Admin Dashboard for Cities

1 Introduction

Purpose:

The admin dashboard allows cities manager to interact within the system and ensure users (citizens, urban planner, …) can visualize relevant data about their cities.

Description:

The administration dashboard allows cities managers to manage the data of two different tools.

First, it is possible to manage the communication tool. The UI allows to manage all the hotspot data shown to the citizens. Along this, managers can create, update, delete, show or hide hotspots and manage the data of each individual hotspot too. Each hotspot may have one or multiple pages, these can contain various data, such as pictures, videos, 360 views, texts and all of these either as standalone elements or as a comparison (left/right – past-present or present-future). In these pages, survey and bookings can also be included.

Should we also be able to manage maps in communication tools? => Google Maps (…) or custom maps? Check with Emilie

Should be able to define bounding boxes of the cartography.

Forgot the aspect about comparison of multiple possibilities

Second, it is possible to manage the co-design tool. It allows to configure maps (upload, delete, …), 3D assets with some attributes, proposals (and their related KPIs) and POI (points of interest).

For both tools, surveys can be managed (and linked in the respective elements of the tools) to measure the impact of the data shown in the tool towards citizens and/or urban planners.

KPI TBD

Bucharest wants to upload assets in the cartography

Attach parameters to assets and attach assets to interventions

In the end, visualize KPI changes among multiple scenarios

2 Where this component fits in the ReGreeneration architecture

This component lies in the end-user layer, making it directly accessible to cities managers.

It communicates with a cloud service, which is exposing the data stored and ensuring consistency of data saved by that UI.

This component has direct incidence on the data shown in the “Map based visualisation of city data” and “Citizen’s interactions” blocks.

Should it interact also with the “Map Integration SDK” somehow?

Should it interact also with the “Data” block, as configuration? (new data, remove data, …)

3 Module description

The module is mainly a frontend application, written in Angular (?).

It is composed of multiple Angular components interacting with each other.

Each component should be as reusable as possible, allowing easy integration at different location and preventing rewriting of existing elements.

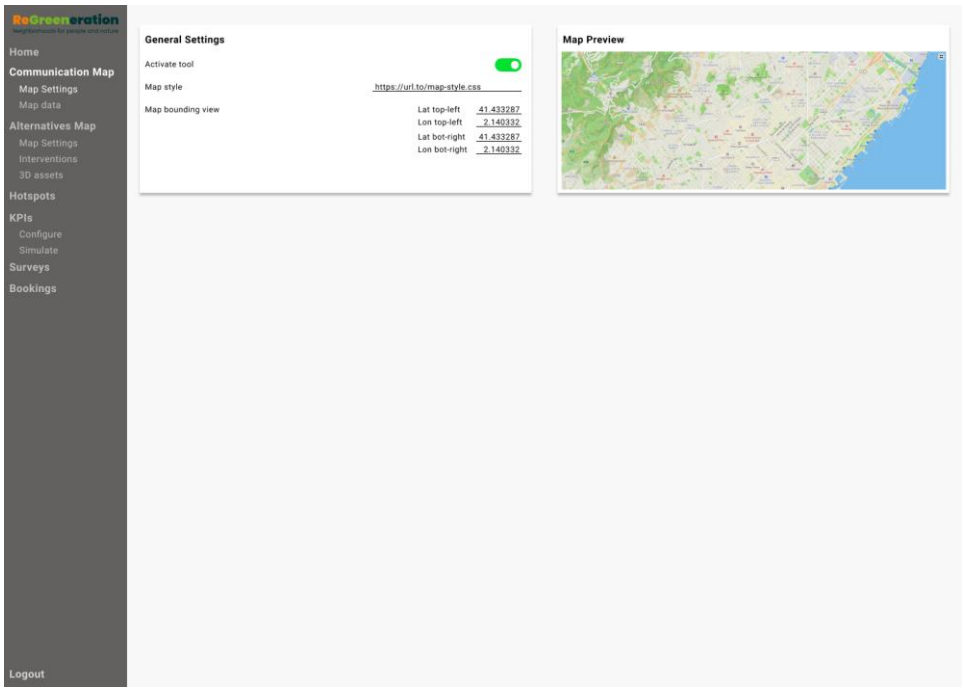
3.1 Mock-ups of a potential UI

3.1.1 Communication Map

3.1.1.1 Map Settings

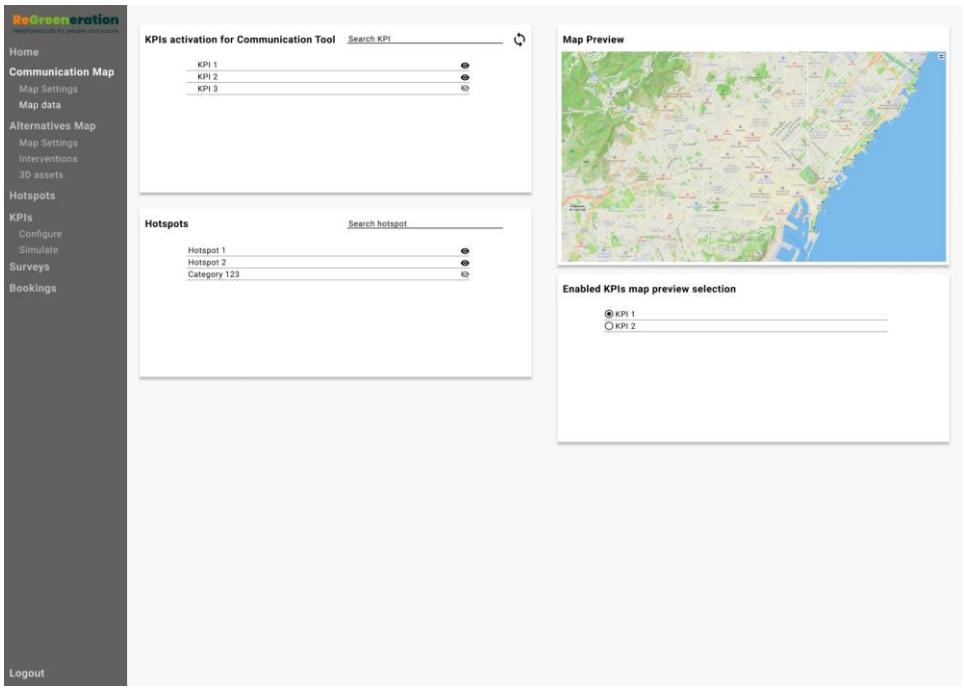
This page allows cities to manage the settings of the communication map:

- Activation
- Map style
- Map boundaries (limit the map for the citizens)



3.1.1.2 Map Data

It allows to select the data shown on the map, by selecting the KPIs and hotspots that will be shown on the communication map.
A preview of the selection is also visible on the right side, as well as the selection of the current KPI data on the map.

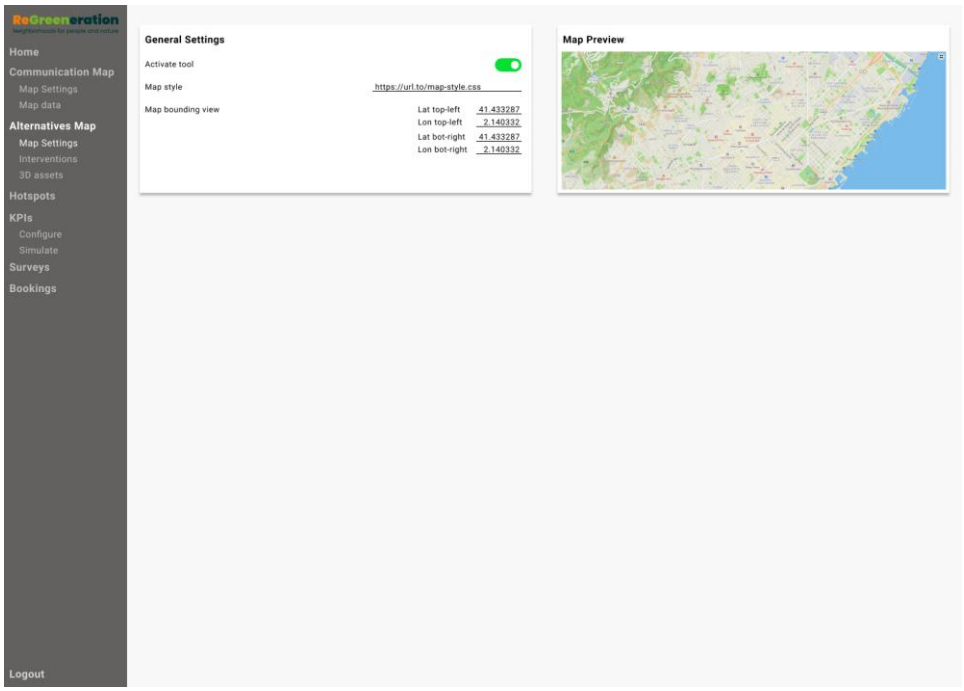


3.1.2 Alternative Map

3.1.2.1 Map Settings

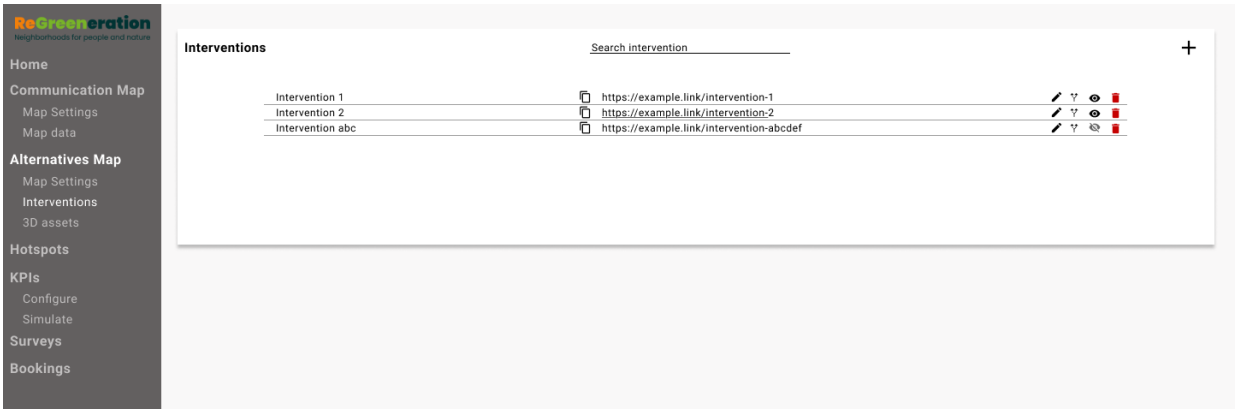
This page allows cities to manage the settings of the communication map:

- Activation
- Map style
- Map boundaries (limit the map for the citizens)



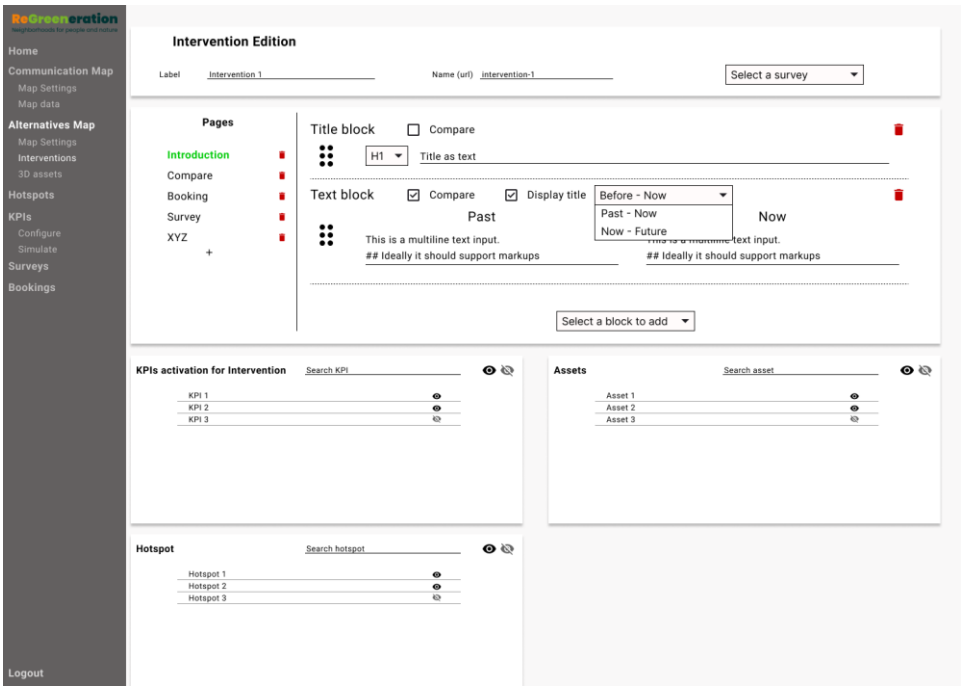
3.1.2.2 Intervention

- Listing: Listing page of all interventions, showing their links and allowing to edit them, edit their alternatives, enable/disabled them and delete them.



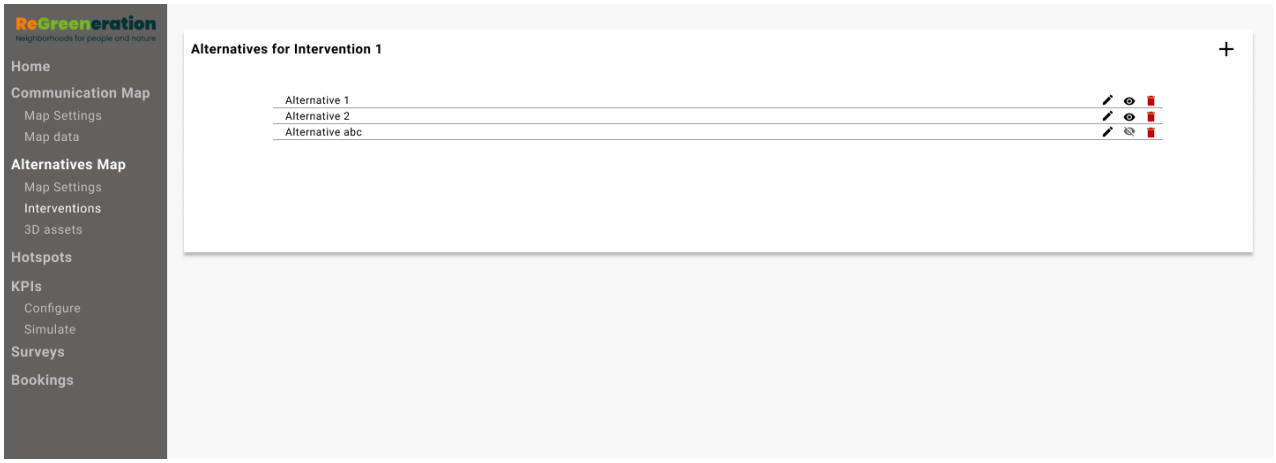
- Edition/Creation:

  Allows to define an intervention's details, including:
  - Label
  - URL name
  - Survey linking (call to action button)
  - Description definition as pages with blocks
  - KPI activated for the whole intervention
  - Assets activated for the whole intervention
  - Hotspots activated for the whole intervention



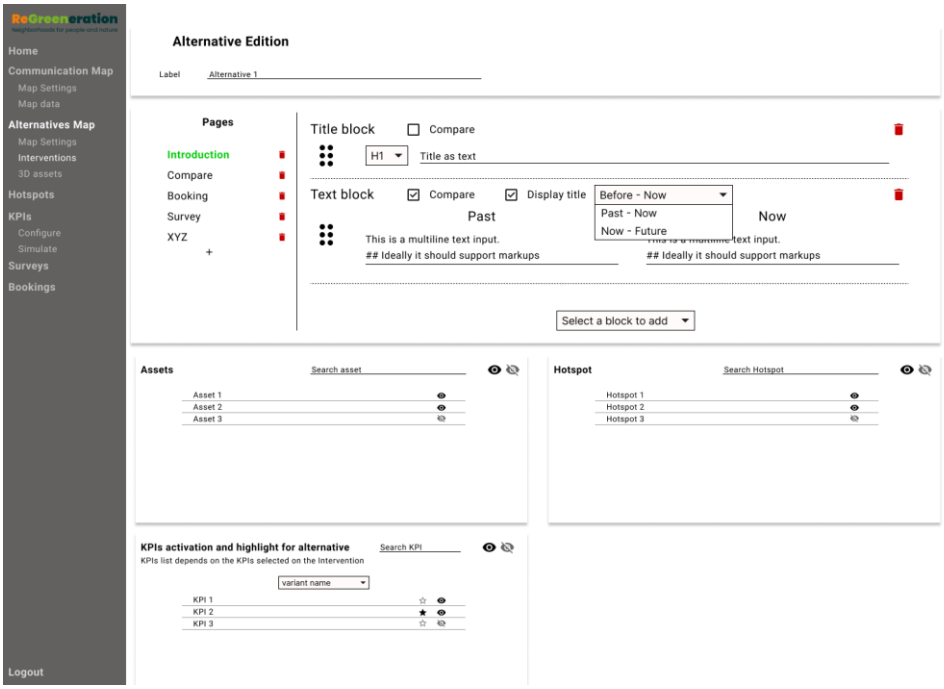
- Alternatives listing of an intervention

  List an intervention's alternatives, allowing to create, edit, enable/disable or delete them.



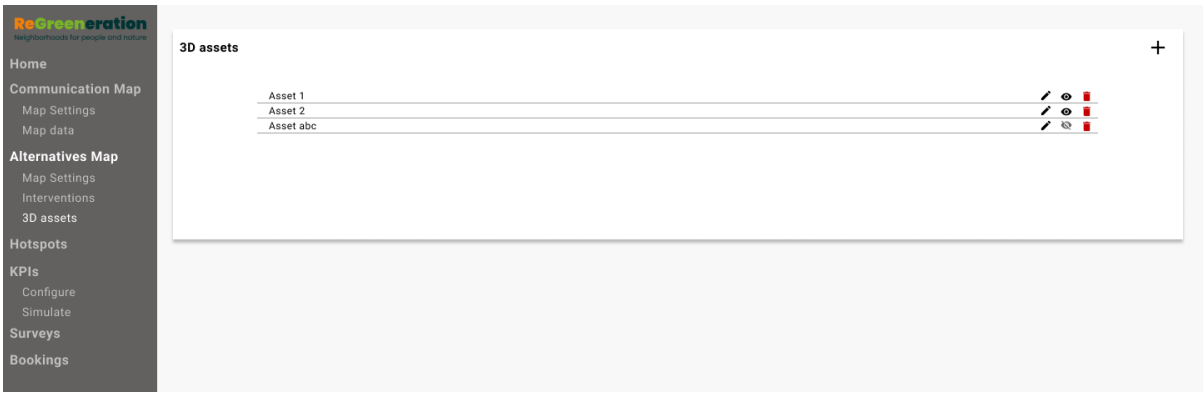
- Alternative edition/create

  Allows to define an alternative detail, including:
  - Label
  - Description definition as pages with blocks
  - KPI activated for the alternative (when selecting it as a citizen)
  - Also, a possibility to highlight up to 3 KPIs
  - Assets activated for the alternative (when selecting it as a citizen)
  - Hotspots activated for the alternative (when selecting it as a citizen)



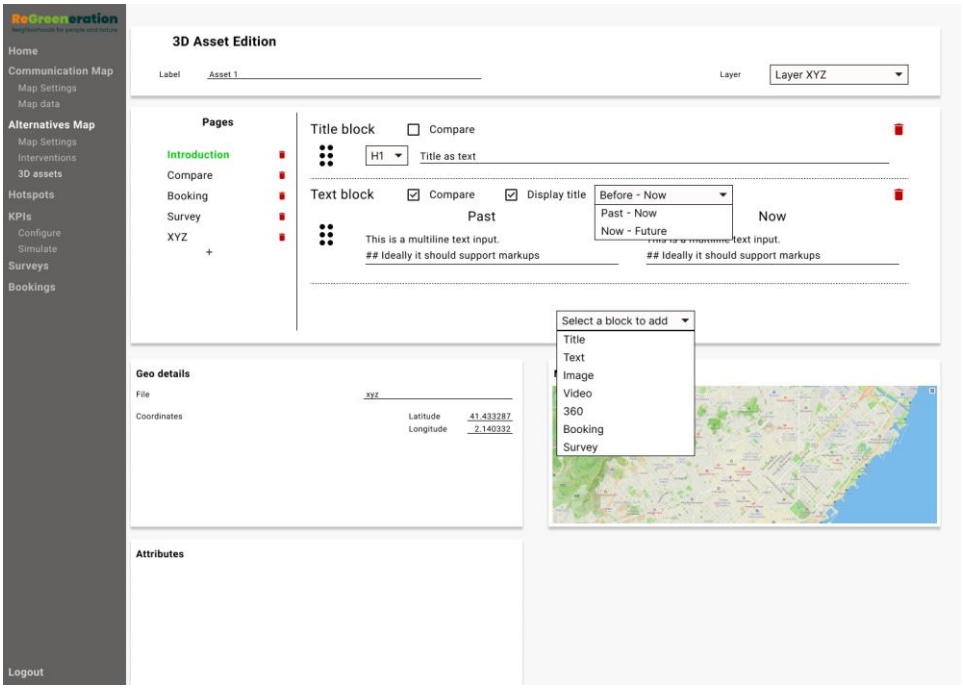
3.1.2.3 3D Assets

- Listing: Shows the list of all the 3D Assets, allowing to create, edit, delete or disable/enable them globally.

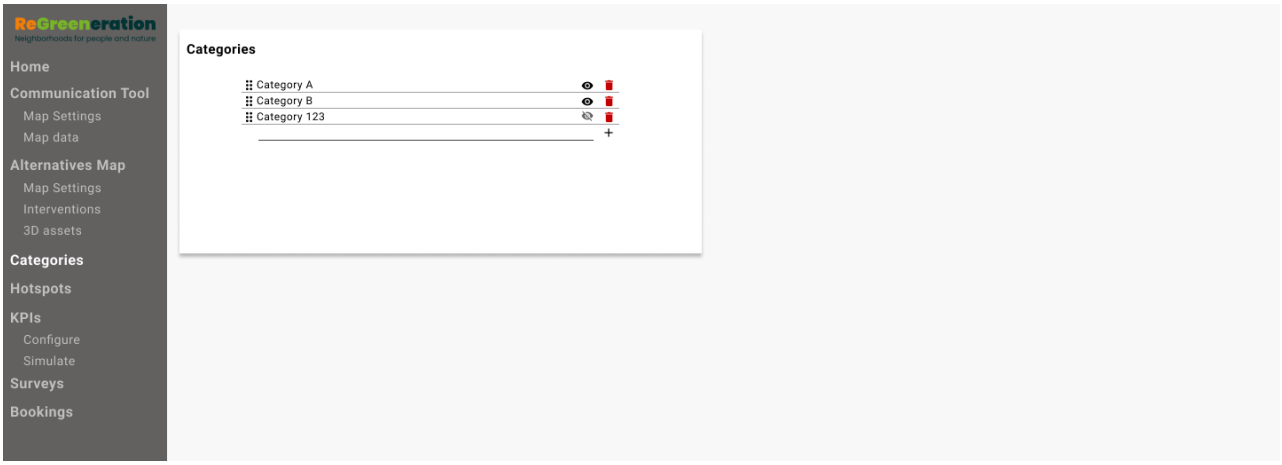


- Edition/Creation

  Allows to define an intervention's details, including:
  - Label
  - Category
  - Description definition as pages with blocks
  - Geo details
  - Attributes (TBD)

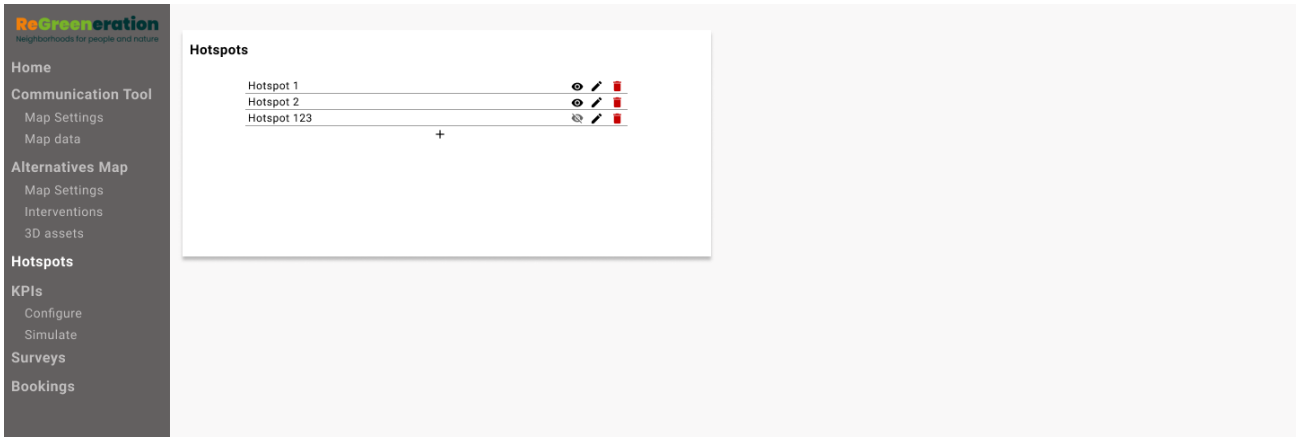


- Categories:
  Allow to create, edit, delete and enable/disable categories used for hotspots and 3D assets.



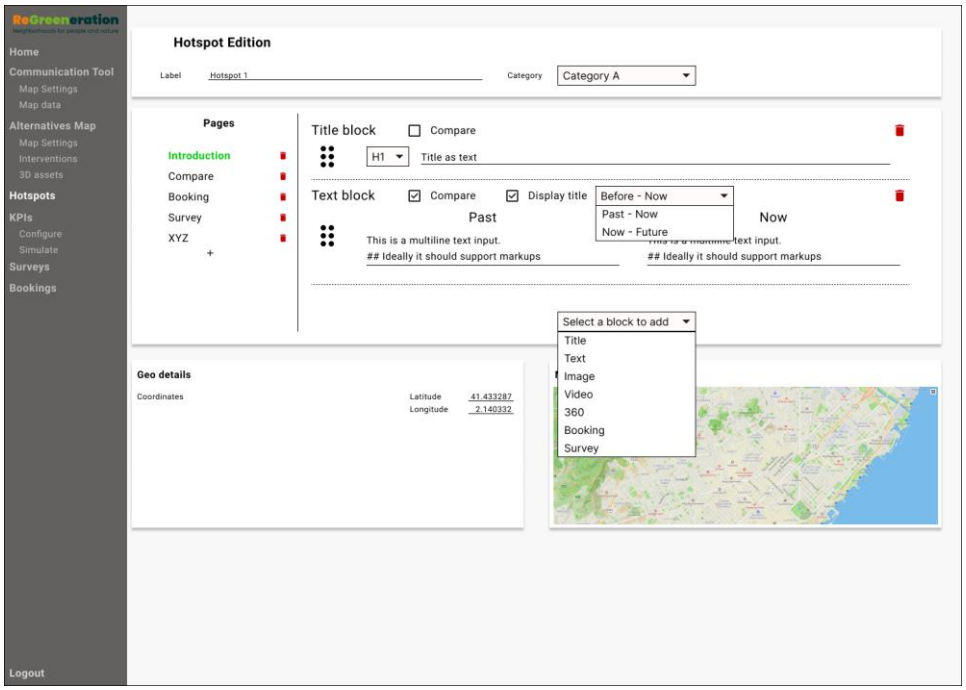
3.1.2.4 Hotspots

- Listing: Listing of all the hotspot available in the app. Hotspots can be created, edited, deleted or enabled/disabled.



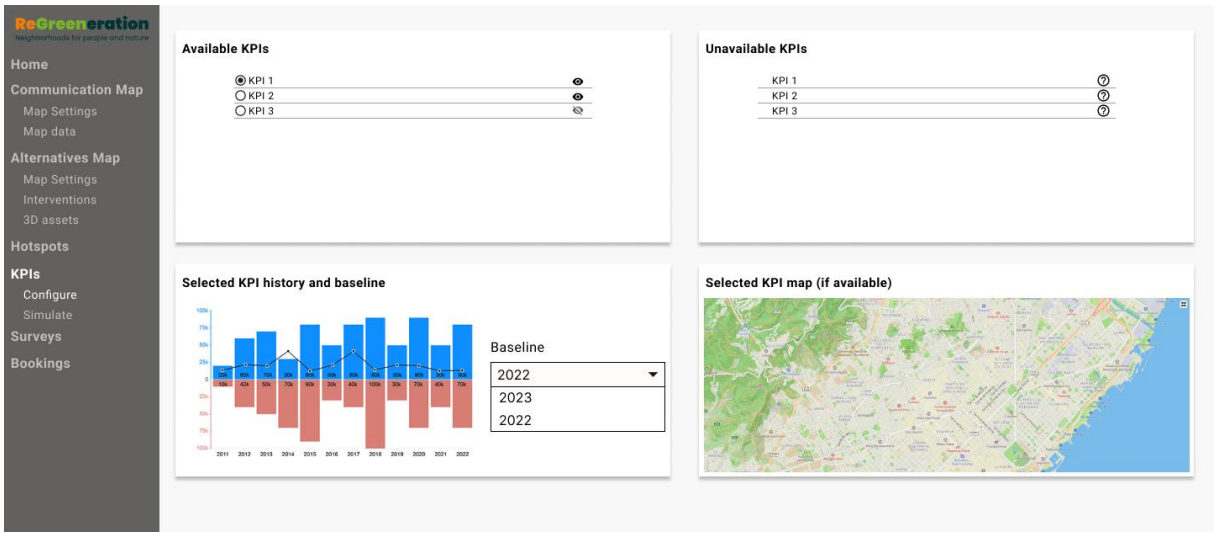
- Edition/Creation:

  Allows to define an intervention's details, including:
  - Label
  - Category
  - Description definition as pages with blocks
  - Geo details



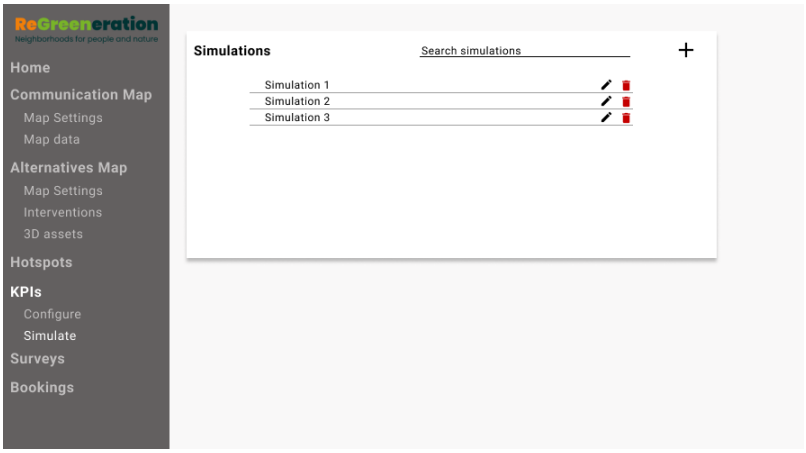
3.1.2.5 KPIs

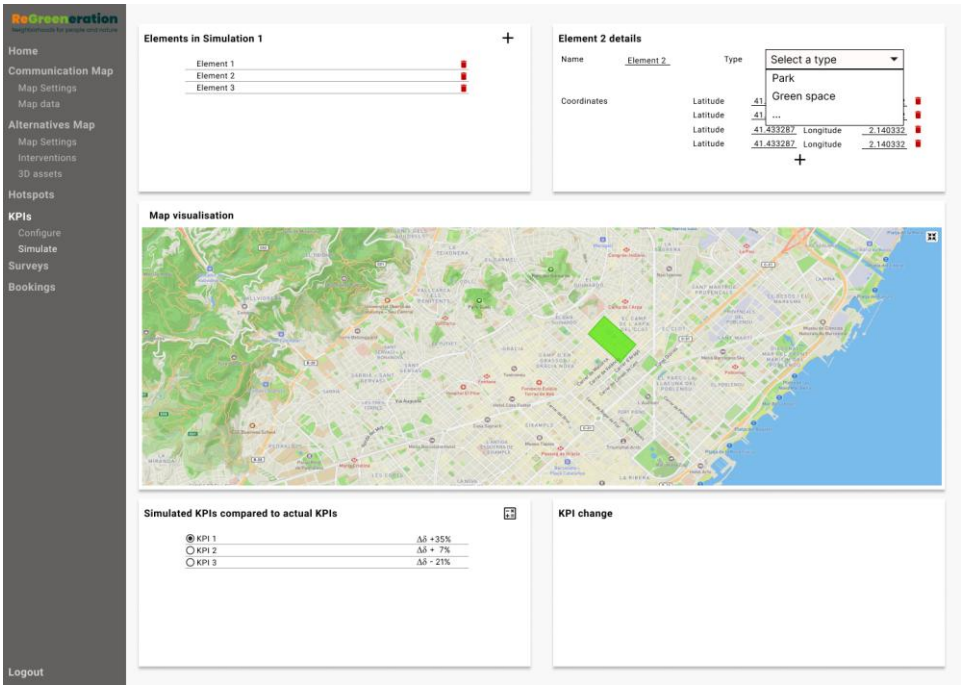
- Configuration
  KPIs baseline can be configured in the system. Available KPIs are also listed (data available) and unavailable KPIs too, along with an explanation on which data need to be uploaded.



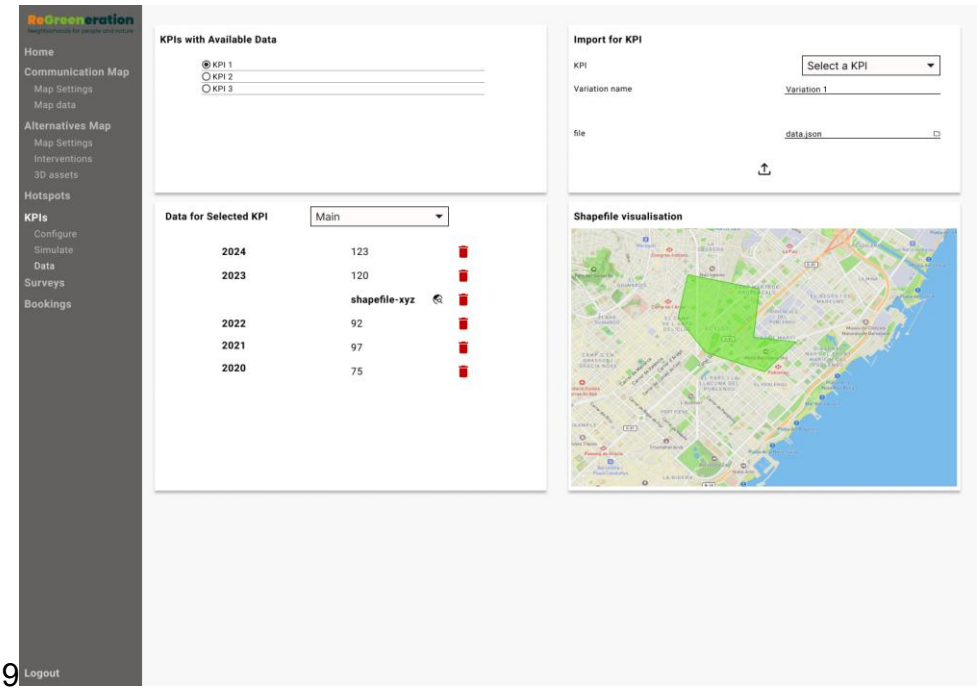
- Simulation
  Simulations can be created, edited or deleted.
  When creating or editing one, multiple elements can be added on the map, each of them has a definition:
  - Name
  - Type (Green space, parking, playground, …)
  - Coordinates

A simulation can then be computed by the system (TBD), creating a variation for the KPIs and showing the delta between current value and simulated value at the bottom of the page.

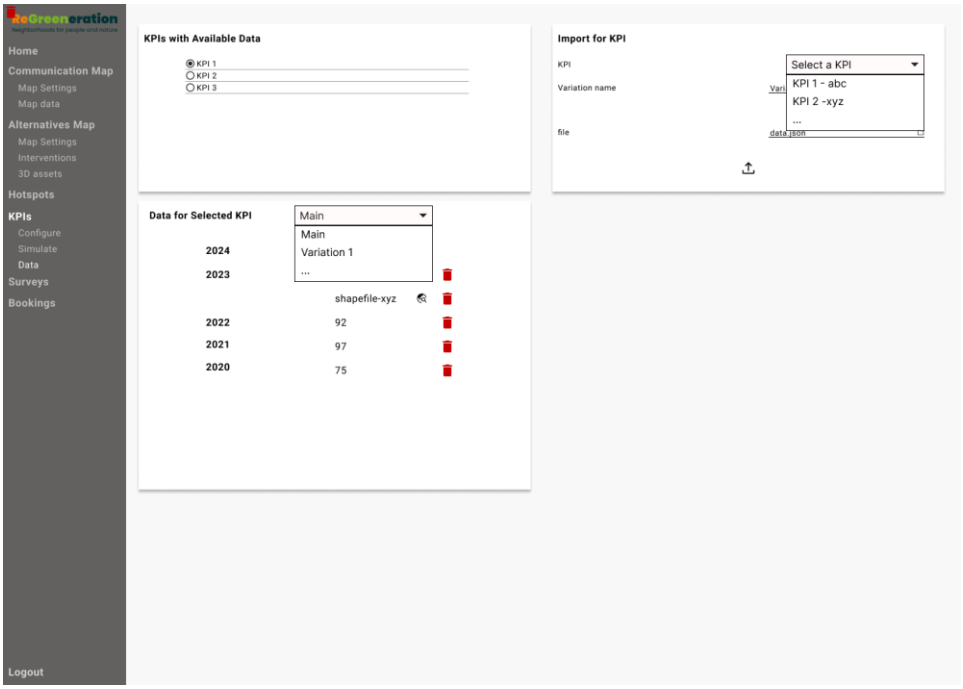




- Data Ingestion/Visualisation
  Data can be ingested directly from the administration panel.
  To ingest data, a KPI must be selected, as well as a variation name, and finally a file.
  This section is subject to change as it depends on the data to be uploaded.
  When visualizing the data at the bottom, managers can select a variation (related to simulation) of the data and, if applicable, visualize it on the map.



9



4 Technical Foundations and Background

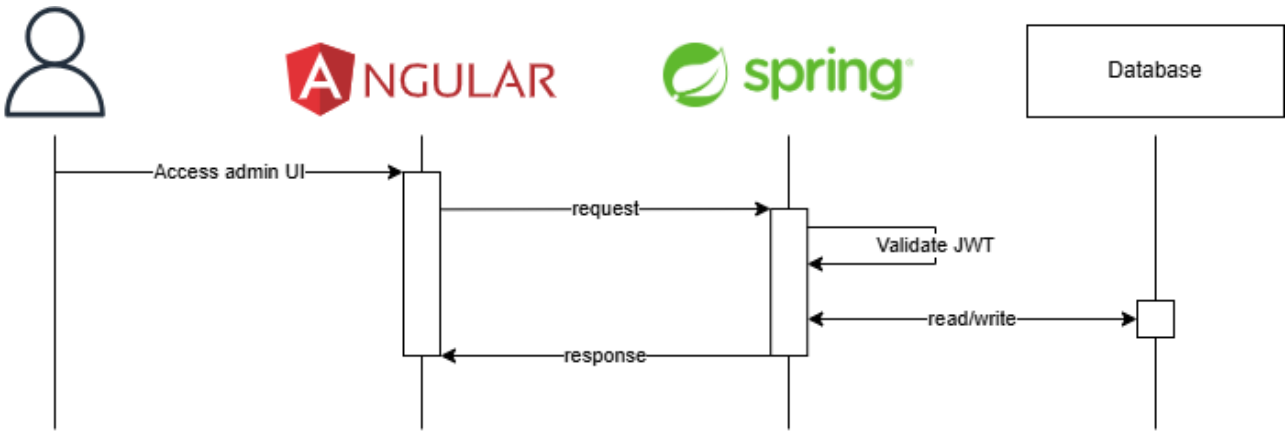
Subcomponent/Component	Owner	License

5 Sequence diagram of module components

The following sub-sections describe the sequence diagrams of the module.

5.1 Communication Map

General flow of requests



- **Retrieve Map Settings**

Users navigate to the map settings page; the angular application reaches out to the Spring backend which retrieve the settings from the Database.

- **Save Map Settings**

Users save their changes, which are sent to the Spring application by Angular and stores them in the database.

- **Retrieve Map Data**

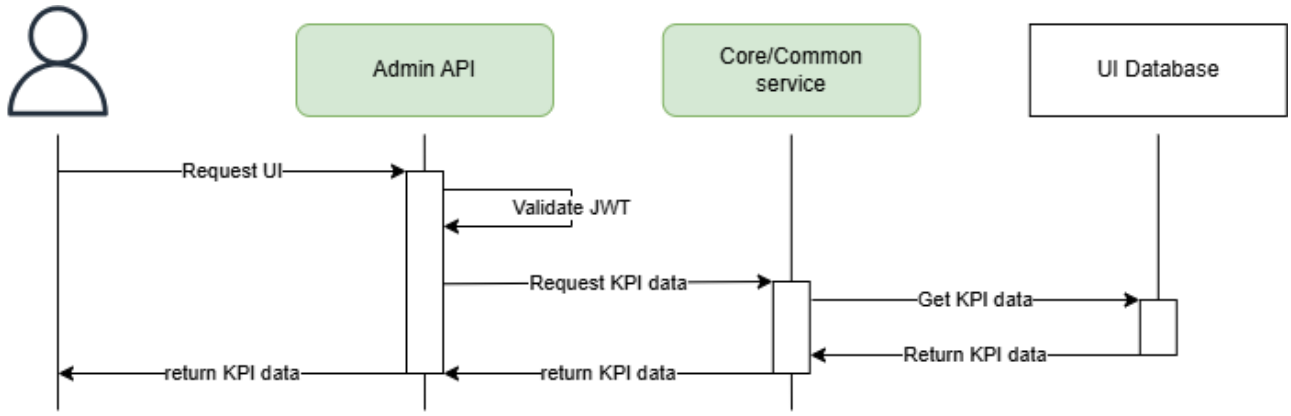
Users navigate to the map data page; the angular application reaches out to the Spring backend which retrieve the map data from the Database.

- **Save Map Data**

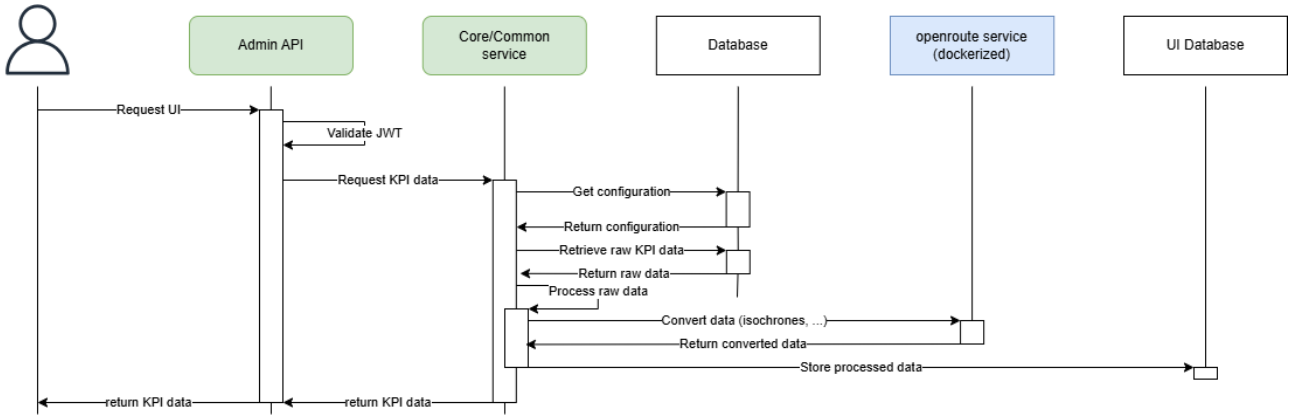
Users save their changes, which are sent to the Spring application by Angular and stores them in the database.

- **Retrieve KPI data for Map Preview**

Whenever users select a KPI to render on the map, the Angular Application sends a request to Spring which retrieve the data in the UI database.
If data is available:



If data is not available:



5.2 Alternative Map

- **Retrieve Map Settings**

Users navigate to the map settings page; the angular application reaches out to the Spring backend which retrieve the settings from the Database.

- **Save Map Settings**

Users save their changes, which are sent to the Spring application by Angular and stores them in the database.

- **Retrieve list of Interventions**

When users access the list of intervention page, the Angular application requests the list of intervention to Spring which retrieves it from the database.

- **Retrieve an intervention**

When users access an intervention, the Angular application requests the intervention details to Spring which retrieves it from the database.

- **Enable/Disable an Intervention**

Users change the status of an intervention (disabled/enabled), Angular sends the status to the Spring application which stores it in the database.

- **Delete an Intervention**

Users delete an intervention, Angular sends the intervention id to the Spring application which marks it as deleted in the database.

- **Create an Intervention**

Users create an intervention, Angular sends the intervention details to Spring which validates the data and save it in the database.

- **Edit an Intervention**

Users edit an intervention, Angular sends the intervention details to Spring which validates the data and save it in the database.

- **List an Intervention Alternatives**

When users access an intervention's alternatives page, the Angular application requests the alternative list to Spring which retrieves it from the database.

- **Retrieve an alternative**

When users access an alternative page, the Angular application requests the alternative details to Spring which retrieves it from the database.

- **Enable/Disable an Alternative**

Users change the status of an alternative (disabled/enabled), Angular sends the status to the Spring application which stores it in the database.

- **Create an Alternative for an Intervention**

Users create an alternative, Angular sends the alternative details to Spring which validates the data and save it in the database.

- **Edit an Alternative for an Intervention**

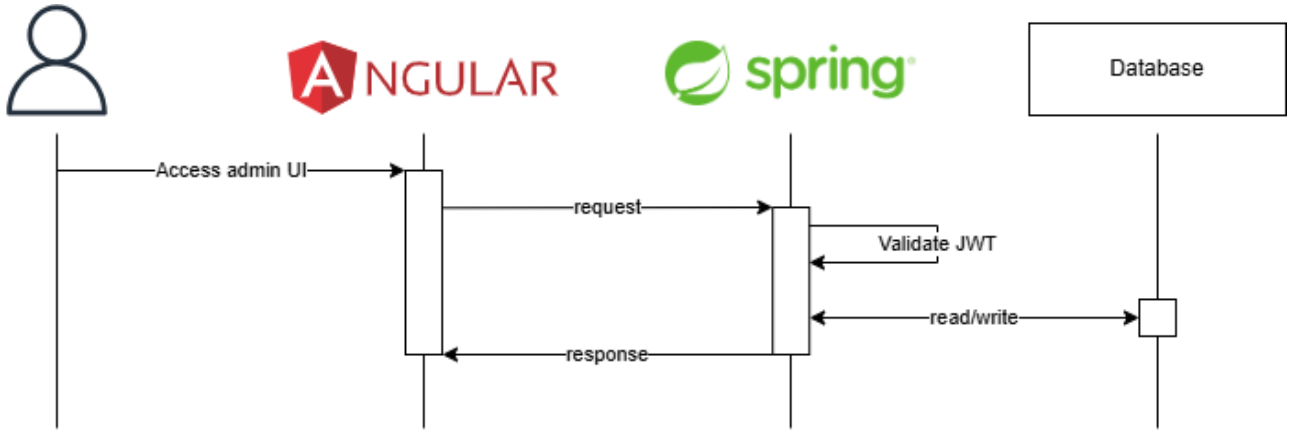
Users edit an alternative, Angular sends the alternative details to Spring which validates the data and save it in the database.

- **Delete an alternative**

Users delete an alternative, Angular sends the intervention id to the Spring application which marks it as deleted in the database.

5.3 3D Assets

General flow of requests



- **List all Assets**

When users access the list of assets page, the Angular application requests the list of assets to Spring which retrieves it from the database.

- **Retrieve an asset**

When users access an asset, the Angular application requests the asset details to Spring which retrieves it from the database.

- **Enable/Disable Asset**

Users change the status of an asset (disabled/enabled), Angular sends the status to the Spring application which stores it in the database.

- **Delete an Asset**

Users delete an asset, Angular sends the asset id to the Spring application which marks it as deleted in the database.

- **Create an Asset**

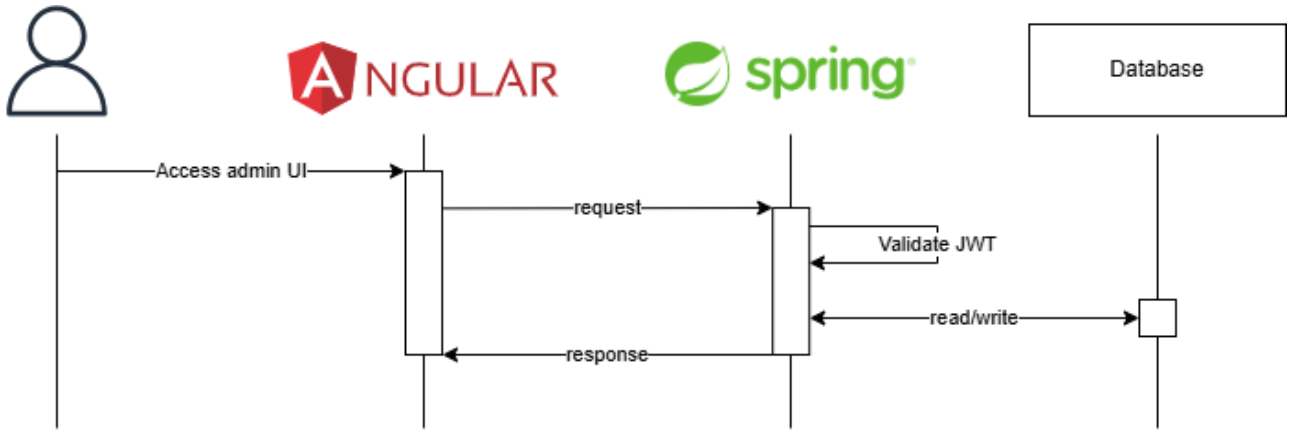
Users create an asset, Angular sends the asset details to Spring which validates the data and save it in the database.

- **Edit an Asset**

Users edit an asset, Angular sends the asset details to Spring which validates the data and save it in the database.

5.4 Categories

General flow of requests



- **Fetch Categories list**

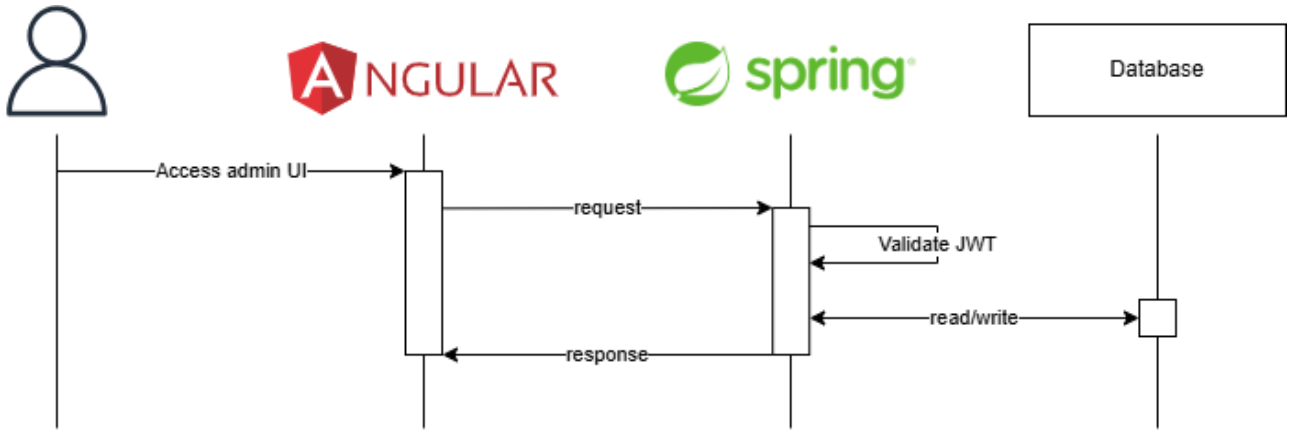
When users access the list of categories page, the Angular application requests the list of categories to Spring which retrieves it from the database.

- **Save Categories list**

When users save the categories after changes on the page, the Angular application sends the list to the Spring backend which saves it in the database after validation.

5.5 Hotspots

General flow of requests



- **List all hotspots**

When users access the list of hotspots page, the Angular application requests the list of hotspots to Spring which retrieves it from the database.

- **Enable/Disable a hotspot**

Users change the status of a hotspot (disabled/enabled), Angular sends the status to the Spring application which stores it in the database.

- **Delete a hotspot**

Users delete a hotspot, Angular sends the hotspot id to the Spring application which marks it as deleted in the database.

- **Create a hotspot**

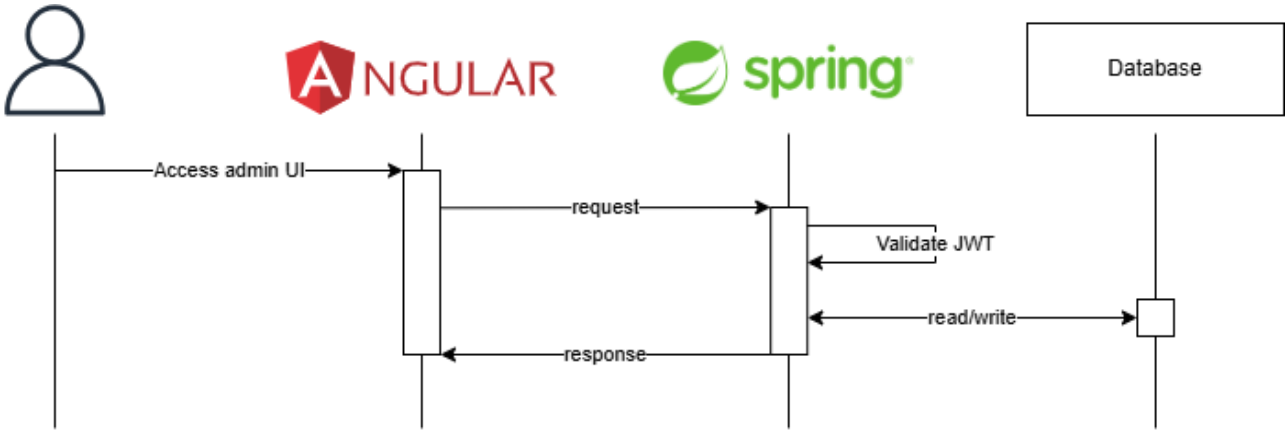
Users create a hotspot, Angular sends the hotspot details to Spring which validates the data and save it in the database.

- **Edit a hotspot**

Users edit a hotspot, Angular sends the hotspot details to Spring which validates the data and save it in the database.

5.6 KPIs

General flow of requests



- **Retrieve list of available and unavailable KPIs**

Users navigate to the KPI configuration page, the angular application reaches out to the Spring backend which retrieve the KPI available with their data and the unavailable KPIs (without data) from the Database.

- **Save a KPI baseline**

Whenever the user edits a KPI baseline, it is automatically saved by the angular application by sending the new baseline reference for that KPI to the Spring application which saves it in the database.

- **List all KPI simulation**

When users access the list of KPI simulation page, the Angular application requests the list of existing simulation to Spring which retrieves it from the database.

- **Get a KPI simulation**

When users access the page of a KPI simulation, the Angular application requests the simulation details to Spring which retrieves it from the database.

- **Delete a KPI simulation**

Users delete a KPI simulation, Angular sends the simulation id to the Spring application which marks it as deleted in the database.
Should it delete also the simulated data?

- **Create a KPI simulation**

Users create a KPI simulation, Angular sends the KPI simulation details to Spring which validates the data and save it in the database.
Should it also automatically compute it?

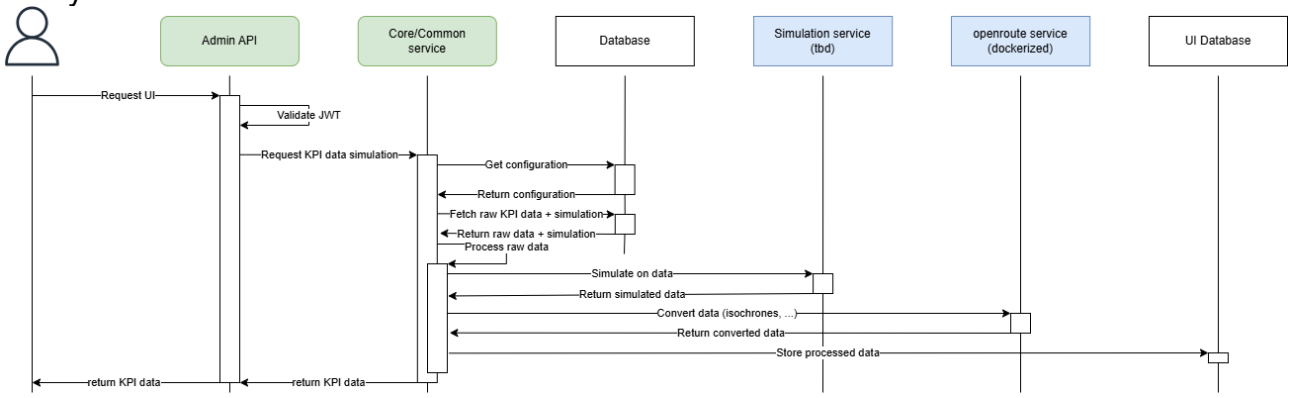
- **Edit a KPI simulation**

Users edit a KPI simulation, Angular sends the KPI simulation details to Spring which validates the data and save it in the database.

Should it also automatically compute it?

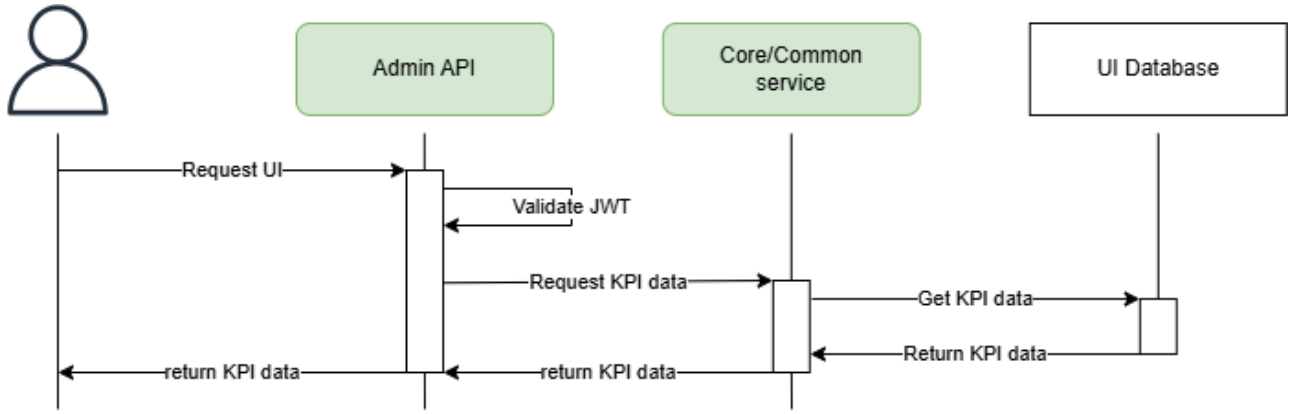
- **Compute new KPIs for a simulation**

Users defined a KPI simulation and run it. Angular sends a request to simulate the data. Spring retrieves the data and the simulation details and sends that to the simulation service (TBD). This data is then converted using open route service if applicable and finally stored in the DB.

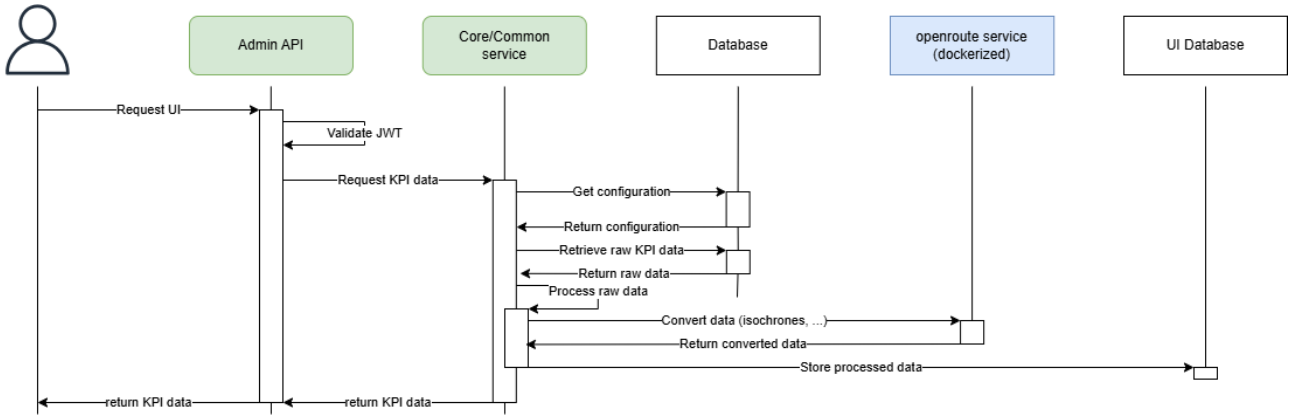


- **Retrieve KPI simulated and actual data**

Whenever users select a KPI and its variation, the Angular Application sends a request to Spring which retrieve the data in the UI database.
If data is available:



If data is not available:



- **List available KPI**

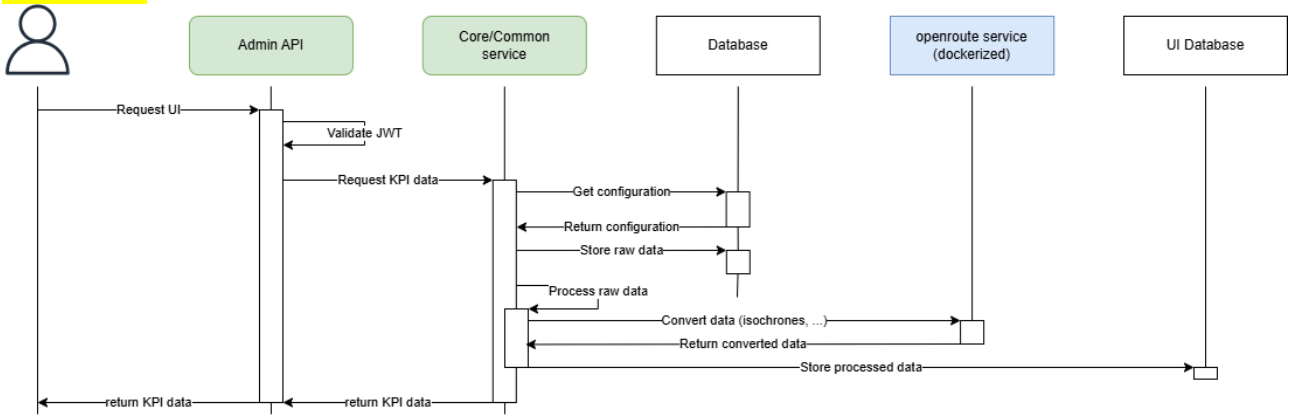
Users navigate to the KPI data page, the angular application reaches out to the Spring backend which retrieve the KPI available with their data from the Database.

- **List variation available for a KPI**

When users access the KPI data page, the Angular application requests the list of existing variation for each KPI to Spring which retrieves it from the database.

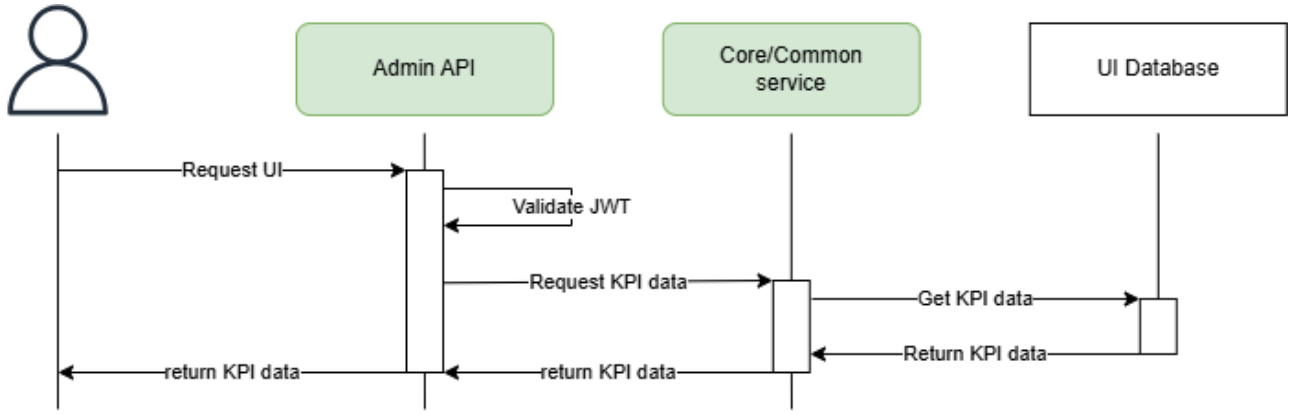
- **Upload data to a KPI and variation**

Should we allow custom variation too? => Probably yes so cities can do their own simulation

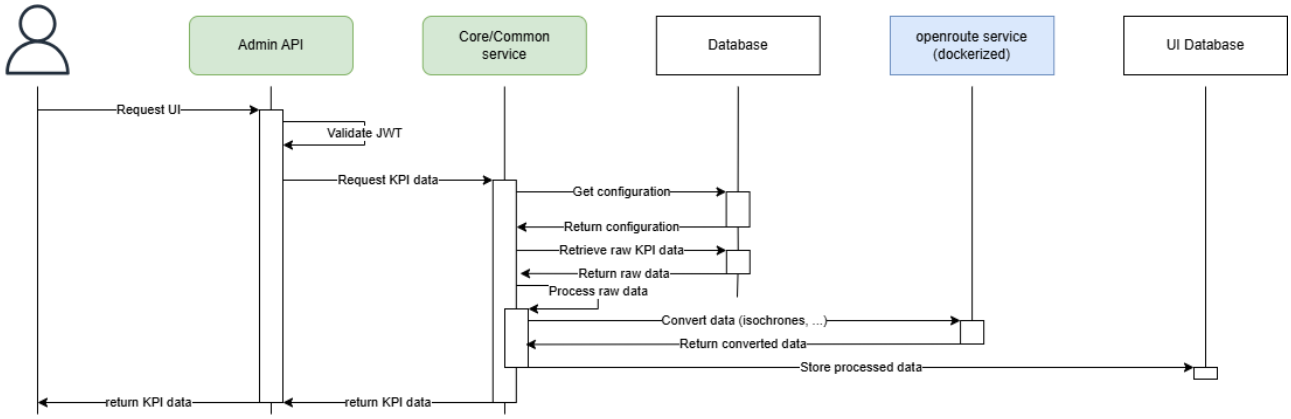


- **Retrieve data for a KPI and variation including map data**

Whenever users select a KPI and its variation, the Angular Application sends a request to Spring which retrieve the data in the UI database.
If data is available:



If data is not available:



5.7 Survey

- **List surveys**

TBD in OGS

- **Create Survey**

TBD in OGS

- **Edit Survey**

TBD in OGS

- **Close Survey**

TBD in OGS

5.8 Bookings

Should we maybe create a Booking module?

- **Listing all hotspot, including rooms**

The admin UI sends a request to the backend.
The backend retrieves the list of hotspot and rooms from the database and sends it to the frontend.

- **Creating a room for a hotspot**

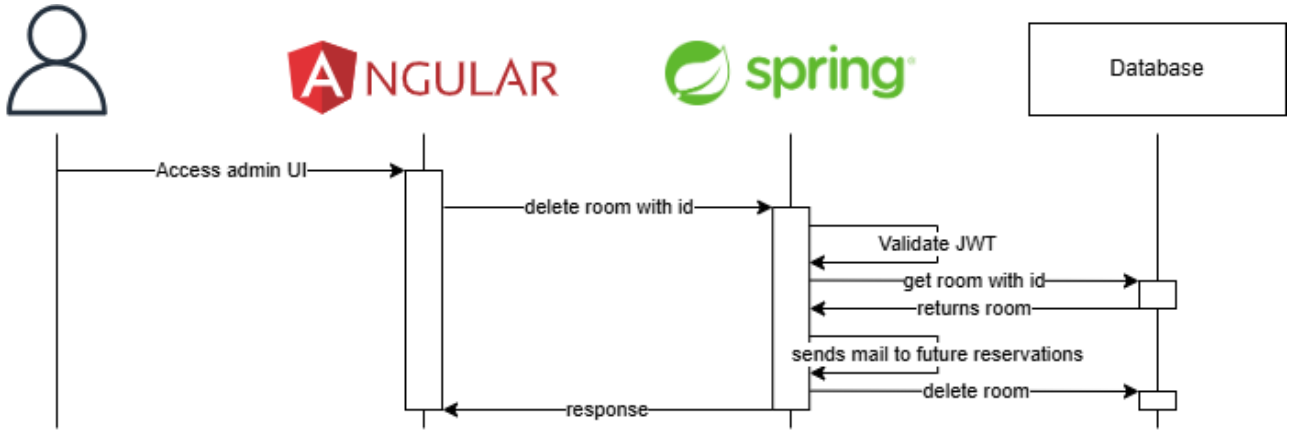
The admin UI sends a request to the backend.
The backend processes the request, stores the hotspot room in the database after validating it and sends it to the frontend.

- **Updating a room for a hotspot**

The admin UI sends a request to the backend.
The backend processes the request, stores the hotspot room in the database after validating it and sends it to the frontend.

- **Deleting a room for a hotspot**

The admin UI sends a request to the backend.
The backend processes the request, deletes the hotspot room in the database, sends a mail to inform about future reservation and sends a confirmation to the frontend.



- **Disabling a room for a hotspot**

The admin UI sends a request to the backend.
The backend processes the request, disables the hotspot room in the database, preventing any further reservations, and sends a confirmation to the frontend.

- **Enabling a room for a hotspot**

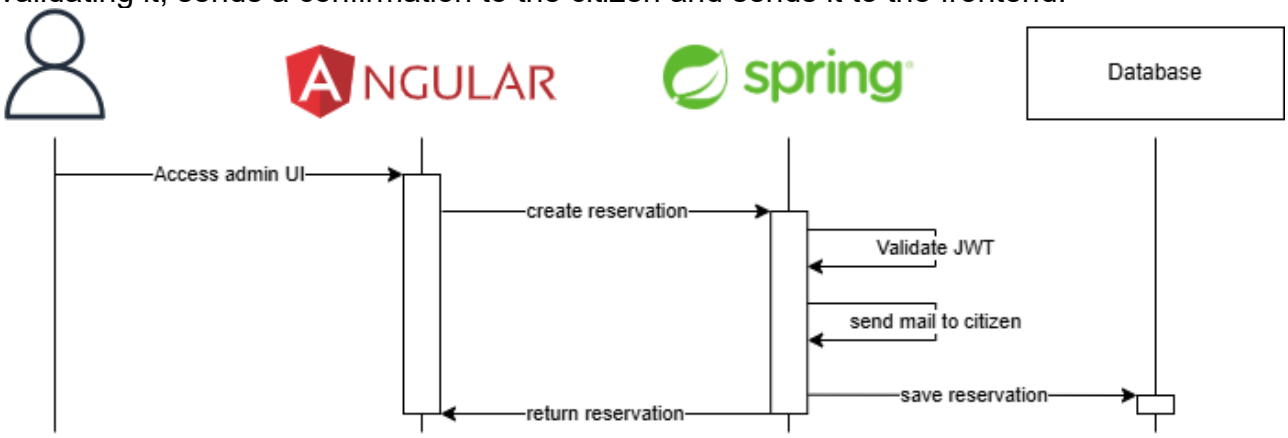
The admin UI sends a request to the backend.
The backend processes the request, enables the hotspot room in the database, allowing reservations, and sends a confirmation to the frontend.

- **Retrieving booking for a room**

The admin UI sends a request to the backend.
The backend retrieves the list of hotspot and rooms from the database and sends it to the frontend.

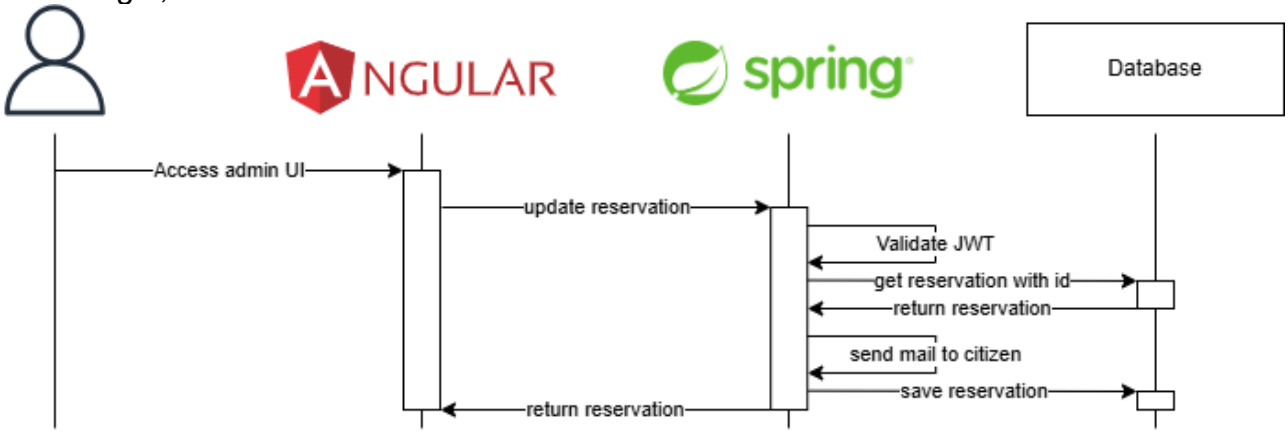
- **Creating booking for a room**

The admin UI sends a request to the backend.
The backend processes the request, stores the booking of room in the database after validating it, sends a confirmation to the citizen and sends it to the frontend.



- **Updating booking for a room**

The admin UI sends a request to the backend.
The backend processes the request, stores the booking of room in the database after validating it, sends a confirmation to the citizen and sends it to the frontend.



- **Deleting booking for a room**

The admin UI sends a request to the backend.

The backend processes the request, deletes the booking of room from the database, sends a notification to the citizen and sends a confirmation to the frontend.