Technical Report - **Product specification**

LockedIn

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 3 de janeiro de 2022
Students:	102491: Raquel Paradinha 103234: Paulo Pinto 103341: Miguel Matos 104142: Tiago Carvalho
Project abstract:	*LockedIn* is a web based application that offers the means to manage a prison and its internal logic.

Table of contents

1. Introduction

This project was made in the context of our Introduction to Software Engineering course. In it, we propose, conceptualize and implement a solution for a prison management system, or, as we call it, *LockedIn*.

This document serves to describe the product requirements and project structure, within the scope of the course. Personas, use cases and main scenarios are explored in the following chapters. The chosen architecture is also described.

Each member of our team has a defined role, in order to split the workload in an efficient manner:

- Raquel - *Team Manager*
- Paulo - *Product Owner*
- Miguel - *Architect*
- Tiago - *DevOps*

2. Product concept

Vision statement

LockedIn was developed with the intention of improving the internal management of a prison.

This system was designed to be administered by the prison warden (a super-user) and used by the guards (regular users). It will provide real-time data about relevant aspects of a prison, like cell block occupation or inmates location. It will also serve as a management hub for the prison warden, where he can manage guards' permissions within the system, from the system itself.

With this, we can easily supply guards with the information they need to fulfill their jobs, and the means for the prison warden to handle all his workload.

Personas and motivations

Mário Santos is a 49 years old male prison guard, living in Candieira with his wife and daughter. He is generally an extrovert person, who enjoys spending time with his friends and family, despite also being a very focused person.

Inside the prison, he is responsible for many functions, especially being certain that his prison's inmates are doing as they should. Due to his crippled knee, normally, he stays in charge of supervising prison rooms/spaces to ensure respect and protection to everyone.

Even though he tries really hard, sometimes it's difficult to count everyone who is and should be in the room, making it an almost impossible mission for Mário.

Carlos Correia is the prison warden. He is 41 years old and divorced, with two daughters (13 and 15). He splits custody 50/50 with his ex-wife.

He holds himself in very high regard, as well as his work; despite its nature, he still believes in doing it well and fairly.

In his free time, he likes to read and bake with his daughters. At work, he is the one that makes sure that everything goes smoothly in the prison. He approves, delegates, and decides.

Main scenarios

- In one of his shifts, Mário received a riot notification in the canteen, so he interfered, stopping it and putting the responsible inmates in the solitary. Next, he opened the system on his phone to report this decision.

- Carlos just hired a new prison guard, and needs to provide him with access to the facilities and system. For this, he opens his application, logs in, and sets up the new guard account.

User Stories

1. As a prison guard, I want to have access to who are the prisoners in the room that I'm responsible for; I also want to be able to report to the warden when something out of the ordinary happens.

2. As the warden, I must have access to everything, I must be able to add new guards and give them access to their respective accounts with the permissions they need as well as control all the sensors of the prison.

3. Architecture notebook

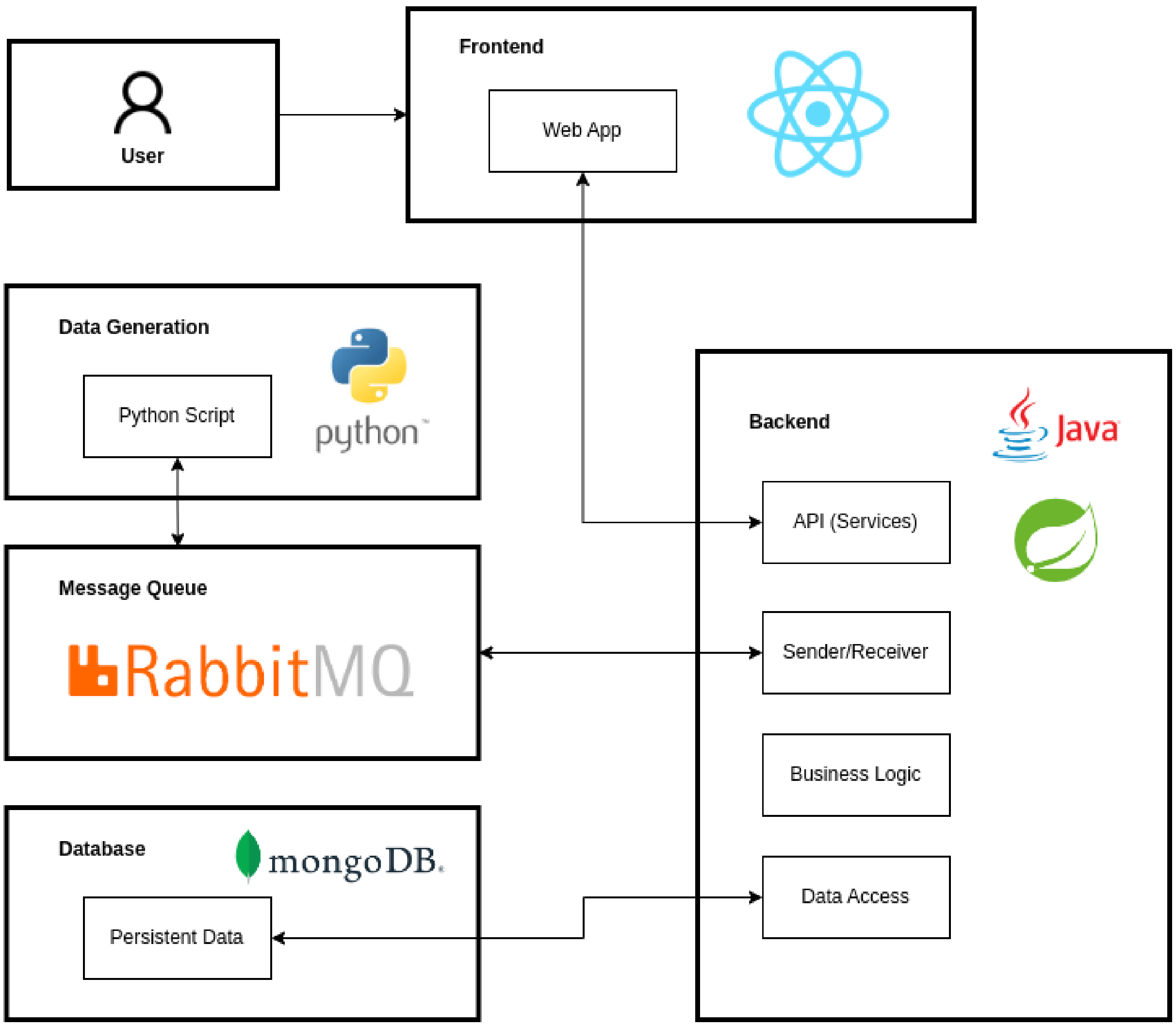
Key Requirements and Constraints

- The guards should be able to have access to the WebApp at any moment while using the prison computers.
- The guards should be able to see where they are going to work.
- The guards should be able to change their personal account information.
- The warden should be the only one able to manage all accounts (create, update and delete guard and prisoner accounts).
- The guard account parameters should be name, email, phone number, birthday, area and password.
- The warden account parameters should be name, email, phone number, birthday and password.
- The prisoner account parameters should be name, email, phone number, birthday, sentence starting day and final day.
- The prisoner account has a read only use for the guards and warden, the prisoner will not be able to use the app.
- The warden account is the admin and must have access to the information of every guard and prisoner.
- The guard should have access to where each prisoner is, for example, the guard can see the names/id of every prisoner in the solitary.
- The sensors on each room should detect a prisoner and update their location.

Architectural View

The project architecture will be split in 4 main parts:

- The data generation layer, which is responsible for generating data about the application context, and will send it through message queues to the backend for processing. The data generation will be simulated using python scripts and the messages queues will be implemented using RabbitMQ;
- The database, which will store all our persistent data. It was first thought to be implemented in MySQL, using a relational-type database. However, after taking a closer look at our data classes, we realized that a documentational database would be more fitting, so we decided to go with MongoDB;
- The backend layer, responsible for handling all internal logic. It fetches and stores from the database, receives data from the message queues, processes it and displays it in the frontend, through Web Sockets. It will be implemented using Spring Boot with Spring Security;
- The data presentation layer, or frontend, which will communicate through the API - part of the backend - to fetch the data it should present. It will be implemented using React.


User
Frontend
Web App
Data Generation
Python Script
python
Backend
Java
API (Services)
Message Queue
RabbitMQ
Sender/Receiver
Business Logic
Database
mongoDB.
Persistent Data
Data Access

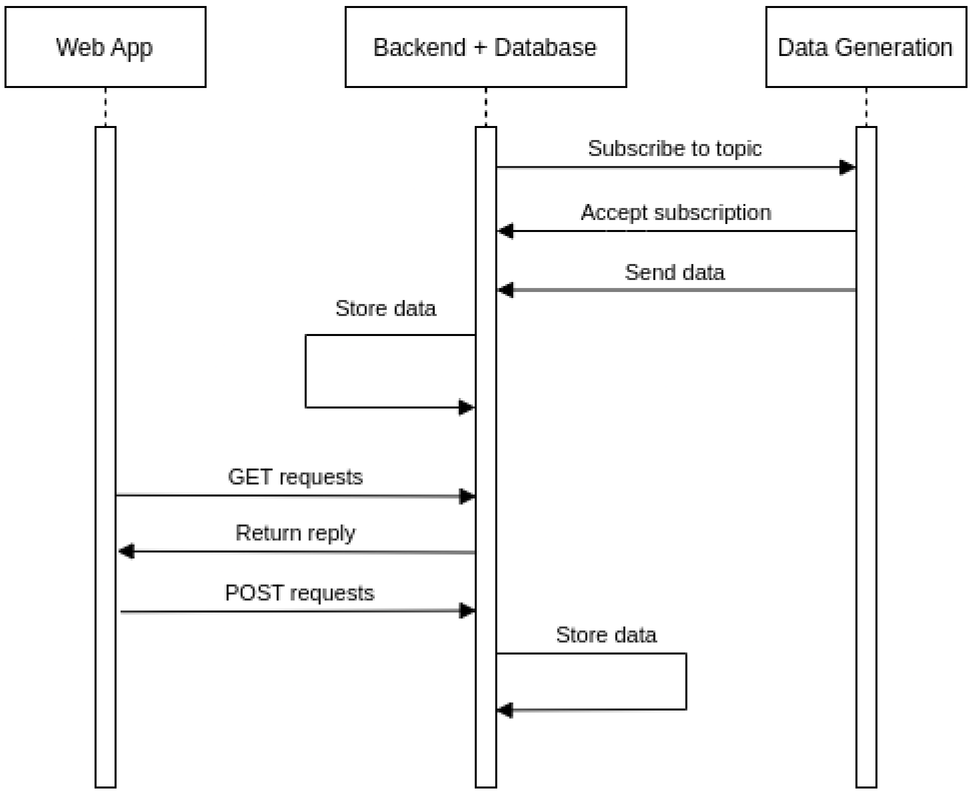
Module Interactions

The interactions between modules respect each module's responsibility and role in the app. For example, for the login, the frontend posts the login credentials to the API. The backend then checks for the authenticity of said credentials, using the information stored in the database.

Assuming the login went as expected (valid credentials), the Web App will then display the respective dashboard according to the type of the logged in user (guards and wardens have different dashboards, tailored to their needs).

The data generation layer generates data according to the prison context, which is passed to the backend for processing and storage.

Most of the module interactions can be summed up with the following sequence diagram.


Web App
Backend + Database
Data Generation
Subscribe to topic
Accept subscription
Send data
Store data
GET requests
Return reply
POST requests
Store data