Technical Report - **Product specification**

SmartHome Manager

Disciplina:	IES - Introdução à Engenharia de Software
Data:	Aveiro, 2 de janeiro de 2023
Estudantes:	102477: Artur Afonso Ferra Correia 103453: Bruna de Melo Simões 77036: Daniel Luís Fernandes Carvalho 103925: Diogo Alves e Silva
Abstrato do Projeto:	Aplicação web que permite o controlo de uma smart home e todas as suas funcionalidades (tais como regulamento de temperatura, níveis de monóxido de carbono, ligar e desligar luzes em várias divisões da casa, etc) de modo a aumentar a confortabilidade quando o utilizador quer dentro ou fora de casa. A aplicação web deve ser acessível a todas as pessoas que vivem/gerem a casa, e, dependendo da sua posição relativamente à mesma, podem ter mais ou menos privilégios associados (por exemplo, uma empregada doméstica não poderá ter tantos privilégios como os donos da mesma).

Conteúdo

1 Introdução

Neste projeto foi implementada uma aplicação web para a gestão e monitorização de dispositivos e sensores numa casa inteligente. A associação de diferentes dispositivos à aplicação permitirá o rastreio de várias condições presentes nas diferentes divisões da casa, tais como, indicação da temperatura, indicação da humidade ou a deteção da qualidade do ar, bem como a gestão do gasto de recursos energéticos e o controlo de dispositivos ligados à smart home (luzes, aquecedores, medidor de monóxido de carbono, entre outros). O administrador da smart home pode conceder privilégios sobre a sua casa a quem quiser, bem como definir os mesmos e a quem os quer atribuir.

Assim, o desenvolvimento deste projeto permitiu a aplicação de vários conceitos e tecnologias aprendidos ao longo desta disciplina, desde logo a utilização de uma REST API, desenvolvida na framework Spring, para gerir o backend da aplicação e permitir a interação entre a obtenção e processamento dos dados dos sensores, a base de dados persistente e o frontend do sistema. Serão também utilizados e desenvolvidos alguns conceitos de desenvolvimento agile, com a utilização de ferramentas de gestão de backlog, conceitos de desenvolvimento iterativo e continuo, através da utilização de GitHub, entre outros.

Equipa

- **Team Manager:** Daniel Carvalho
- **Product Owner:** Diogo Alves
- **Architect:** Bruna Simões
- **DevOps Manager:** Artur Correia

2 Conceito do produto

Visão

O produto desenvolvido tem como função primária a criação de uma plataforma que permite registar e gerir todos os sensores/dispositivos do tipo IoT presentes numa casa inteligente, deixando que utilizadores autorizados tenham acesso aos dados atuais e históricos de condições ambientais monitorizadas por estes sensores, bem como tenham a possibilidade de realizar certas ações com base na informação apresentada pela plataforma.

Deste modo, serão registadas para cada divisão da casa os valores de condições tais como a temperatura, a humidade ou a qualidade do ar (por exemplo em termos de níveis de monóxido de carbono). A plataforma permitirá também a geração de alertas e notificações quando forem verificados valores perigosos de qualquer um destes valores. Em termos de ações permitidas, a aplicação poderá, por exemplo, permitir aos utilizadores ligarem/desligarem o ar condicionado de acordo com os valores de temperaturas medidos, controlar o estado de luzes presentes nas diferentes divisões, ou até fechar/abrir remotamente janelas na casa (que poderá ocorrer automaticamente, por exemplo, se forem detetados níveis perigosos de monóxido de carbono).

Simultaneamente, a plataforma permitirá também um controlo dos níveis de energia consumido, podendo gerar alertas caso um limite imposto pelo utilizador seja atingido.

Personas e motivações

Nome: Jaime Soares

O Jaime, de 37 anos de idade, trabalha como gestor comercial numa multinacional tecnológica. Ele é casado e tem 3 filhos, vivendo com a sua família numa casa que comprou recentemente. Após esta compra, o Jaime decidiu instalar vários sensores e dispositivos inteligentes, que monitorizam diferentes condições da sua casa, e realizam certas ações em resposta às medições efetuadas.

Devido ao seu trabalho, o Jaime viaja bastante, passando muito tempo fora de casa. Assim, ele gostaria de ter uma plataforma onde poderia verificar remotamente o estado da sua casa, por exemplo a qualidade do ar, bem como os níveis de eletricidade ou gás consumidos, de maneira a garantir que os valores se mantêm dentro do orçamento definido para o mês.

Nome: Eduarda Santos

A Eduarda, de 45 anos, trabalha como empregada doméstica numa moradia inteligente. Durante o seu horário de trabalho, a Eduarda encontra-se sozinha, uma vez que os habitantes da moradia estão também eles no seu local de trabalho, ou nas suas escolas. Deste modo, a Eduarda precisa de acesso à plataforma que gere as funcionalidades inteligentes da moradia.

Cenários Principais

- Após a compra da nova casa, o Jaime verificou que algum do equipamento existente não está nas melhores condições – por exemplo, o sistema central de aquecimento não funciona corretamente, podendo ter algumas fugas. Assim, o Jaime instalou sensores de monóxido de carbono pela casa, querendo utilizador a nossa plataforma para registar os valores deste gás medidos ao longo do dia, sendo notificado caso algum destes valores seja perigoso. Além disso, quer definir que, se a concentração numa dada divisão for demasiado elevada, as janelas desta divisão deverão ser abertas.
- O Jaime pretende manter um olho sobre o consumo de energia, na sua casa, enquanto está fora em viagem. Assim, ele utiliza a nossa plataforma para verificar os dados de consumo deste recurso.
- A Eduarda encontra-se sozinha a limpar a sala da sua moradia. Ela repara que está demasiado quente, e pretende aceder à aplicação para controlar baixar a temperatura através do ar condicionado.
- A Eduarda quer definir os horários de ativação dos regadores da moradia onde trabalha, de acordo com os dados de medição da temperatura externa, registada nos jardins desta.

3 Notas de Arquitetura

Requisitos e Restrições Chave

- Os utilizadores necessitam de uma ligação à internet, devendo aceder ao website do sistema.
- Para se registarem na plataforma, os utilizadores necessitam de introduzir parâmetros tais como o seu email, nome e password. Caso já tenha sido registado, tem de efetuar o seu log-in.
- O utilizador deve introduzir quais as divisões que tem, bem como registar todos os sensores para cada divisão da sua casa, sendo que a qualquer altura pode editar estas informações.
- O sistema deve receber e enviar valores de forma automática, sendo que os mesmos são gerados através de sensores virtuais que simulam as condições de temperatura, qualidade do ar e humidade para cada divisão da casa.
- Após deteção de valores excessivamente elevados/baixos, serão executados determinados protocolos automaticamente de modo a regular esses valores para a normalidade, bem como será envio um alerta para notificar o utilizador, na aplicação.
- O backend funciona como um middleware, que estabelece uma ligação entre os sensores/dispositivos conectados e o frontend.
- O sistema deve armazenar os valores medidos, de modo a providenciar ao utilizador um histórico de registos acessível ao mesmo.
- O sistema dever permitir ao utilizador definir parâmetros personalizados para reagir mediantes os dados medidos no sensor (por exemplo, ligar/desligar AC de acordo com temperaturas definidas, definir horários/temperaturas para os regadores do exterior se ativarem, entre outros)
- Não deve ser permitido acesso de terceiros a dados confidenciais, pelo que a informação acessível pela REST API deve ser protegida, de forma que apenas utilizadores autenticados consigam contactar os REST endpoints desta.
- O frontend deve ser reativo e responder a alterações da informação recebida do sistema.

Vista da Arquitetura

A arquitetura é composta por 4 camadas principais: a geração de dados e o seu envio para uma *message queue*, o *backend* que envolve o processamento lógico dos dados, a lógica de negócio, a REST API e o modelo de dados, o *frontend* e a camada de persistência.

A geração de dados foi feita com recurso a sensores virtuais, escritos em linguagem Python, que simulam sensores do mundo real. Foram criados scripts para simulação de sensores de temperatura, humidade, e medição de monóxido de carbono, sendo estes controlados através de um script que determina quais os sensores necessários para ativar de acordo com o utilizador que está a utilizar a aplicação. A informação gerada será processada e enviada através de um sistema de fila de mensagens para a camada de persistência da base de dados, onde os dados são armazenados, utilizando como *message broker* o RabbitMQ. Neste, deverão ser criados tópicos para a receção de dados de cada sensor desenvolvido. A camada de persistência foi desenvolvida através do modelo relacional, sendo implementada com recurso ao mySQL.

No entanto, se aquando do processamento dos dados gerados se detetarem valores anormais ou perigosos para as condições medidas, deverá ser enviado uma notificação de alerta diretamente

para a camada do *frontend*, através de websockets, tendo sido criado um servidor Python para lidar com a receção destas mensagens no *frontend*.

Por sua vez, *backend* foi construído e implementado na linguagem Java, utilizando a Spring *framework*. A interação com a base de dados é feita através do Spring Data JPA (*object-database mapping framework*). O *backend* engloba também a criação de uma REST API para a interação com o *frontend* (client-side), em Spring Boot, fornecendo os métodos do tipo GET, POST, UPDATE e DELETE necessários o bom funcionamento do serviço e a sua correta integração com a camada de persistência e de geração de dados.

Foram também desenvolvidos mecanismos de autenticação e autorização utilizados para proteger o acesso aos métodos desenvolvidos para a API, através da utilização do Spring Security, em conjunção com tokens JWT. Desta forma, a API desenvolvida possui uma configuração de segurança na qual foram declarados métodos públicos, que não requerem autenticação, e que permitem o login de utilizadores existentes e o registo de perfis de novos utilizadores. Por sua vez, os restantes métodos são privados, necessitando da autorização de uma JWT Bearer token associada a um perfil autenticado para serem acessíveis, com o intuito de proteger os dados confidenciais registados no serviço, e de impedir que terceiros consigam manipular as definições de um utilizador. Esta token é gerada aquando do login de um utilizador, sendo enviada para o *frontend*, que a vai armazenar e enviar em todas as subsequentes chamadas à API.

Por fim, o *frontend* foi implementado utilizando a biblioteca ReactJS, que permite desenhar a aplicação de acordo com o modelo SPA, cuja estrutura deverá ser alterada conforme a atualização de dados (isto é, a plataforma deve ser responsiva). Para comunicação deste serviço com a API no Spring Boot foi utilizada a biblioteca Axios, que permite a realização das chamadas necessárias quando as diferentes páginas são recarregadas. A receção de alertas e notificações diretamente dos sensores foi feita com recurso a WebSockets, conforme anteriormente mencioando.

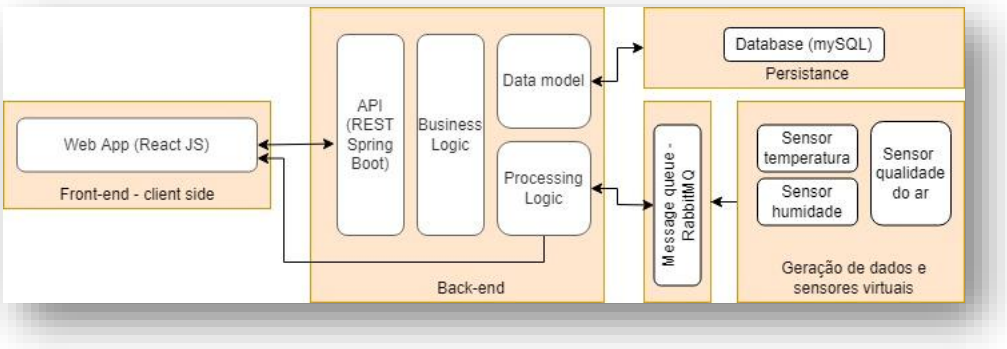

Figura 1 – Diagrama de arquitetura para a aplicação SmartHome Manager.

Interações de Módulos

Os diferentes módulos comunicam constantemente entre si, de modo a suportar ao utilizador toda a informação em tempo real e atualizada.

O primeiro diagrama retrata os acontecimentos sucedidos após um dado utilizador consultar o histórico de temperaturas registadas num dado intervalo de tempo.

A aplicação deve sempre ter dados atualizados da temperatura, medidos constantemente pelo sensor, que envia tais dados para o MessageBroker. De seguida, serão processados os tais dados e enviados para a base de dados que os armazena, com a sua referência temporal.

Assim sendo, quando o utilizador consultar o histórico na aplicação web, esta última contacta a API, que irá pedir a informação ao data access, que obtém os dados da base de dados, registados anteriormente na mesma. Quando o data access recebe os dados, envia os mesmos para serem processados, sendo recebidos pela API, que se encarrega de enviar os dados para a responsive app, que os demostra ao utilizador.

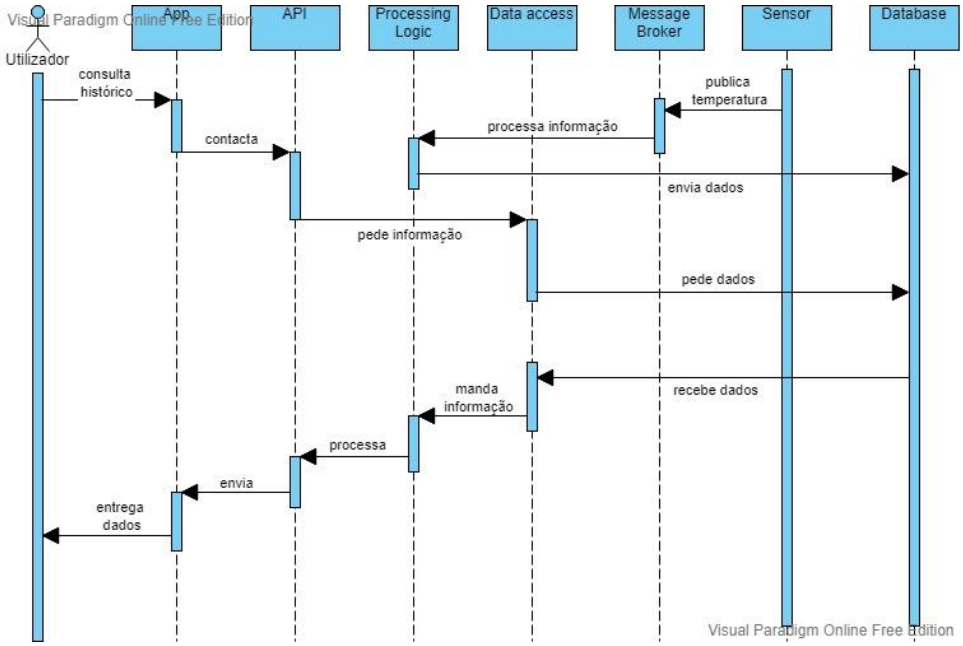

Figura 2 – Diagrama de interação que representa uma ação de um utilizador para acesso a uma funcionalidade do sistema, neste caso, o pedido de consulta do histórico de temperaturas para uma dada divisão.

Já neste segundo diagrama é retratado o evento em que um determinado sensor deteta um nível anormal de um gás tóxico.

Neste caso, o sensor envia, como de costume, os dados que recebe, para o message broker, que processa a informação no processing logic e, enviando os dados para a base de dados para serem gravados (e potencialmente mais tarde, acedidos), verifica a anormalidade dos seus valores, enviando por web socket para a aplicação, que alerta o utilizador da informação recebida.

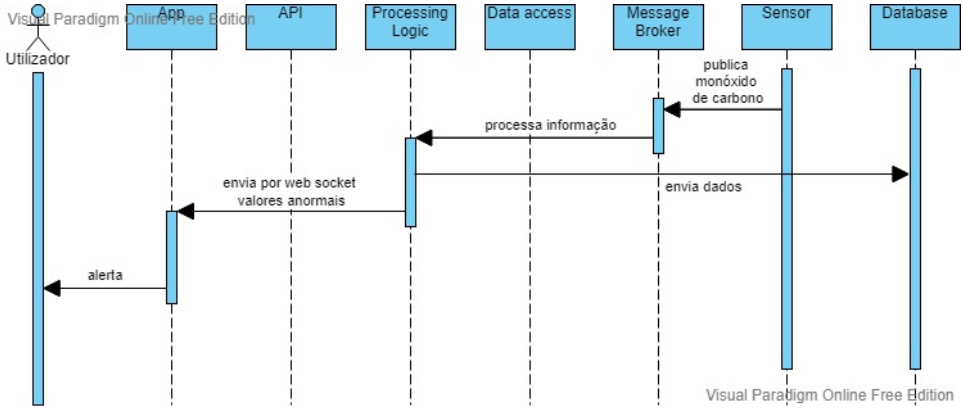

Figura 3 – Diagrama de interação que representa o envio de um alerta ao utilizador, após processamento de um valor anómalo de monóxido de carbono num sensor registado na plataforma.

4 Perspetiva de Informação

Para representar as classes definidas e as suas relações foi desenhado um diagrama do domínio.

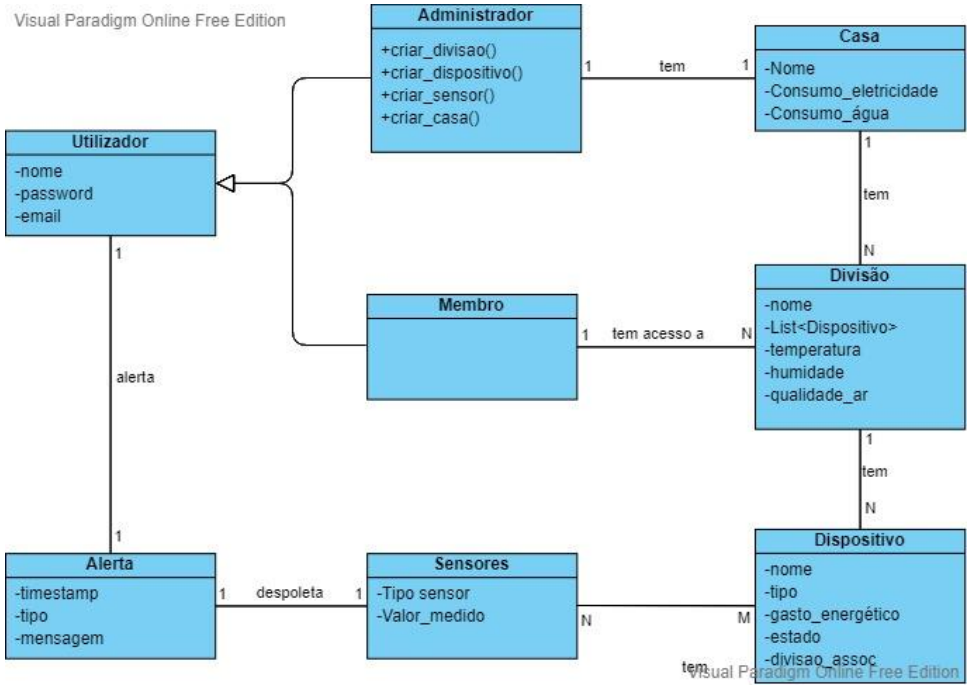

Figura 4 – Diagrama de classes UML que representa as entidades do domínio da aplicação desenvolvida.

Após a análise de requisitos foram definidas as seguintes entidades:

Utilizador

O utilizador é quem interage com a aplicação e efetua a maior parte das ações, como por exemplo associar dispositivos, programar temporizadores, etc. Para além disso este também se subdivide em dois tipos, que serão explicados seguidamente.

Administrador

O administrador é o utilizador com mais permissões. É o responsável por criar uma casa e gerir as suas divisões. Pode adicionar, remover ou editar divisões, e os dispositivos que lhe estão associados. Além de ter acesso a todas as restantes funcionalidades permitidas a qualquer membro.

Membro

Um membro é um utilizador com acesso restrito à aplicação, ou seja, apenas poderá efetuar as ações permitidas pelo administrador. Por exemplo, um utilizador apenas poderá ligar ou desligar dispositivos se tiver o acesso necessário a eles.

Casa

A casa é o elemento principal da aplicação, incluirá várias divisões associadas a ela pelo administrador e também terá estatísticas gerais dessas divisões.

Divisão

Uma divisão é uma entidade que faz parte da casa, possuirá uma lista de dispositivos associados pelo administrador e outros dispositivos especiais, os sensores, que forneceram dados úteis sobre a divisão, temperatura, humidade, etc.

Dispositivo

Um dispositivo pode ser várias coisas, um eletrodoméstico ou uma janela, que será associado a uma divisão pelo administrador. Esse dispositivo terá um gasto energético e um estado (ligado, desligado), estado esse que poderá ser alterador por um utilizador.

Sensor

Dispositivo responsável por detetar e enviar periodicamente os valores de humidade/temperatura/qualidade do ar. No caso, iremos simular os sensores através de geradores, que criam periodicamente valores pseudoaleatórios realistas.

Alerta

O alerta é uma entidade especial no sentido que será gerado na parte de processamento caso exista uma avaliação anómala de um determinado dado, temperatura elevada, níveis de CO elevados, etc. Esse alerta poderá depender dependendo da sua magnitude, por isso tendo um tipo e serve para informar ao utilizador de alguma informação pertinente através do envio de uma mensagem.

5 Práticas de Desenvolvimento

Gestão de Backlog

A gestão do desenvolvimento do trabalho e da distribuição de tarefas foi feita de acordo com a metodologia Scrum, baseada em Agile. Assim, para cada iteração de desenvolvimento foram definidos os objetivos principais e as tarefas a cumprir para os atingir, sendo a distribuição de tarefas realizada com recurso ao software Jira.

Workflow

Recorreu-se à utilização de um repositório no GitHub para controlar as versões do trabalho e para permitir o desenvolvimento colaborativo do projeto.

Deployment

Para permitir o deployment, recorremos à utilização do Docker para encapsular o código e as bibliotecas necessárias para correr os diferentes serviços, tendo sido aplicado o princípio de segregação de responsabilidades, através da divisão dos principais componentes do trabalho em diferentes containers. A reunião dos diferentes componentes para execução da aplicação é feita com recurso ao Docker Compose, que permitiu a definição de diferentes networks para comunicação entre os vários serviços. Os containers utilizados foram os seguintes:

- Um container para correr a base de dados MySql;
- Um container para correr o *message broker* RabbitMQ;
- Um container para os serviços relacionados com o FrontEnd;
- Um containar para os serviços relacionados com a API no Spring Boot;
- Um container para a geração de dados;
- Um container para o backend que permite a comunicação direta do serviço de geração de dados com o serviço do frontend, aquando do envio de notificações e alertas;

No final, o deployment foi concretizada na VM fornecida pelo docente, de acordo com as indicações fornecidas por este.

6 Conclusão

Consideramos que, no final do desenvolvimento do Projeto, foram atingidos os objetivos principais definidos no inicio do trabalho, nomeadamente, o desenvolvimento de uma aplicação full-stack que permite a interação de um utilizador com os diferentes dispositivos presentes na sua casa inteligente, e o armazenamento e gestão do processamento da informação captada por sensores, que o vão ajudar na gestão dos seus dispositivos.

Foi o primeiro projeto desta dimensão e escala que realizamos no curso, tendo sido o nosso primeiro contacto com tecnologias como o Spring Boot, o Docker e até o RabbitMQ, tendo sido também a primeira vez que aplicamos metodologias Agile para a gestão e organização de um trabalho. Deste modo, sentimos que o desenvolvimento deste projeto foi muito positivo para o nosso desenvolvimento como engenheiros de software, tendo alterado a nossa visão de como é feita a gestão de projetos de maior dimensão.

Em termos de trabalho a realizar em potenciais iterações futuras, consideramos como ponto essencial a restruturação do modelo de base de dados desenvolvido. Neste trabalho optamos por um modelo relacional, que nos é mais familiar, e que serviu para o desenvolvimento pretendido como *proof of concept* do sistema. No entanto, para tornar este escalável, o esquema da base de dados teria que ser alterado, e potencialmente consideraríamos aplicar um modelo NoSQL para o armazenamento de alguma informação que necessita de uma maior flexibilidade esquemática.

Gostaríamos também de melhorar algumas funcionalidades, como por exemplo o controlo de privilégios a outros utilizadores por parte dos administradores das contas, ou funcionalidades extra para o controlo de equipamentos e para a gestão de recursos. Por fim, outro ponto importante de futuras iterações passaria por melhorar a usabilidade do Frontend, e aplicar mais mecanismos de segurança tanto neste serviço, como no backend.

7 Referências e Recursos

ReactJS
https://reactjs.org/docs/getting-started.html

RabbitMQ
https://www.rabbitmq.com/documentation.html

Spring Boot
https://spring.io/projects/spring-boot

MySQL
https://dev.mysql.com/doc/