Course: IES - Introdução à Engenharia de Software

Date: Aveiro, 19/01/2021

Students: André Filipe Moniz Morais - 93236
Diogo Moreira - 93127
Eduardo Santos - 93107
Gonçalo Pereira - 93310
Óscar Sánchez - 101631

Project abstract: App to control your smart home devices

Index

1 - Introduction

With this project, we tried to simulate the experience of having a smart home.

The user can control every aspect of it in the web application, from listing the state of his devices to seeing detailed graphs of their functioning.

Project theme: Control of smart homes system.

Goal: a platform where you can control your appliances in a remote way making your life easier

Sensing layer: electrical appliances information, cameras and environment parameters like temperature or humidity will be collected

Data publishing: In each smart home there will be a device that will collect all the information the client would require and will send it to the rest api.

Processing: everything is controlled by spring, as multiple houses and customers can be connected to springboot.

Integration API: the exposed points allow the client to see his home or office information, and also allow him to use some of the devices that are connected to his smart home.

Web Portal: The web page allows the user to see the environmental parameters and the status of their connected appliances, while also allowing them to modify some of them such as the refrigerator temperature or turn off the oven.

2 - Product concept

Vision statement

mySmArtHome is an application where you can control your smart devices in an easy way and allows you to monitorize the status of your home when you're out

Personas

The design and development of mySmArtHome app was supported by 4 examples of Personas that can represent a future client

Jose

Jose is a fifty year old Alimentar Engineer working in Aveiro. One day his home was robbed and he wanted to install security cameras. On weekends he is a little lazy, he doesn't want to get up from the couch. He wants to feel his home is protected when he is away. He also wants to warm his home controlling the shutters.

Francisca Miles

Francisca Miles is a seventy year old woman that grew up in a village where she used to work in a farm helping her parents. After she finished highschool, she went back to the village to continue farming since that was what she loved to do. Now she has her own farm where she grows the food used to sustain herself and spends the day. Due to her old age, she has a hard time moving around her house. It is very hard for her to go home just to close the blinds or to make sure she has not left anything turned on.

Juan

Juan is a middle-age man who wants to monitorize the things that happen in his house. When he goes on vacation he wants to control the heating and the blinds of the house, and also see who rings the door´s bell. When wakes up in the morning, he doesn't have much time to prepare for work, so he wants to prepare his coffee at the same time that he takes a shower

Brendon James

Brendon James is a smart kid, under 16, who is remarkably lazy. He enjoys playing online videogames and going out with his friends. On the opposite, Brendon hates to do his chores, especially the ones who he thinks could be done remotely. Because he doesn't like to stop playing to do his chores, while also being lazy, Brendon wants to be able to do some house stuff through his smartphone. As a technology enthusiast, he hopes he can find a way to stop interrupting his games to do basic chores through his phone. Brendon hopes one day his parents can get a smart home, so he can do exactly that.

Main scenarios

- **Jose** wants to control the temperature of his house
- **Jose** wants to know the humidity of his house to regulate the shutters
- **Francisca** wants to be able to turn on/off the lights or see the blinds when she isn't at home
- **Francisca** wants to be able to watch every spot on the house without needing to go there.
- **Juan** wants to turn up the blinds at 8:00 while he is on vacation.
- **Juan** wants to see who rings the doorbell when he is out.
- **Brendon** wants to turn on/off the lights.

3 - Architecture notebook

Key requirements and constraints

The system requires a way to store device's data, smarthome's devices, system users logs for each device, a way to notify user about his devices, and a way to control the devices from anywhere in the world. In order to complete this we created 3 major subsystems:

1. **Client :** it is **scalable** to a mobile application, since it has been created in HTML / CSS / JS and there are some services that allow us to convert this project to APK / IPA / WPA, even for browser (like chrome, opera, firefox, safari) extension. After login, client receives a token that is stored in LocalStorage and that blocks unauthorized users from navigating to certain pages.
2. **Spring Boot / Persistence :** Developed in **Java** and **MySQL**, it is **responsible** to receive the client's request for login check, getting devices, turning devices status, storing logs, receiving notifications from the broker / process notification / send notification to client. Some client's requests require data from a specific device, some information that **isn't stored in the database**, like their status, get the type, hardcheck the devices in network, notifications… By the other side, it receives every 3 seconds, from the broker, information from device values. So **Spring Boot** can process the data and evaluate if the data is harmful or normal for that device. For example, if a termal device is at 99º it means it may be overheating.
3. **Broker :** Developed in python, it **simulates an hub** in which all devices connect. It is the bridge between the Spring Boot and the devices and controls the session in which devices are connected. It means that, if the broker is off, all the devices go off and they need to be searched again. The broker finds out automatically which devices are in the network by testing all ips in the network. Periodically (3 seconds), it sends to Spring Boot the current value of devices. **Notifications** are "stored" in the queue so, if spring does not get any notifications at a moment, it can be accessed after that accessing the queue.

This is an universal application, it means that it is **scalable** and every new

broker can connect to Spring Boot, and one single application can connect to any home depending on the user's authentication.

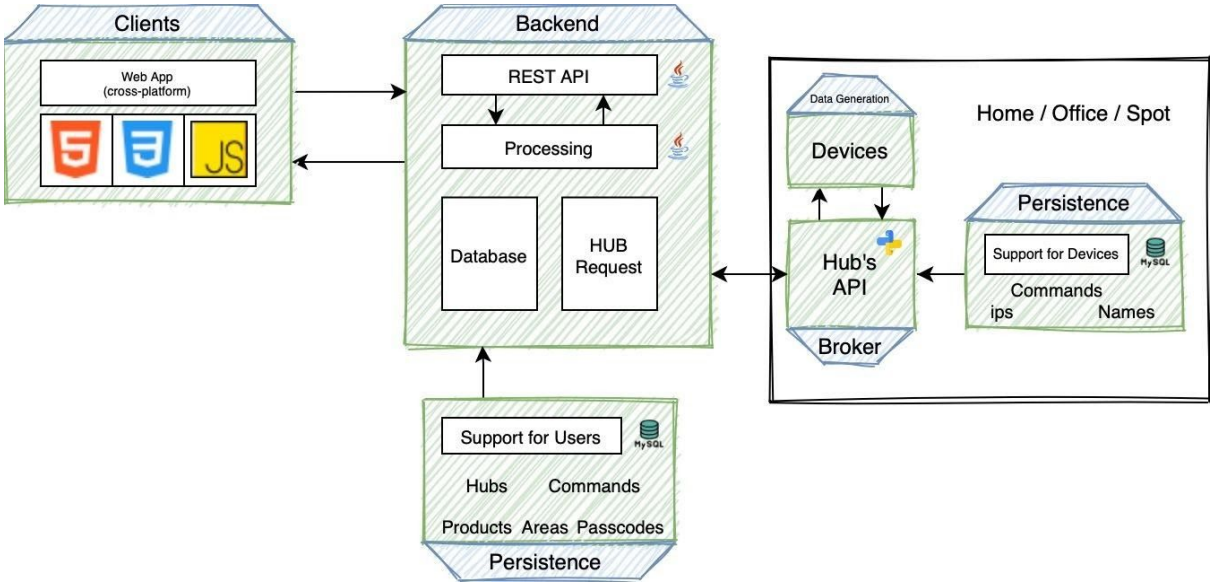
The VM does host client modules however it does not need a server to be hosted, and that is the advantage (scalability to mobile apps).

There is a module called **TapoController** (functional) that is based on the VirtualController module and allows us to control the **real devices**. It takes longer to find all devices because of real connectivity, but in the end **all devices are found**.

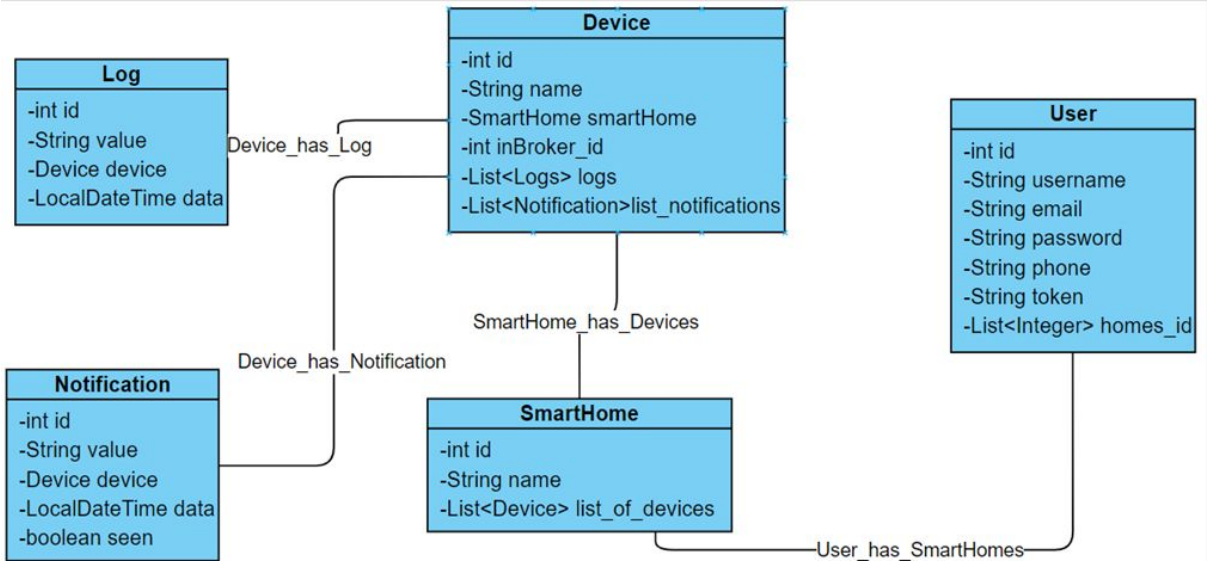
There are some **key requirements** and **system constraints** that have a significant bearing on the architecture. They are:

- Real devices must be from TP-Link subsidiary 'Tapo'
- Real devices are not implemented on Spring Boot or Client
- Any device must be able to be controlled from anywhere
- It needs internet connection both in Client as in Back end
- Each user has a different username and a chosen password
- Users do not need to login again after the first time unless they are trying to connect from a different device
- There are some problems in syncing the Spring Boot and the Broker, related to rabbitmq responses.

Architectural view



Database Model



User: Represents the client's information. It´s connected with the smart home table through an Element Collection(homes_id), where the user can store and localize his smart home devices.

SmartHome: A simple table that allows the user to see and modify the status of his smart devices in a safe way.

Device: Represents a user´s appliance. It can collect information from the environment in a passive way or it can't interact with it in an active way if the device allows it and the user wants to use it.

Notification: Message that receives the user when the device status has changed.

Log: Identifier of the notification.

4 - DevOps

For Serving Static Content, we used NGINX. This allowed us to access our frontend part.

Regarding the DevOps, we have a docker-compose.yml. Running "*docker-compose up --build*", everything is up and we can access the website from **192.168.160.210.**

We have an already created user for testing purposes:

- **Username**: admin
- **Password**: admin

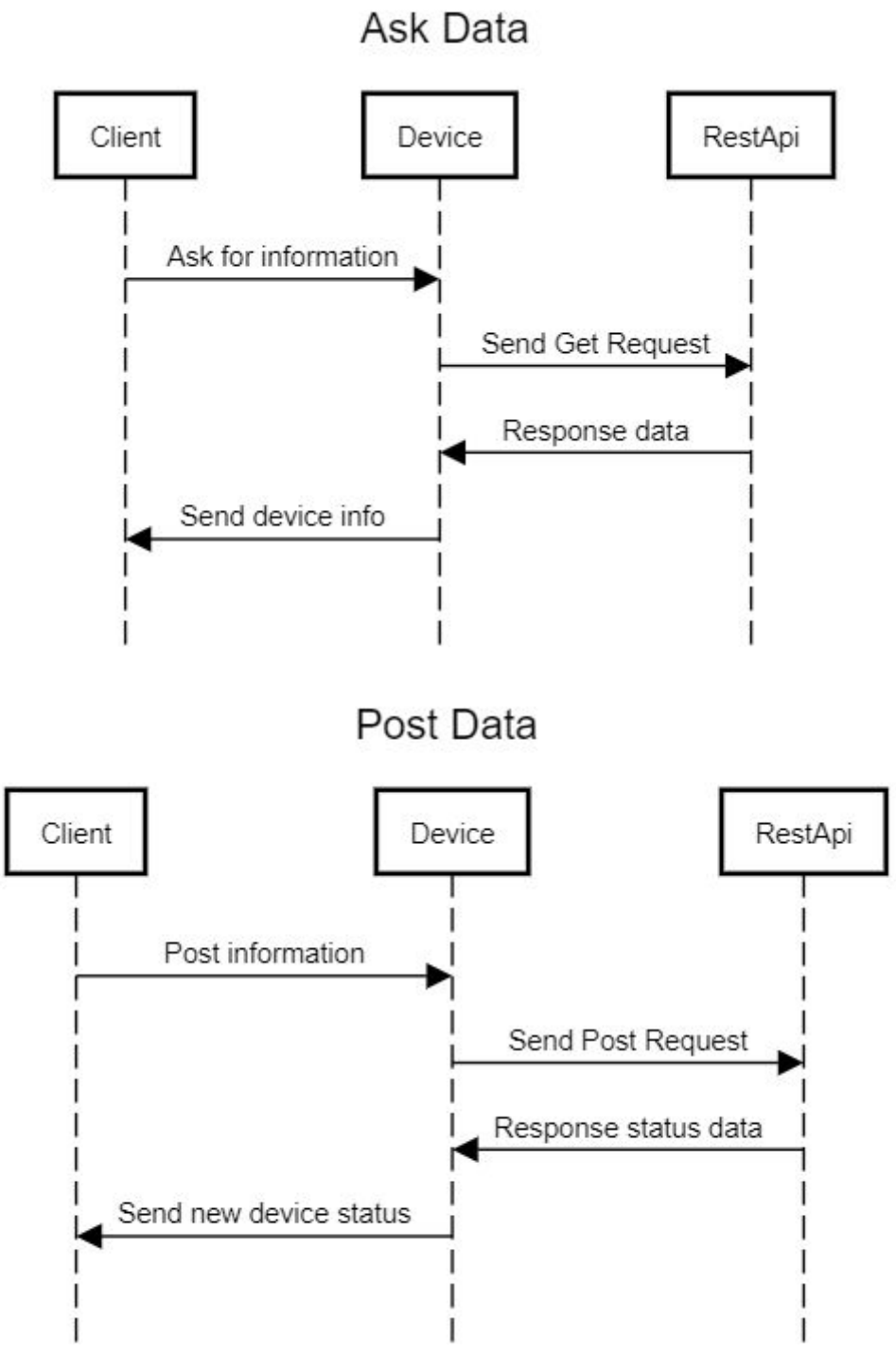
To run the devices, both **rpc_server.py** and **producer.py** from **broker/** need to be running.

5 - Project Management

For the project management, we used ZenHub, a platform which can be integrated with GitHub, allowing us to use Agile methodology to manage the project.

Along with this, we had 4 iterations, each one with a specific number of issues, those were decided by all the group members.

6 - Information Perspective



7 - API Documentation

Regarding the API documentation, we used Swagger (https://swagger.io/) .

After running the docker-compose command, the documentation becomes available on:

http://localhost:8080/swagger-ui.html

8 - Conclusion

Even though our project was incomplete, we managed to learn a lot while working on it.

There are things that could be improved/implemented in a future work, such as, for example, having the paging implemented, as well as options to filter data from the alarms logs.