LEI

Project specification report

Home Monitor

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, January 11, 2021
Students:	93080: Dinis Cruz 92963: Duarte Mortágua 93019: Lucas Sousa 93456: Tiago Oliveira
Project abstract:	Web application to monitor and control light presence temperature, humidity, movement, flame presence and water level inside your house.

Index

Introduction

This project was made in the scope of Introdução à Engenharia de Software.

Main objectives:

- Develop a product specification, from the usage scenarios to the technical design;
- Propose, justify and implement a software architecture, based on enterprise frameworks;
- Apply collaborative work practices, both in code development and agile project management.

We decided to work in a Home Monitor, a Web application to monitor and control light presence, temperature, humidity, movement and flame and flood detection inside a home.

After deciding the project's theme and objectives, we distributed the team roles:

- Team manager: Duarte Mortágua
- Product owner: Tiago Oliveira
- Architect: Dinis Cruz
- Devops: Lucas Sousa
- Developers: The whole team

Then we finally created the git repository and defined which goals were most important to start implementing.

Product concept

Vision statement

Our product consists in a web application that receives data from a raspberry placed in the clients' house. It was supposed to have several distinct sensors but we only got hold of the movement sensor. The others such as: light presence, temperature, humidity, flame and water level are virtualized through the raspberry as if they were actually built in the clients' house (the data-flow is the same).

The application can be accessed through the browser and the user will be able to:

- Control the light presence, temperature, humidity, flame presence, movement and water level in his house;
- Consult the feedback from each sensor in a scope of time.

Our product makes the users' lives easier because they can check the status of their houses in an app whenever they want and without being even in their houses.

As a team of four people finding use cases was a relatively simple task. Thus, we came up with these ones:

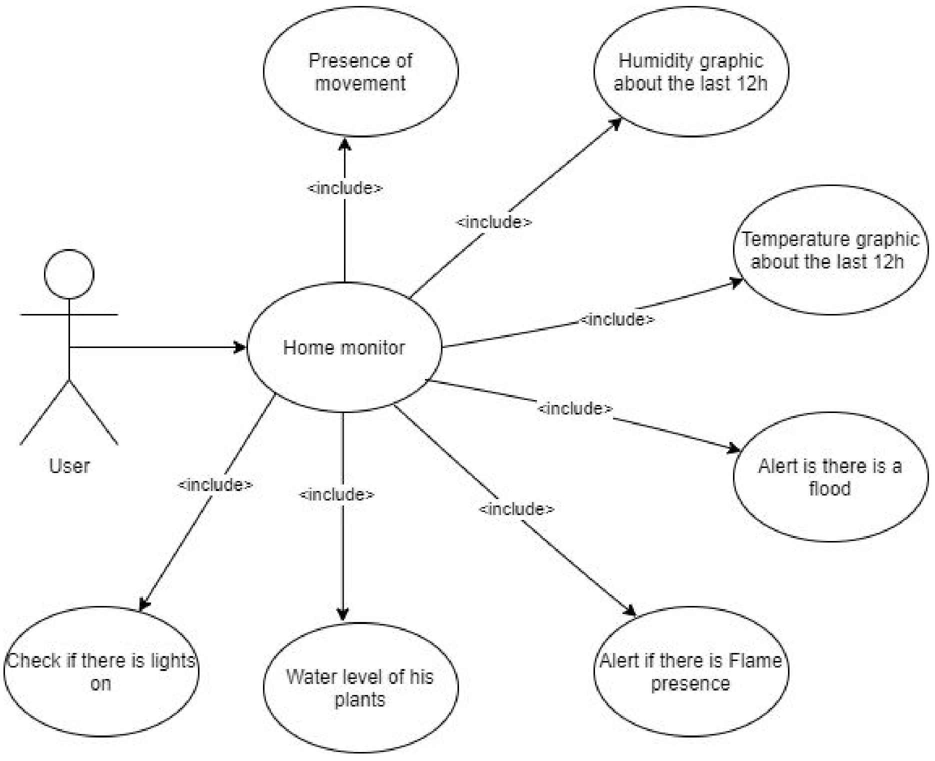

Presence of movement
Humidity graphic about the last 12h
Temperature graphic about the last 12h
User
Home monitor
Alert is there is a flood
Check if there is lights on
Water level of his plants
Alert if there is Flame presence
<include>

Diagram 1: Use cases

Personas

Ana:

Ana is a 27 year old woman that lives in Aveiro and works as a nurse in Hospital da Luz. She likes spending her free time reading and spending time with her family and friends.

She started living alone for about 3 months. Until then, she was sharing a house with a friend that worked at home and so she had always someone taking care of her house.

In order to feel more satisfied, secure and in control of her house, Ana is searching for an application that tells her if everything is going well inside her house like if the temperature is fine for her pets and plants.

António:

António is a 33 year old man that lives in Guimarães. He works as a manager and loves new technologies. He's a reasonable person and he's eco friendly (friend of the environment).

A water pipe broke in his previous house and as soon as he got from his work there was a big flood and there was so much leaked water that all his books were gone. He is searching for an application that can alert him if a flood happens while he is gone.

João:

João is a 28 year old man and lives in Coimbra. He works in a software development company as a security engineer for about 8-10h a day. As he passes so little time in his apartment and he already experienced a burglary while he was gone, he decided to search for an application that alerts him in case there is movement in his house.

Main scenarios

First scenario:

Ana wants to check if the temperature and humidity in her house are at a good level for her pets. Having an internet connection, accessing our application and doing her login, there will be options for each room of her house with data of the temperature and humidity.

Design notes: After she logs in, there will be several options, each one will represent a room at her house, as soon as she clicks on one of the options there will be shown graphics about the temperature and humidity in that room.

Second scenario:

António wants to check if there is a flood going on in his house, so he can stop it before his books get damaged. Having an internet connection, accessing our application and doing his login, there will be a flooding alarm box: if it is green, it means that everything is normal, if it's red, it means that there is a flood going on.

Dev notes: Data that will be shown should be the one read by the sensor (in this case simulated).

Design notes: After he logs in there will be a button on the left side of the page, if it is red, it means that there is a flood going on, if it's green, it means that there is no flood risk at that moment.

Third scenario:

João wants to check if there is someone in his house so he can report to the police before he gets robbed. Having an internet connection, accessing our application and doing his login, there will be options for each room of his house if he checks the entrance door and the exit door he can see the last time (in hours and minutes) that someone entered through that door.

Design notes: After he logs in there will be several options, each one will represent a room at his apartment, as soon as he scrolls down until he sees entrance and exit doors, there will be shown the last time someone entered through.

Architecture notebook

Key requirements and constraints

Key requirements:

- Simulated data needs to be consistent and close to the reality;
- Data needs to be dealt with carefully and sent in a way that keeps its integrity;
- In case the system shuts off we need to recover all the data using a database to store it;
- To access the data we need an API to make it easier to fetch the data.
- Deploy to VM (Google Cloud)
- Run build tests (Github Actions)

Constraints:

- It is expensive to have a raspberry with a sensor in each room of the house, so we will only have 1 raspberry;
- We could only obtain one physical sensor, the rest of the data is virtually generated in the rasp;
- Having several clients, live data updates can overcrowd the API;
- Having only one endpoint for the brokers in the backend. If several raspberries were actually being used, one listener could not handle all the connections.

Key requirements and system constraints that lead us to our architecture:

- Use more than one sensor? How to generate realistic data?
- What is the best option to store data? Which DBMS to use?
- How to deal with failures?
- Migrate it to a mobile app or not?

Architectural view

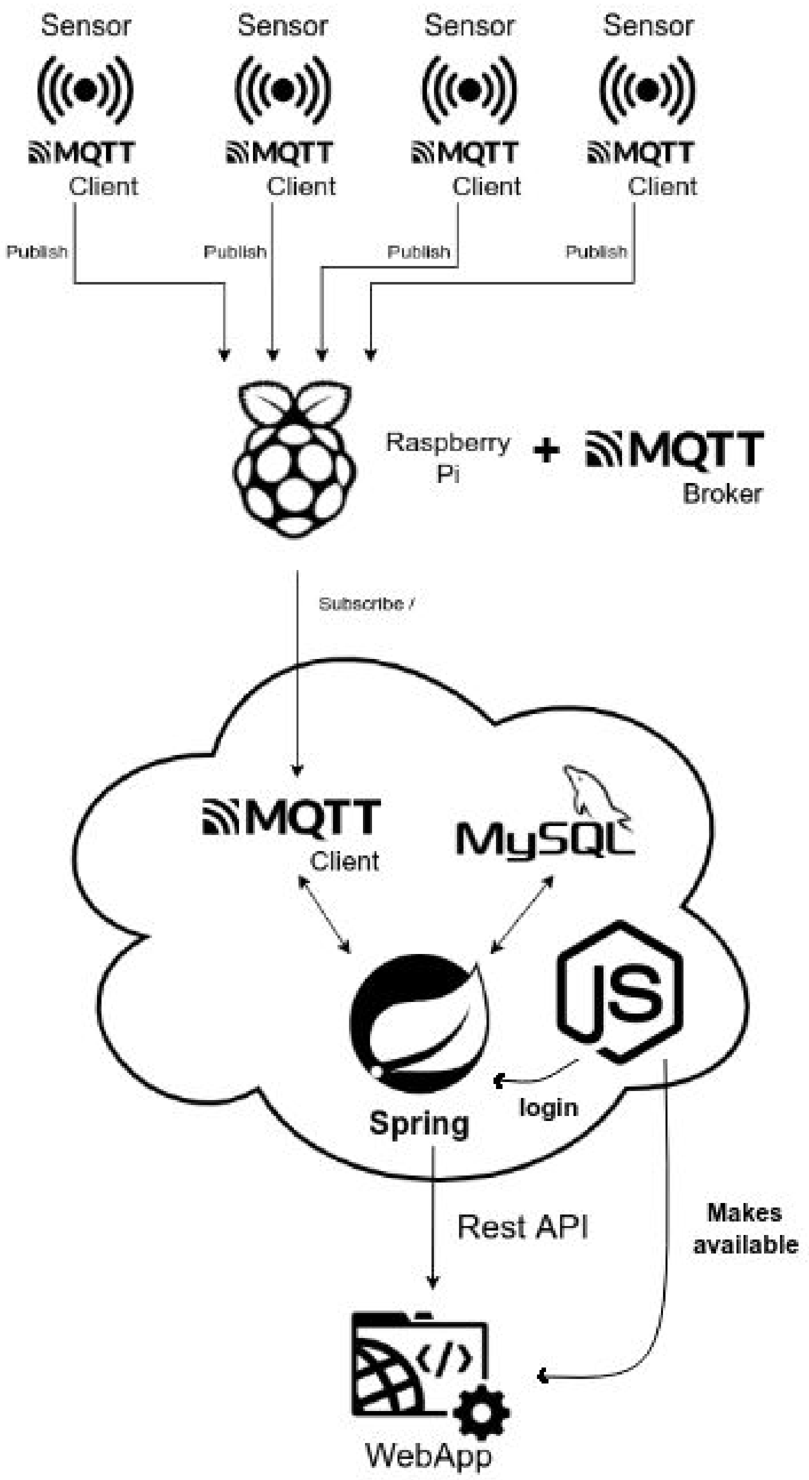


Diagram 2: Base architecture

Our architecture is divided in:

- Frontend;
- Data generation;
- Backend;
- Data store.

Frontend:

The app shows the user interface with the data, graphics with stats and a few more things that are directly related with the users' house. The frontend will be made available to the client through node.js running an express server. This server handles access to the dashboard page according to the user login. The client will retrieve data through regular API calls and render it through React components.

Data generation:

Supposedly we'd have sensors plugged in a raspberry and spread around the users' house sending data such as light presence, humidity, temperature, movement detection and flame and flood detection. In reality, the movement sensor does work, the rest of the sensors are simulated in the raspberry - we wrote python scripts that generate realistic data continuously.

The data is then sent to the backend to be stored and fetched through the rest API.

Backend:

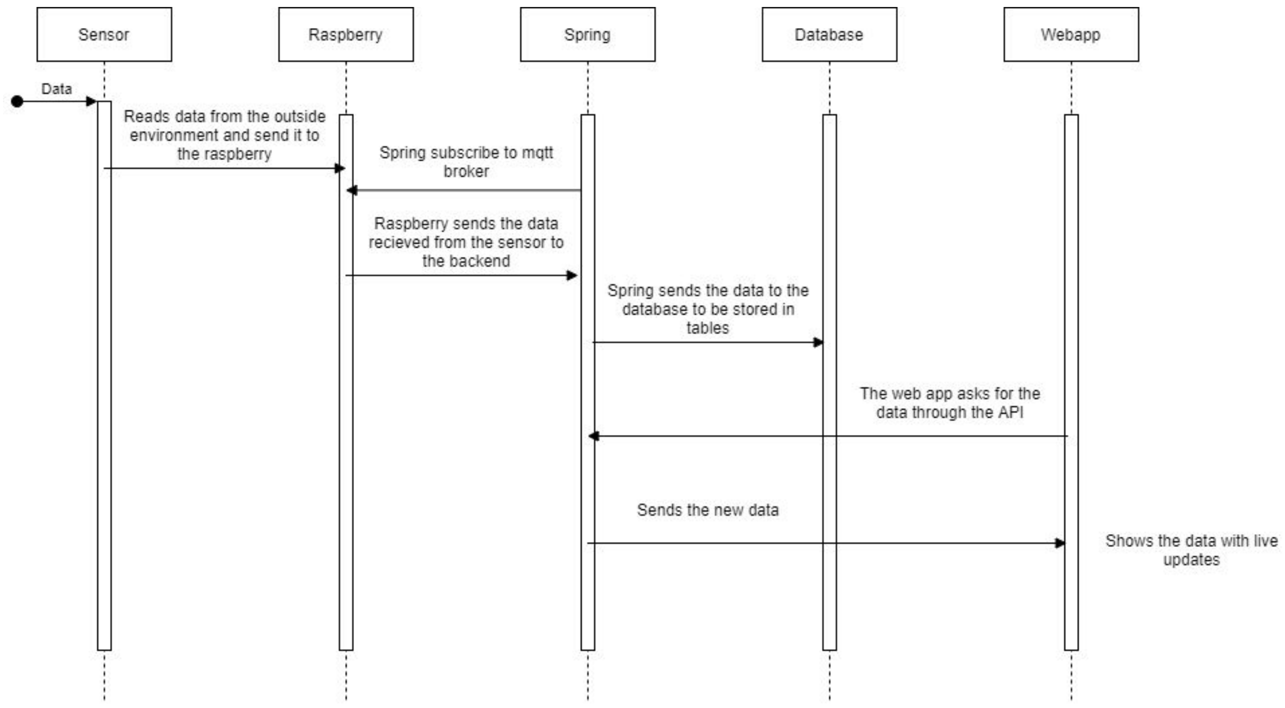
The backend, manipulated through spring, has a message broker which is subscribed to the raspberries. When a message comes, the listener loads it into the database. The backend handles the frontend API requests too, providing persistence sensor data to the client. The communication with the persistence is done with JPA.

Data store:

The data is stored in a MySQL database. It stores the users, the sensors and all the values received. This layer is in constant communication with the backend, and is manipulated through the JPA.

Module interactions

Each sensor is responsible for reading a determined type of data and then for publishing it to the raspberry. The raspberry gets the data and processes it. Then, the raspberry sends it to the backend (subscriber). Data will then be processed by spring in the backend and then stored in the database. Once we have the data stored in the database, it will be available through the Rest API to our application, which will show live data, updated in each 5 seconds.



Sensor
Raspberry
Spring
Database
Webapp
Data
Reads data from the outside environment and send it to the raspberry
Spring subscribe to mqtt broker
Raspberry sends the data recieved from the sensor to the backend
Spring sends the data to the database to be stored in tables
The web app asks for the data through the API
Sends the new data
Shows the data with live updates

Diagram 3: Sequential diagram about module interactions

As it is expected this path is a loop that is repeated every 5 seconds until the app is shut down.

References and Resources

API documentation:
https://app.swaggerhub.com/apis-docs/d2987/homemonitor/1.0.0

GitHub repository and Project:
https://github.com/l-sousa/ies-home-automation

Live Website Demo:
http://35.246.39.11/

Resources used to learn from scratch some of the concepts implemented:

Tutorial: Intro to React
https://reactjs.org/tutorial/tutorial.html

Node Express Guide:
https://expressjs.com/en/starter/installing.html

Google Cloud Documentation:
https://cloud.google.com/docs

Git-flow Cheatsheet
https://danielkummer.github.io/git-flow-cheatsheet/index.html

All the contents used and learned in classes about Spring Rest API, JPA, Docker and Project Management.