# 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:

# 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 explored - 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 all the rooms that can accommodate for the number of people in their group. 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 (1 points)**

    The user may declare the intent, through the application, of needing a room for a specified number of people. This request is then processed to return 10

rooms from the desired department that can accommodate the number of people the request states and the current state of occupancy of the rooms. This way, the user will be able to chose which a room.

**Tasks:**

- Suggestions based on occupancy and number of people on request

- **Analyst**
  - **Check historic processed data and current state (1 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
  - **See graph of occupancy of each room over time (3 points)**

    See event data of all rooms, in the form of graphs.

    **Tasks:**

    - Process and obtain structured graph data
    - Present the structured graph data
- **Security guard**
  - **See current state of rooms (3 points)**

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

    **Tasks:**

    - Statically map tracked rooms to the floor map
    - Real-time data of occupancy on each room
  - **See entry logs (2 point)**

    Check the event logs of all rooms, with real-time access data.

    **Tasks:**

    - Show list of all current logs
  - **See entry logs of a specific room (2 point)**

    Check the event logs of a room, with real-time access data.

**Tasks:**

- Show list of the current logs of a room

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
- **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

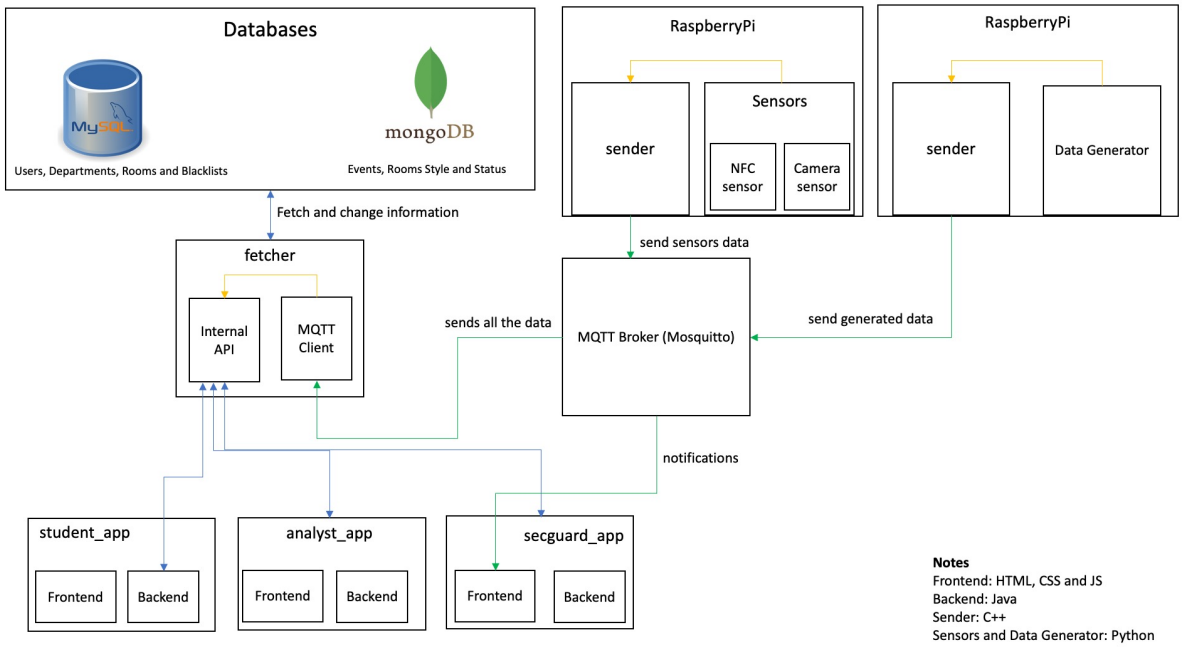
The architecture is mainly composed by the raspberry pi, message broker, database, fetcher and the apps.

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.

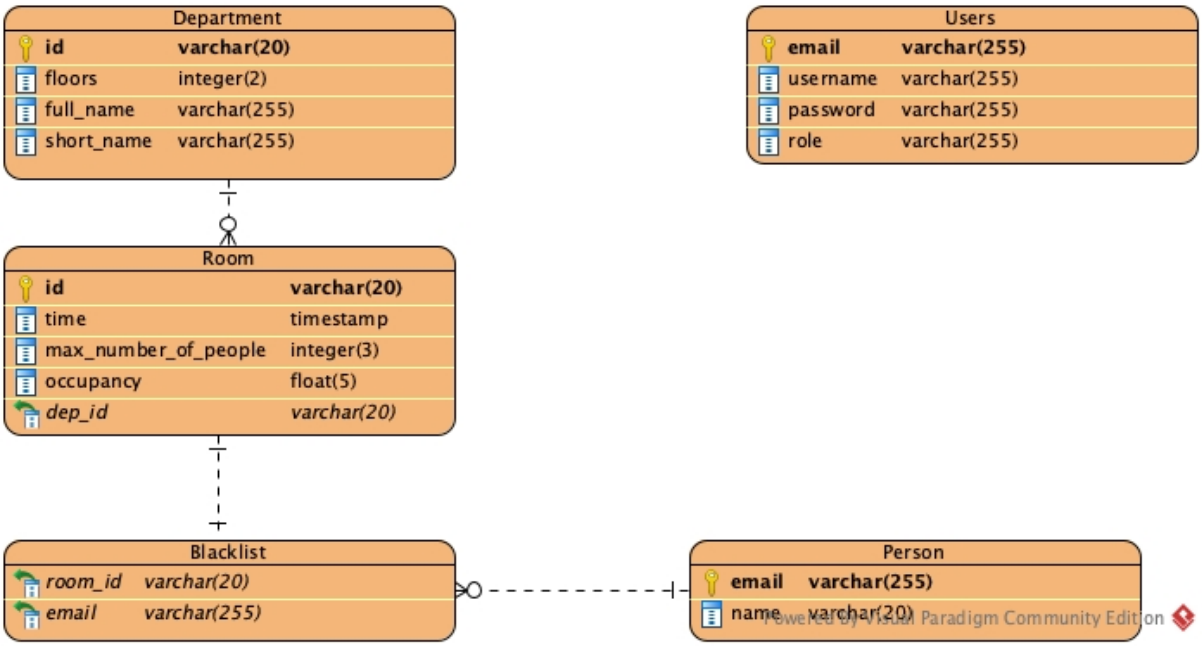
The Message Broker sends all the data to the fetcher and notifies the security app of new events.

The fetcher saves all the data into MongoDB and updates the MySQL tables, and allows the apps to have access to it through the internal API. This centralized approach to data access from the 3 apps was done for easier project development (database entities and repositories would have to be managed for each app, and be in sync) and this allows other future apps to easily utilize our data through a REST API, completely abstracting them from how it is stored.

The apps receive the data from the internal API and process it in their own way, the frontend of each app provides JavaScript scripts that utilize AJAX Calls and the Eclipse Paho MQTT library to obtain database data as well as data from the event streams.




The MySQL database is structured in the following way.




The tables ‘Department’, ‘Room’, ‘Blacklist’ and ‘Person’ are all connected due to their relation regarding each event, while the last table ‘Users’ is separated due to being used only for the users of the GetARoom apps.

## Module interactions

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

For the login, the student’s backend sends a POST request to the Internal API to verify the user credentials.

Furthermore, as we know, all the data retrieved from the Raspberry Pi is sent to the MQTT Broker, which sends the correspondent data to the MQTT Client on the fetcher. This one will store and update new information to the databases.

In order to serve the frontend request to search for a study room, the student’s backend sends a GET request to the Internal API to get all the departments, which will give the user the option to select the desired department to study in.

After that, the student’s backend sends a GET request to the Internal API to get all the rooms for the selected department. We take into consideration the number of people the study group has, and with that, we serve the frontend for the suggested rooms.