André Coutinho (pg39284) - Francisco Saraiva (pg39287) - Rui Santos (pg41850)

Business functions

SPWS:

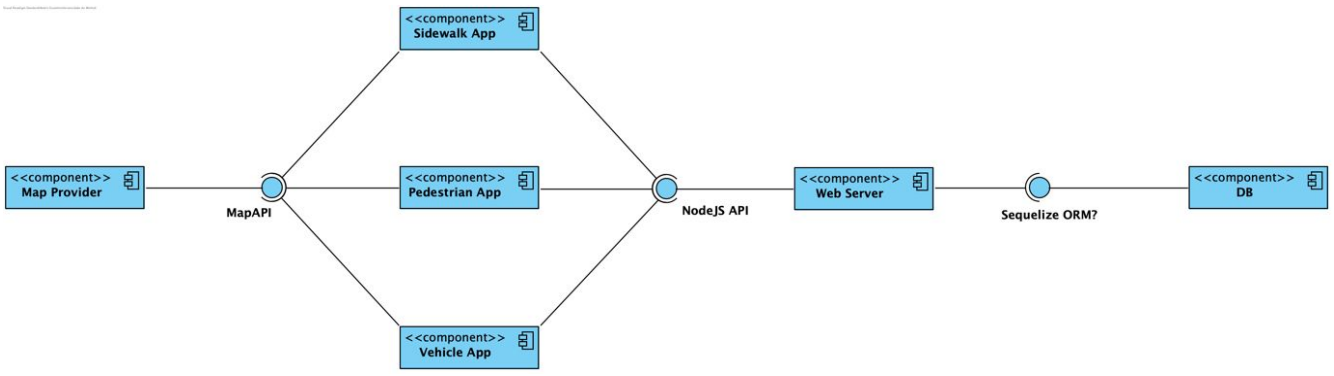
1. The user should be able to create, read, update and delete information on all crosswalks;
2. The user should be able to list all crosswalks;
3. The user should be able to consult, for a specified crosswalk, the number of pedestrians and/or cars in the surroundings as well as their location;
4. The user should be able to consult the number of pedestrians and vehicles recorded on a given day in the immediate surroundings of a crosswalk;
5. The system should notify vehicles (Vehicle App) of the state of the traffic light in an upcoming crosswalk.

Vehicle App:

6. The system should transmit its location to the SPWS when inside the radius of a crosswalks neighbourhood;
7. The system should stop transmitting its location whenever outside a crosswalks neighbourhood;
8. The system should alert about the presence of pedestrians in an upcoming crosswalk as well as alert about the state of the corresponding traffic light.

Pedestrian App:

9. The system should register its coordinates when neighbouring a crosswalk;
10. The system should stop reporting its coordinates after leaving a crosswalk;
11. The system should notify the SPWS about its location when inside the radius of a crosswalks neighbourhood.



Map Provider should be a third-party component;
Sidewalk App is the front-end for the component number one*;
Vehicle App is the front-end for the component number two*;
Pedestrian App is the front-end for the component number three*;
Web Server is the back-end component encapsulating business logic;
DB is the database used to store information.

*according to the assignment document

The business functions at hand should be built considering an *asynchronous* mechanism, except when the systems should perform operations in the database. For example, the business functions that are responsible for transmitting coordinates and notifications (5, 6, 11) should benefit from an *asynchronous* mechanism while business functions such as number 1 should be *synchronous*.

Considering this decision it will make our system mostly *request/response* based.

The *architectural style* we will be using will be **Orchestration** with the backend API being the component responsible for providing shared logic to the remainder components of our system. This way it will function as the center communicator for all our apps.
Orchestration is useful in this project since we're dealing with just a proof of concept, making it a small system.
It is understandable to resort to **Choreography** when our environment consists of hundreds or even thousands of microservices. It is also worth noting that in a **Choreography architectural style, event brokers** are used. These are especially useful when our services may take a significant amount of time coordinating logic between them which is not the case at hand.