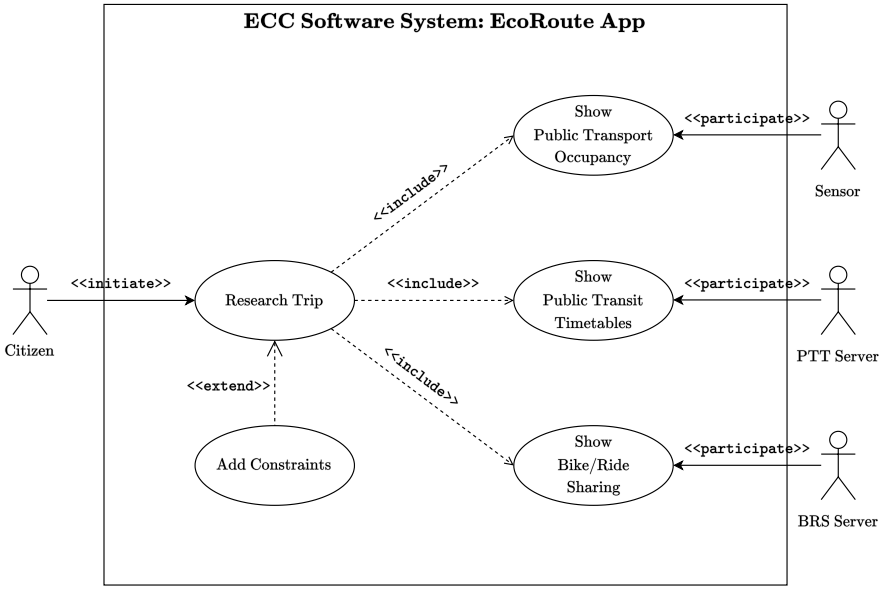

Figure 1: Use case diagram for the EcoRoute App.

To view/download the Use Case Diagram in high quality, click or scan the QR code below:

As we can see from the use case diagram (Figure 1), there are 5 use cases identified. On page 7, in Table 1, we give an exhaustive explanation of the Research Trip use case. However, in the following list, we talk about the other use cases:

- **Show Public Transport Occupancy**. The Public Transport Occupancy is a crucial resource gathered by the EcoRoute (app) from the sensors deployed on the Public Transport and used to avoid bottlenecks at peak times.

  We have included this use case in the Research Trip use case.

- **Show Public Transit Timetables**. The Public Transit Timetables are essential for finding the correct public transport route for the user. The app also uses them to find (and combine) the most environmentally friendly route.

  We have included this use case in the Research Trip use case.

- **Show Bike/Ride Sharing**. Bike/Ride Sharing is mainly used to suggest more ecological means of transport to the user, such as bicycles. The app also uses it to combine with public transport. We have included this use case in the Research Trip use case.

- **Add Constraints**. The user can add constraints, but this choice is not required by the system (as the project problem says). So the user can decide if to impose some limitations or not. We have extended the Research Trip use case with this use case.

Use Case: **Research Trip**	
Primary Actor	Citizen.
Supporting Actor	• Sensor; • PTT Server (Public Transit Timetable); • BRS Server (Bike-Ride-Sharing).
Description	A citizen, the application user, inserts the information about their trip. The information required is the origin and the destination. The constraints can be optional (Add Constraints use case extend Research Trip). Once the user does the research (e.g., by clicking the "research" button), three use cases extend the Research Trip: the Sensor, the PTT Server, and the BRS Server participate in the research made by the user, showing the different ecological choices. The user sees the final result, but the app makes the low-level requests, calculations, and others.
Data	Origin, destination, (optional) constraints, environmentally friendly routes.
Comments	The citizen must enter the origin and destination within the urban area.

Table 1: Detailed use case explanation - Research Trip.

2.3 Domain assumptions

The domain properties are descriptive assertions that are assumed to hold in the world (the part of the real world that is affected by the "machine", in our case EcoRoute). In this project, the domain assumptions are as follows:

- *Sensors are correctly deployed on public transport means and can share occupancy rates with the ECC.*
- *Public transit timetables are available and can be acquire in real-time from the ECC.*
- *Services that offers bike and/or ride sharing can share their information with the ECC.*
- *The Citizens stakeholder interacts with the software system provided by the ECC through a mobile application (called EcoRoute).*
- *Citizens must have an internet connection to use the mobile application.*

2.4 Requirements

2.4.1 Functional requirements

The functional requirements describe the interactions between the system and its environment. Then our application provides the following requirements:

- The mobile application will allow users (citizens):
  - To find an environmentally friendly route.
  - To add any constraints.
  - To view all or one of the Public Transport Occupancy, Public Transit Timetables, Bike/Ride Sharing information in the final result.
- The system should guarantee that the environmentally friendly route computed is the most eco-friendly and give preference to combinations of different modes of transport.

2.4.2 Non-functional requirements

The non-functional requirements specify or constrain characteristics of the system as a whole. Then our application provides the following requirements:

- The EcoRoute application must be available to all users (citizens) 24 hours a day, 7 days a week, including public holidays.
- Downtime during normal working hours should not exceed 1 minute, as this is when the application is most used by citizens.

  Holiday downtime should not exceed 5 minutes.
- The algorithm time to calculate the best environmentally friendly routes should not exceed 10 seconds.

3 Design

3.1 General description of the architecture

The architecture chosen for the EcoRoute project is Client-Server, but with the worker pooling (NGINX web server). The reason for this choice is discussed in section 3.3 on page 13.

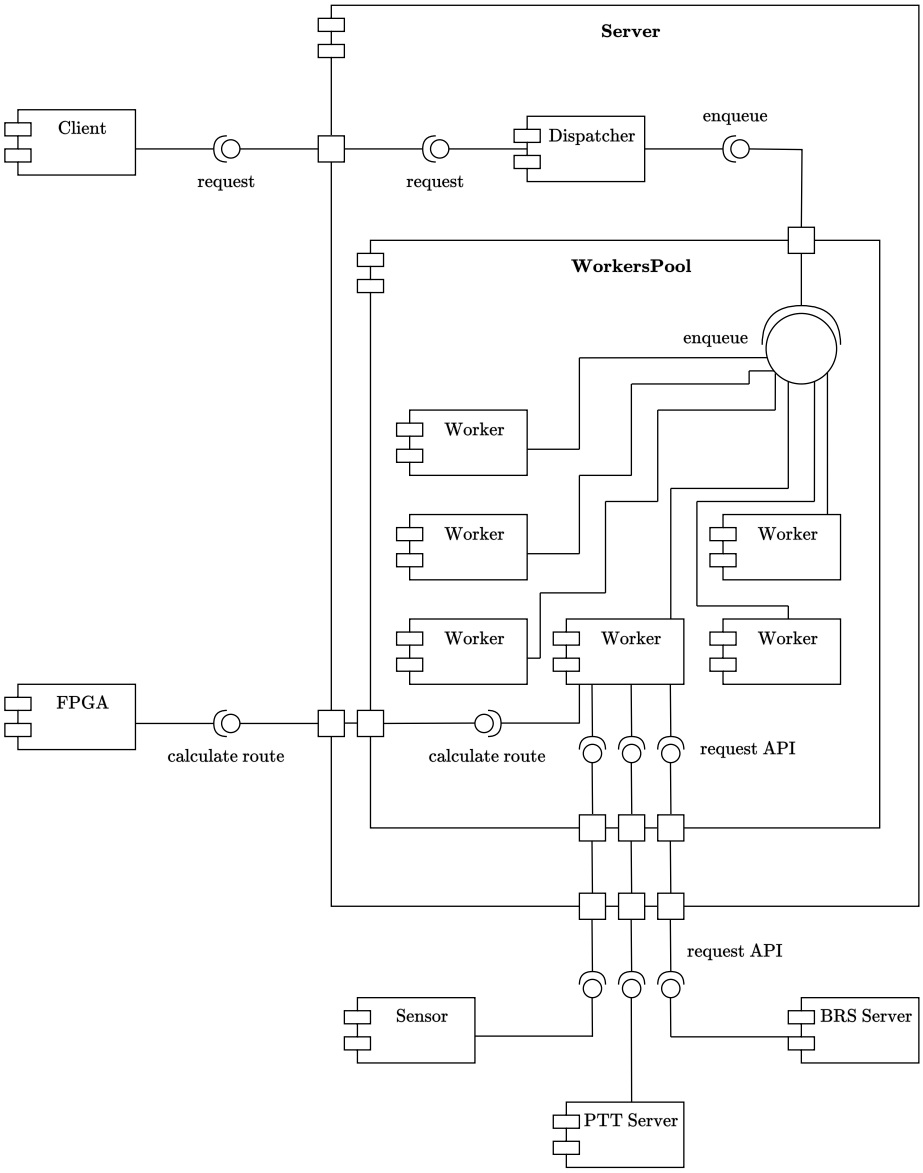

Figure 2: Component Diagram for the EcoRoute App.

To view/download the Component Diagram in high quality, click or scan the QR code below:

Before we go into details, let us assume that sensors are deployed on public transport means and that they publish updated data on occupancy rates at each stop. The data are saved into a HPC database, such as ScyllaDB. We have analyzed this assumption again in the section 3.3 on page 13.

It is also important to note that only one module (Worker) is connected to the Sensor, PTT Server and BRS Server. This is a design choice to avoid a less readable component diagram. We can imagine that each Worker is connected to Sensor, PTT Server and BRS Server.

The following list examines each component of the component diagram (Figure 2):

- **Client**
- **Server**
- **Dispatcher**
- **Workers and Workers Pool**
- **Sensor (ScyllaDB)**
- **PTT Server**
- **BRS Server**
- **FPGA**

3.2 Sequence diagrams

3.3 Critical points and design decisions