

Draft Project Specification

Technical Report - Project specifications

GetARoom

Course: IES - Introdução à Engenharia de Software

Date: Aveiro, 5th December 2021

Students: 98475: Rodrigo França Lima

98262: Martinho Tavares

98157: Miguel Monteiro

89123: Tomás Candeias

Project abstract:

GetARoom is a web-based application with the focus status report on infrastructure, allow users to request a room that can accommodate a specific number of people, allow analysts to query specific logs, allow security management to restrict access to certain people or to get status updates on specific places.

Index:

GetARoom

Index:

1 Introduction

2 Product concept

Vision statement

User stories

3 Architecture notebook

Key requirements and constraints

Architectural view

Module interactions

1 Introduction

This paper has the goal of analysing the requirements for the development of the GetARoom application. This will be done within the scope of the IES course.

GetARoom is a web application that aims to simplify the search of rooms to use with a group in a given infrastructure and to simplify data gathering of events in each space within scope.

The main concepts of the application will be explore - the analysis of Use Cases, creation of Personas and Main Scenarios, developments of User Stories, main architecture and future goals for the application.

2 Product concept

Vision statement

Our system is designed for a variety of people, from students looking for a free room to work in a group on a project, to security personnel making sure no one enters rooms they are not allowed to, to analysts looking for more data about movement in specific rooms or departments. First, we want to give students the ability to see how full each room is and book a room for a specific number of people. Second, we want to give security staff the ability to see the status and history of entries in each room, blacklist people for certain rooms, and receive an alert when someone tries to enter a room that is not allowed. Finally, we would like to provide analysts with as much information as possible, from the yearly entries in each room to its current status.

The main goal of this project is to make it easier for analysts to maintain spaces by knowing what time of year they have less movement, if they need to be completely redesigned, or what time of day they are most likely to have no movement so they can make changes quickly. This gives security staff more control over certain spaces. And students save time searching for an available room.

User stories

- Student

- **Ask for a room for a work group (4 points)**

The user may declare the intent, through the application, to go to a certain room, which will be stored in the database. These aren't reservations, since the system doesn't block accesses. Instead, every other user of the application will know the current intents for each room, having thus an estimate of the future occupancy.

Tasks:

- Database tracking intents
- Intent rate representation on the application
- Intent management (add/remove intents, check current declaration)
- Suggestions based on predicted occupancy

- **See the state of multiple rooms in a certain building (2 points)**

Check the occupancy rate of the rooms in a specified building/floor.

Tasks:

- Search for department/floor/rooms and check occupancy

- **Analyst**

- **Check historic processed data and current state (3 points)**

Access to an API to obtain and process raw data from the database.

Tasks:

- Provide basic tools for data analysis and processing
- Show various representations of the different datasets

- **Security guard**

- **See current state of rooms (2 points)**

See the floor map of a certain floor in a department, with dynamic room coloring indicating occupancy.

Tasks:

- Statically map tracked rooms to the floor map

- Real-time data of occupancy on each room
- **See entry logs (1 point)**

Check the event logs for a certain room in a department's floor, with real-time access data.

Tasks:

 - Show list of current logs for a room
- **See graph of movement in a room (3 points)**

See historical event data in a specific room, in the form of tables or graphs.

Tasks:

 - Process and obtain structured graph data
 - Present the structured graph data
- **Receive notifications of a room being used by someone inside a blacklist (4 points)**

Ability to blacklist certain people for each room, which will allow the security guard to be alerted when a certain access should not have been made.

Requires precise person identification for accurate blacklisting and warnings, which is dependent on how each individual is identified.

Tasks:

 - Add/Remove people to/from a room's blacklist
 - Fire event when a person enters a room they are blacklisted in
 - Manage notifications (remove, add, mark as seen...)

3 Architecture notebook

Key requirements and constraints

The **system** has 3 types of users: student, security o analyst. All of them should be able to access the web application from their personal computers with a proper internet connection. Therefore, should be available in a steady and continuous way.

The **student** can search for a room to study in and also can see the room occupation.

The **security** is able to, as the student, see the room occupation, but also the movements in the rooms (entrances and leaves), add students to the blacklist and receive a warning when a certain blacklist student enters a room. Furthermore, he's able to see this information in graphics and more information with the logs.

The **analyst** has access to the GetARoom API for analysis and statistical purposes. He can see the raw data for the current events, yearly history and system status.

It is important to maintain authenticity within the system, meaning that a user only has access and power within his role in the system and nothing more.

Architectural view

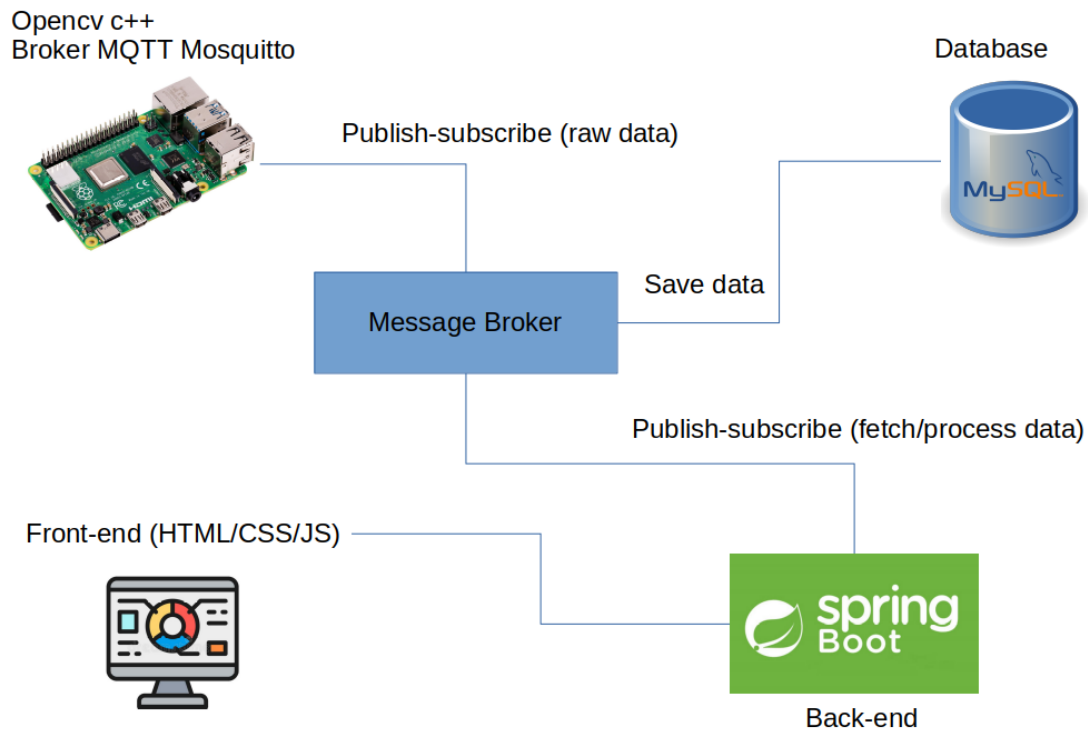
The architecture is mainly composed by the raspberry pi, message broker, database, front-end and back-end.

The Raspberry Pi receives inputs from the camera and the card reader, then sends the raw data to the message broker. Here the Raspberry Pi is the publisher and the Message Broker is the subscriber.

The Message Broker saves all the data, as received, into a MySQL database. Also sends the data to be processed by the back-end. Here the Message Broker is the publisher and the back-end of the web application is the subscriber.

The Back-end is developed in Spring Boot to communicate with the Front-end and give the information as needed.

The Front-end is going to fetch the needed information through an API.



Module interactions

A typical interaction, for example, would be a student searching for a room to study.

As we know, all the data retrieved from the Raspberry Pi, which is a MQTT Broker, is stored in the database.

In order to serve the front-end request to the API, the back-end will process the data, given by the message broker, to successfully fulfil this demand.