Technical Report - **Product specification**

Health Manager

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, 31/12/2022
Students:	102751: Alexandre Lima Gazur 102885: Daniel Ferreira 103078: Ricardo Pinto
Project abstract:	Manager web app that lets you easily check on and manage all your patients; add, edit or remove them, view their details in real time; app regularly calculates patients' health state from attributes like heart rate; sort by endangered patients, receive alert when our app deems a patient just became endangered; many forms of data visualization including agregation in graphs.

1 Introduction

With the knowledge about Software Engineering and OOP that we have learned in this course, we made a multi-layered web app that proves our skill and knowledge in these subjects.

2 Product concept

Vision statement

Our web app is used to easily and intuitively manage patients in a hospital or clinic or similar and check their state and data, both past and present.

It can be very hard to keep track of the many patients in a hospital/clinic. Professionals that work there are often busy and need to quickly manage and check on all the patients regularly.

initially, our prototype had a page for each patient, where all their data could be seen. But later, we added a dashboard that displays all of them and their data in a summed up way.

We believe the uniqueness in our system is that the app regularly, accurately and automatically calculates patients' health state (e.g. "Unhealthy") based on attributes like heart rate, body temperature, respiration rate, etc. Another feature, but not as unique, is an immediate alert notification when our app detects a patient just became "Very unhealthy" or "Unhealthy".

Our teacher in IES pratical classes helped a lot, through regular reviews of our work/progress and suggestions.

Personas and motivations

Doctor Steve is a 45 year old licensed medical professional who wishes to be not have to spend a lot of time reviewing a patient's blood pressure, heart rate, etc or simply checking on them regularly.

Motivation: Steve wants to view a patient's health state, past and present, in a quick and easy way, without any of the hassle. He also wants to be able to have more complex ways of viewing said data, like data agregation on a graph, to observe the progress of patients. He wants to remove patients no longer in the hospital with a single click and quickly add new ones.

Main scenarios

- Doctors should be able to see given the patient's current health details.
  Steve, a doctor, enters an appointment with a regular patient of the hospital, plagued by certain chronic illnesses. In order to check his current blood pressure to his past measurements, he accesses that patients' page which shows the current/latest data and all past data.

- Doctors (and system administrators) should be able to add, edit and remove patients from the system records.
  Steve, a doctor, finds himself in an appointment with a new patient who has never been to the hospital. As there is no previous history with him, he finds it necessary to register this patient in the system, in order to store and later use his medical data. Later that day, a patient is treated and can now leave the hospital, so Steve quickly removes them from the system with 1 to 5 clicks.

- A doctor should be able to be quickly check if any patient is in danger (for example missing oxygen).
  Steve accesses the web app's patients dashboard, which sorts all patients by the ones that our app considers currently in danger.

- Doctors should be immediately notified when a patient needs assistance.
  Our web app calculated that a patient just became endangered because his heart rate and blood pressure dropped too much, so the doctor is immediately notified about it.

3 Architecture notebook

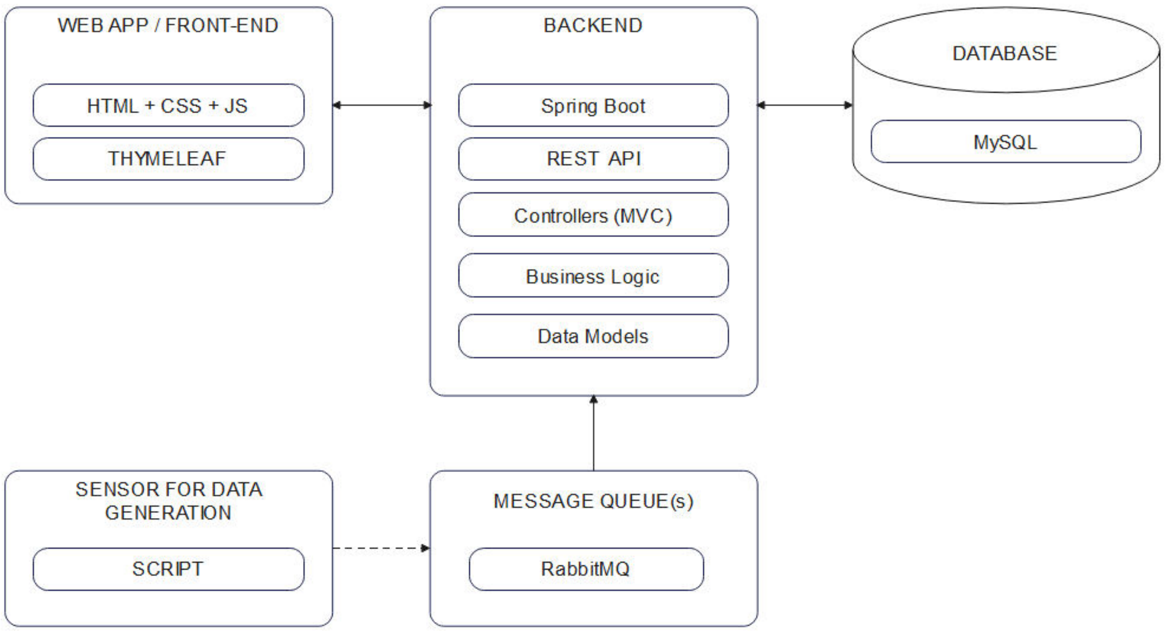
Key requirements and constrains

- The application should be accessible via web.

- The application should always be available. It cannot fail when traffic becomes too large, nor when there's too much data in database to process, and it must have redundancy and be robust against errors.
- Given the nature of the app, it is easy to fill the database with a lot of data, making queries slow. For example an heart rate record added every second will quickly add up to thousands of thousands of rows in database, so when it is time to retrieve/query them it'll be slow. This means we need a robust and fast DBMS configured with proper indexes.
- Doctors should have access to patient's data in various forms (table with all raw data, graphs with agregated data, etc).
- Something like a dashboard that displays all patients at once with their data summed up.
- The app should somehow immediately alert when a patient becomes unhealthy (e.g. accelerated heart rate).
- App must somehow regularly estimate how healthy a patient is, based on his health details (e.g. heart rate).
- The client/doctor needs to somehow be able to add, edit, remove patients.
- Robust REST API.
- The front-end must continue to work flawlessly while backend is processing received health data.

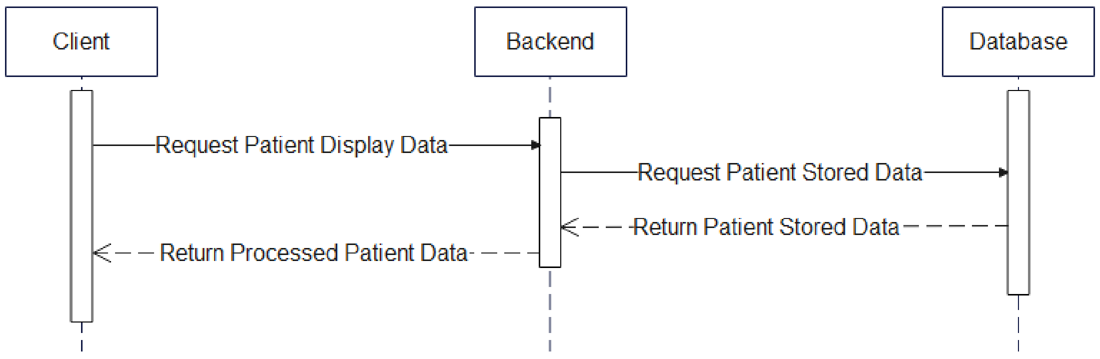
Architetural view

The architecture is composed of 5 parts: Web App (front-end), Back-end (MVC, business logic, manage API, listen to message queue to receive health data, store that data in database), persistent database where all patients' data is permanently stored and acessible at any time, sensor that generates health data (heart rates, body temperatures, etc) and publishes them in the message queue.

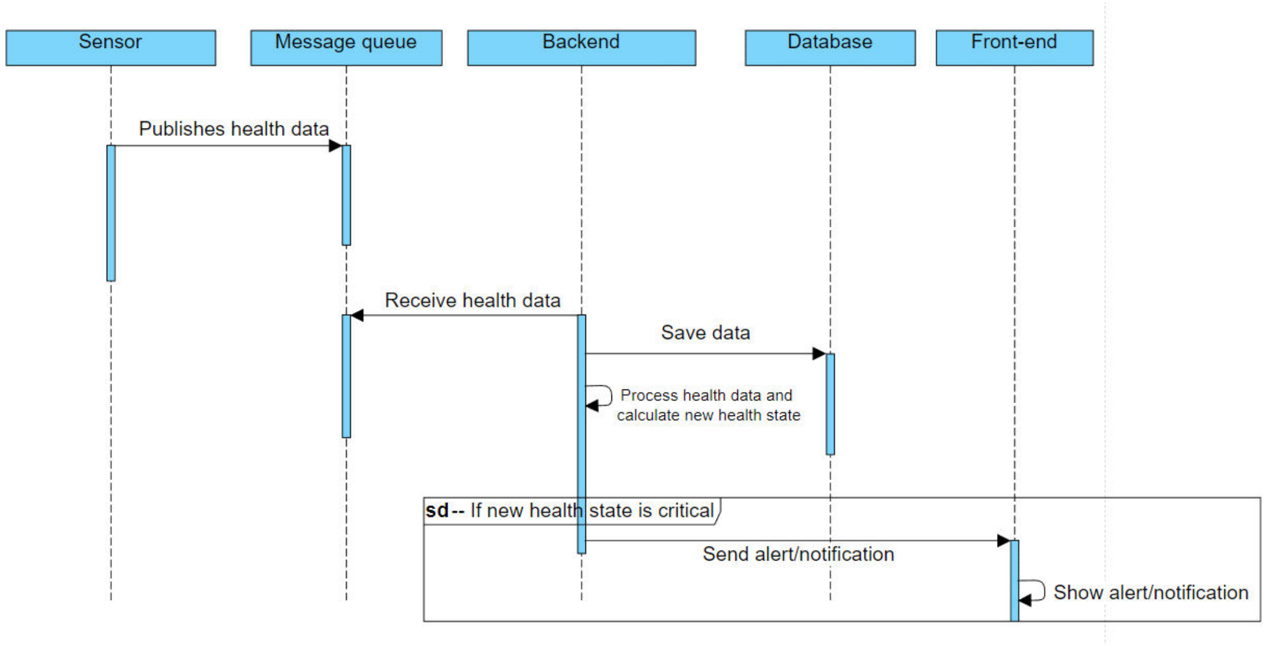


Module interactions

When home page, /patients page or /patients/{id} page or API is accessed, the backend queries the database for the necessary patient data, processes it, and forwards it to the front-end.


Client
Backend
Database
Request Patient Display Data
Request Patient Stored Data
Return Patient Stored Data
Return Processed Patient Data

The sensor generates health data (e.g. heart rate records) and publishes it to the message queue. The backend listens to that message queue to receive the health data. When it receives, it saves that new health data in the database. If, with that new data, the app deems that the patient is now endangered (health state calculation) (e.g. received an extremely high heart rate and respiration rate), an alert is sent to the front-end, which displays a notification alerting the client.

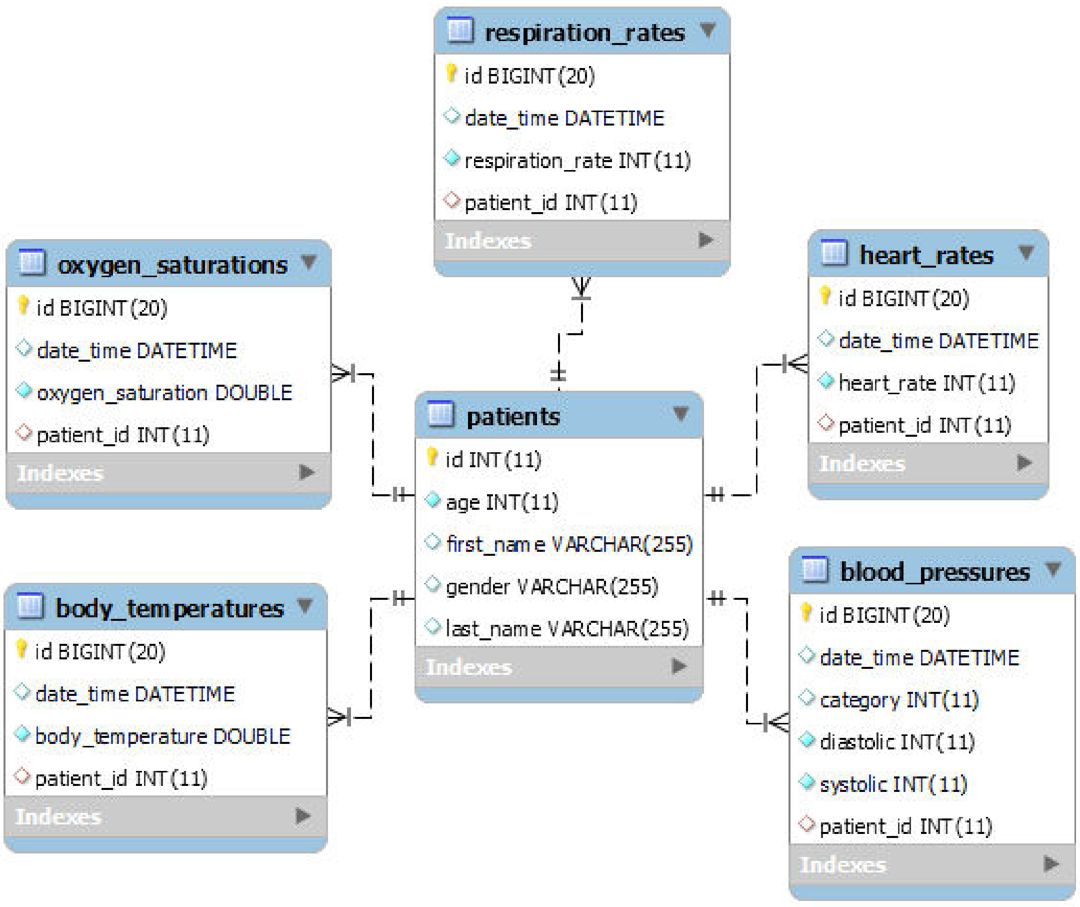


When a patient is added, removed or edited by the user, an API call is made to the backend, which updates the database.

4 Information perspective

A patient has unique id, first name, last name, age, gender and 5 health attributes: heart rate, blood pressure, body temperature, respiration rate and oxygen saturation. Each patient has many heart rate records, many blood pressure records, etc.

All of the 5 health attributes have a unique id, a datetime, and the patient id. Then, each specific attribute has specific fields (e.g. blood pressure has diastolic, systolic and category fields).



5 References and resources

Github repo: https://github.com/zzzzz151/IES_Project

Backlog with user stories: https://grupo-ies.atlassian.net/jira/software/projects/I2/boards/2