Relatório Técnico - Especificações de Projeto

Campus Monitoring

Disciplina:	IES - Introdução à Engenharia de Software
Data:	Aveiro, 19 de Dezembro de 2019
Alunos:	88969: Alexandre Lopes 89198: André Amarante 89323: Edgar Morais 51531: Gabriela Santos
Resumo do Projeto:	Este projeto consiste num sistema de monitorização de espaços universitários, através da recolha de informação em tempo real de parâmetros como a temperatura, humidade, pressão atmosférica e níveis de CO_2, providenciando uma forma de assegurar um maior controlo por gestores e trabalhadores com vista a tornar o espaço adequado às suas funções, bem como uma forma de fornecer informações acerca de eventuais catástrofes com o objetivo de desencadear respostas rápidas e eficazes às mesmas.

Índice:

1 Introdução

Este projeto prático tem como foco a simulação do ambiente de desenvolvimento de uma aplicação web em contexto empresarial. A integração com a disciplina de IES é assente na exposição aos alunos de princípios de gestão de projetos, paradigmas de arquitetura e ferramentas tecnológicas largamente usadas na indústria de software. Após esta etapa de exposição inicial, os alunos irão adquirir competências fundamentais para o dia-a-dia do trabalho de um engenheiro de software, tanto num âmbito organizacional, como de análise de soluções.

Relativamente ao âmbito organizacional, a equipa propõe-se a implementar uma seleção de princípios Agile para gestão do projeto. O desenvolvimento do mesmo será organizado em iterações de 2 semanas, cujo progresso será avaliado mediante a capacidade da equipa concluir as tarefas selecionadas do backlog, segundo o paradigma Scrum. A implementação de novas funcionalidades ocorrerá de modo incremental. Iremos utilizar também algumas práticas comuns de Extreme Programming, como a concepção de funcionalidades da aplicação na perspetiva de user stories geridas por um Product Owner. A decomposição das funcionalidades em tarefas mais simples e a hierarquização da prioridade das mesmas serão essenciais para atingir um bom fluxo de trabalho entre equipa.

Adicionalmente, pretendemos utilizar uma arquitetura *multilayered* como referência e analisar qual a melhor forma de obter uma interação ótima entre componentes da nossa aplicação. Nesse sentido, temos também como objetivo fazer escolhas informadas de quais as ferramentas tecnológicas (*frameworks*, *template engines*, bases de dados e *message brokers*) que nos permitirão tirar partido das suas características para a melhor implementação da nossa visão inicial para a aplicação.

Finalmente, para aumentar a produtividade da equipa e podermos concretizar entregas constantes das novas funcionalidades introduzidas no projeto, propomo-nos também a adotar mecanismos de DevOps ao longo do mesmo, bem como ferramentas de código partilhado e de gestão de tarefas.

2 Conceptualização do Produto

Visão Global

O produto Campus Monitoring encontraria o seu espaço no mercado como uma ferramenta essencial à execução do conceito de Edifícios Inteligentes. Este conceito mistura um domínio que tem sido tradicionalmente estático, como o edifício clássico, e confere-lhe características responsivas e dinâmicas. Com os avanços tecnológicos no campo de automação, atualmente é possível controlar à distância componentes de um edifício (como por exemplo, sistemas de ar condicionado, abertura de portas e janelas para ventilação, movimentar persianas), assim como utilizar sensores para medir em tempo real

o desempenho térmico e da qualidade do ar do mesmo.

A nossa solução centra-se na resolução do problema fulcral de como fazer a medição contínua e processamento das mudanças constantes de temperatura e níveis de CO_2 de espaços públicos em tempo real. A legislação atual de eficiência energética prevê um intervalo restrito de valores aceitáveis para estes parâmetros num edifício público, contudo, a fiscalização tem-se revelado demasiado difícil de concretizar, o que poderia ser solucionado pela nossa aplicação.

Como tal, iremos introduzir mecanismos de reação a eventos, sob a forma de alertas, que podem ser vistos pelos utilizadores da nossa aplicação como notificações no seu browser ou e-mail, o que garantiria uma monitorização 24/7.

Adicionalmente, pretendemos fornecer aos utilizadores do espaço um canal de comunicação para expressar feedback relativo às suas necessidades de conforto, visto que o desempenho dos mesmos no seu local de trabalho poderá ser significativamente afetado por condições adversas.

Uma aplicação com estas características poderia vir a ser integrada com facilidade num projeto tecnológico com maior escopo, que possibilitasse controlar equipamentos do edifício à distância para corrigir automaticamente desvios das condições ideais climatéricas da cantina após serem alertados; assim como estender o âmbito da aplicação para cobrir os restantes espaços públicos do campus.

Personas

Pedro Bastos: O Pedro tem 40 anos e é administrador dos serviços da cantina da Universidade de Aveiro há 6 meses. Devido à natureza do seu trabalho de escritório e horário exigente, não tem muita disponibilidade para falar diretamente com os seus funcionários ou dirigir-se múltiplas vezes durante o dia à cantina. Como tal, quer efetuar mudanças na gestão das condições do local de trabalho e ter um canal de comunicação direto para receber feedback. O seu objetivo principal é aumentar o desempenho dos seus trabalhadores, pelo que pretende a atividade profissional destes não seja perturbada por condições adversas e que estes se encontrem à vontade para expressar pontos negativos. Adicionalmente, a cantina poderá ser alvo de inspeções de fiscalização de eficiência energética, pelo que é do interesse do Pedro investigar potenciais problemas antecipadamente.

Motivação: O Pedro gostaria de aceder à visualização da evolução das condições de temperatura e de qualidade do ar ao longo do tempo e do histórico de alertas de violação de limites legislados para análise estatística dos dados. Para além disso, necessita um mecanismo de registo e quantificação dos principais problemas detectados no espaço de trabalho.

Maria Cardoso: A Maria tem 60 anos e trabalha como cozinheira na cantina da Universidade de Aveiro há cerca de 20 anos. Recentemente, foi diagnosticada com uma doença reumatológica cujos sintomas pioram com exposição ao frio. Nesse sentido, o seu

médico de família aconselhou-a a evitar temperaturas mais baixas. O chefe de equipa da cantina pretende acomodar a sua limitação, dado a vasta experiência da Maria como cozinheira e o seu contributo essencial para o bom funcionamento da cantina. Apesar da promessa de acomodação, muitas vezes a Maria passa horas no seu local de trabalho sob temperaturas demasiado frias, principalmente no Inverno, o que interfere com a sua capacidade de movimentação. Por vezes, ela comunica verbalmente o problema ao seu chefe, mas vê as suas preocupações serem ignoradas no meio do volume de trabalho.

Motivação: A Maria gostaria que as suas queixas fossem oficialmente registadas e de receber notificações quando as condições de temperatura do seu local de trabalho se tornarem prejudiciais à sua saúde.

Cenários Principais

Pedro prepara-se para uma auditoria à cantina – O Pedro abre a aplicação e entra no menu de autenticação. Coloca os seus dados de funcionário da UA, o seu login é processado com sucesso. De seguida, é redirecionado automaticamente para a página referente a uma das cantinas que gere. Nesta página, ele pode *navegar para todos os espaços que estão sob a sua responsabilidade* no campus. Na página da cantina, explora a opção de aceder a *registos dos históricos de evolução das condições máximas e mínimas* relativas a temperatura, humidade e níveis de CO_2. Ele decide aceder à opção de *consulta de relatório de desempenho do espaço*. Estes dados proporcionam-lhe uma visão global da performance do espaço ao longo do tempo e da geração de alarmes referentes a violações de condições máximas e mínimas de conforto. Ele irá usar estes dados na auditoria para mostrar o efeito positivo das mudanças que introduziu como novo administrador e que este espaço está a cumprir os regulamentos.

Pedro analisa feedback dos seus funcionários na reunião semanal – Na página principal da cantina, o Pedro seleciona a *opção de exibir críticas*. Ele utiliza estes dados na sua reunião semanal com a equipa de gestão da cantina para aferir o grau de contentamento dos seus funcionários e estratégias de melhoria das condições para o mesmos.

Maria prepara-se para sair de casa para o trabalho – Ao iniciar o seu dia, a Maria tem agora por hábito abrir a aplicação, autenticar-se, dirigir-se à página da cantina e selecionar a opção para *visualizar as condições atuais*. Assim, poderá escolher o seu vestuário de forma a estar mais preparada para se sentir confortável no seu local de trabalho.

Maria realiza as suas tarefas diárias no local de trabalho – Durante o dia, a Maria é *notificada por e-mail* que a temperatura da cantina excede o mínimo que é seguro para alguém com os seus problemas de saúde. Como tal, ela interrompe as suas tarefas e comunica verbalmente ao chefe de equipa que há um problema para resolver. Após a introdução da aplicação, ela notou melhorias no seu espaço de trabalho e tem agora mais facilidade em desempenhar as suas tarefas, com a sua condição de saúde a ser acomodada. Ela acede à página da cantina, *seleciona a opção de deixar uma crítica* e deixa uma crítica positiva e uma pontuação mais elevada ao seu espaço de trabalho.

3 Arquitetura

Requisitos chave e limitações

- O sistema deve fornecer um mecanismos de autenticação de funcionários e administradores através da inserção do email da universidade.
- O sistema deve fornecer uma forma de persistência, segura, estável e funcional, de modo a preservar os dados relativos a funcionários e administradores, bem como dos registos dos sensores.
- O sistema deve ter integração com o DBMS, de forma a tornar possível a gestão, organização e acesso a dados armazenados na base de dados.
- O sistema deve apresentar um desempenho fluído e otimizado.
- O sistema deve fornecer uma interface adequada, que tenha um bom equilíbrio entre a apresentação de toda a informação relevante e a simplicidade de utilização.
- O sistema deve ter implementado um mecanismo de notificações que permita aos utilizadores configurarem eventos para serem notificados imediatamente aquando da sua ocorrência.
- O sistema deve emitir alertas sempre que haja ocorrências de eventos pré-definidos, principalmente quando os valores de temperatura, humidade e CO_2 excedem ou estão abaixo dos valores limite.
- O sistema deve impedir que um utilizador tenha acesso a dados confidenciais e privados de outros utilizadores.
- O sistema deve disponibilizar em tempo real as condições do espaço em que está instalado, medido através de sensores físicos.
- O sistema deve ter uma ligação contínua à Internet.
- O sistema deve ser resistir e continuar em funcionamento em condições anormais de temperatura e níveis de humidade, pressão atmosférica e CO_2.
- O sistema deve ser robusto e ser instalado de forma estratégica, de forma a interferir o menos possível com o fluxo de trabalho do corpo trabalhador existente, bem como resistir a eventuais ocorrências

Visão da Arquitetura

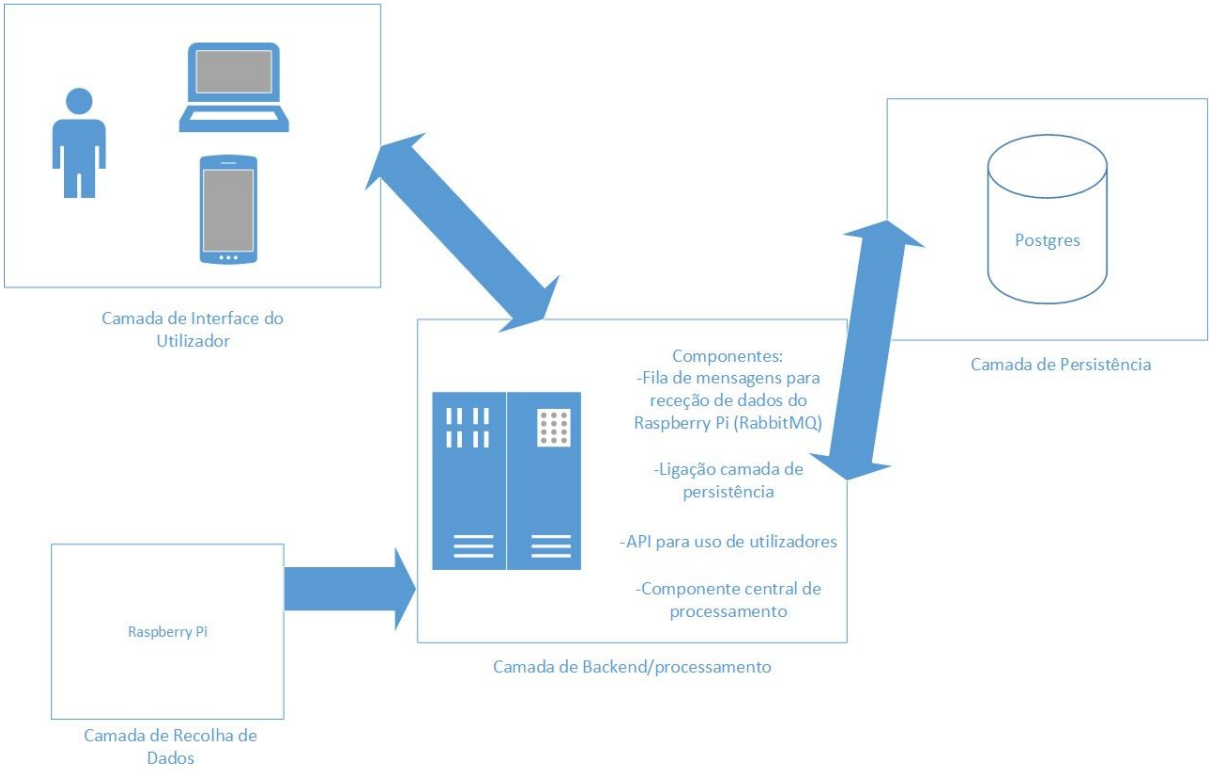
A arquitetura da solução, tal como proposto, será multilayered, podendo ser identificadas 4 camadas/módulos gerais: camada de persistência, camada de backend, camada de user interface and interaction e camada de geração de dados sensoriais.

A camada de dados sensoriais é uma camada de hardware constituída pelo Raspberry pi 3b+ com sensor BME680. A camada de persistência contém a base de dados de armazenamento dos dados recolhidos pelo Raspberry Pi. A decisão de uso de base de dados relacional foi feita pela interdependência dos dados recolhidos e necessidade das vantagens proporcionadas por este tipo de base de dados. Foi escolhido PostgreSQL em vez de MySQL pelas seguintes razões: Postgres tem Pgadmin como gui intuitiva efácil de usar; mais compatível com SQL do que MySQL, logo mais facilmente usada por quem tem background em SQL server; bastante usada em sistemas em larga escala onde operações de escrita e leitura rápidas são cruciais e os dados têm de ser validados; múltiplas escritas sem *lock* possíveis; grande comunidade para suporte e fóruns de ajuda; suporte a

extensões (como Timescale) que serão necessárias no projeto.

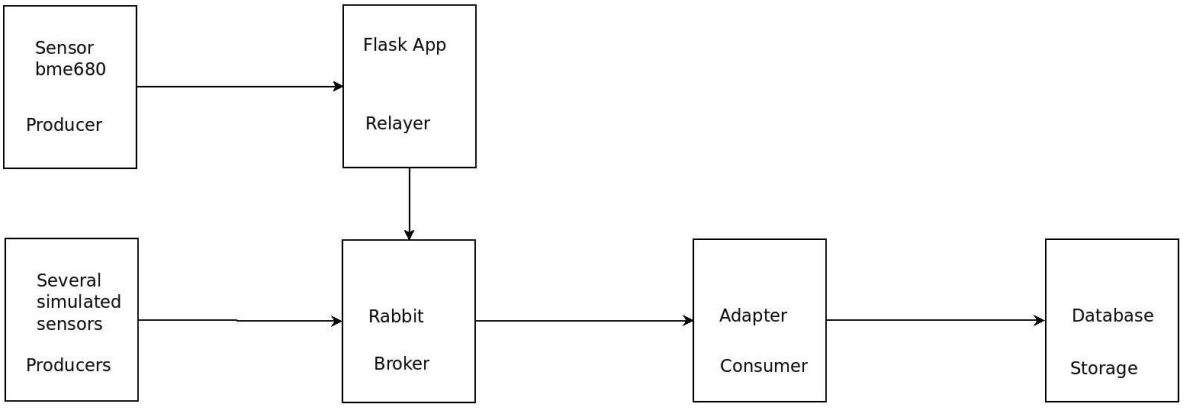
A camada de *backend* trata do trabalho pesado de processamento dos dados, para gestão de dados de interesse (valores elevados de CO_2, valor mais baixo de temperatura num certo dia/mês, etc) e passagem dos dados à camada de *user interface*, que é responsável pela interação com o utilizador. Nesta última camada, a interação com o utilizador é feita através de *web app*. Esta escolha prende-se principalmente com o facto de um dos utilizadores principais do sistema ser um grupo de pessoas (trabalhadores da cantina) que não terão disponibilidade/capacidade para interagir com o telemóvel para visualizar os dados/alarmes.

Primeira versão do diagrama de arquitetura:

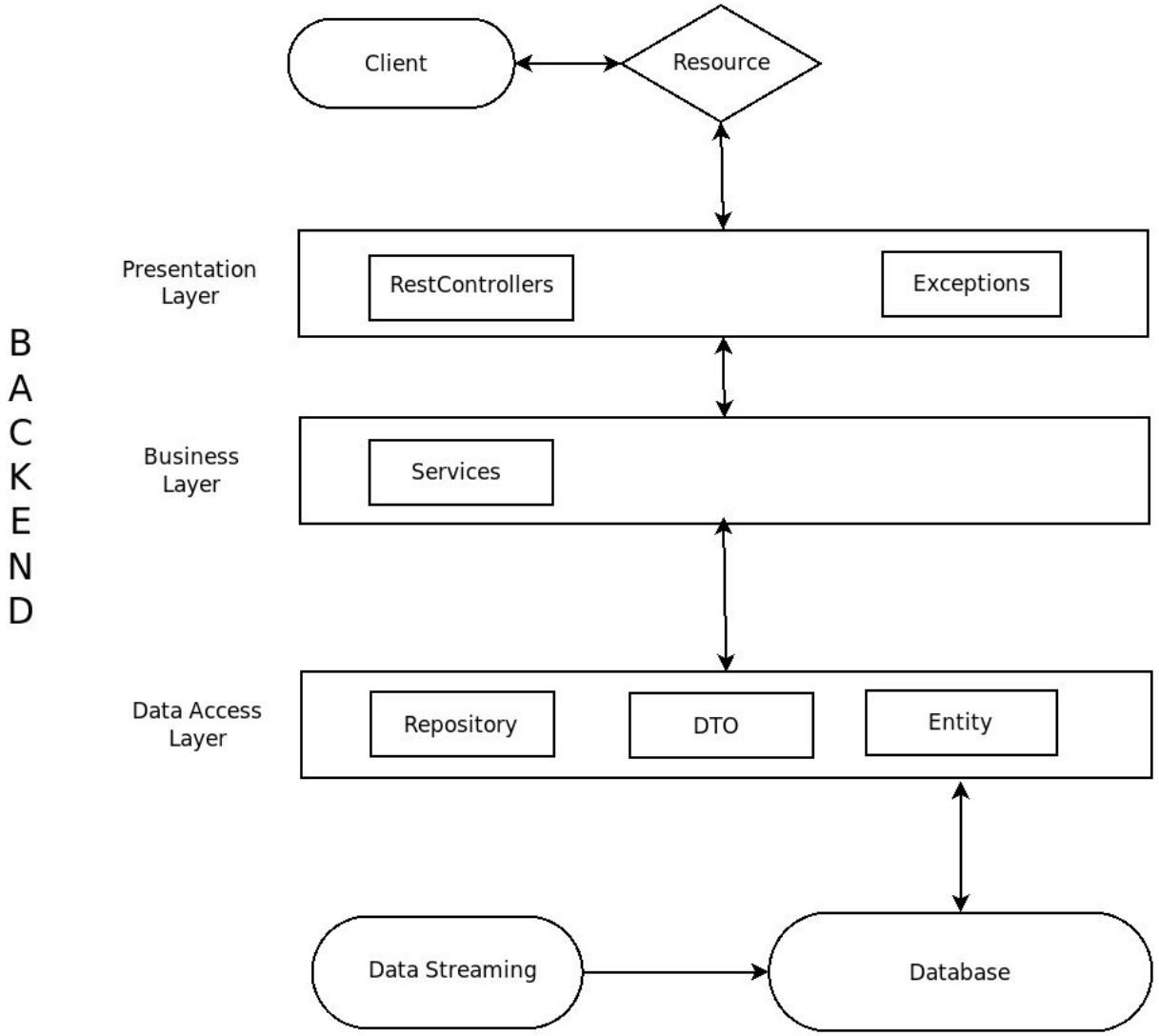


Interações entre módulos

A comunicação entre os módulos tem a camada de *backend* como elemento central. Os dados recolhidos do Raspberry Pi (na camada de recolha de dados) são inseridos na camada de persistência (base de dados) através de um adapter. Este adapter recebe dados dos sensores transferidos através do RabbitMQ. Neste trabalho foi utilizado um sensor real e três geradores aleatórios de valores para simular dados realistas. Os scripts geradores comunicam com o adapter diretamente através do RabbitMQ. No entanto o sensor utiliza um relayer construído utilizando flask que por sua vez os dados para o RabbitMQ. Estes aspetos da camada de recolha de dados pode ser observada no esquema abaixo.

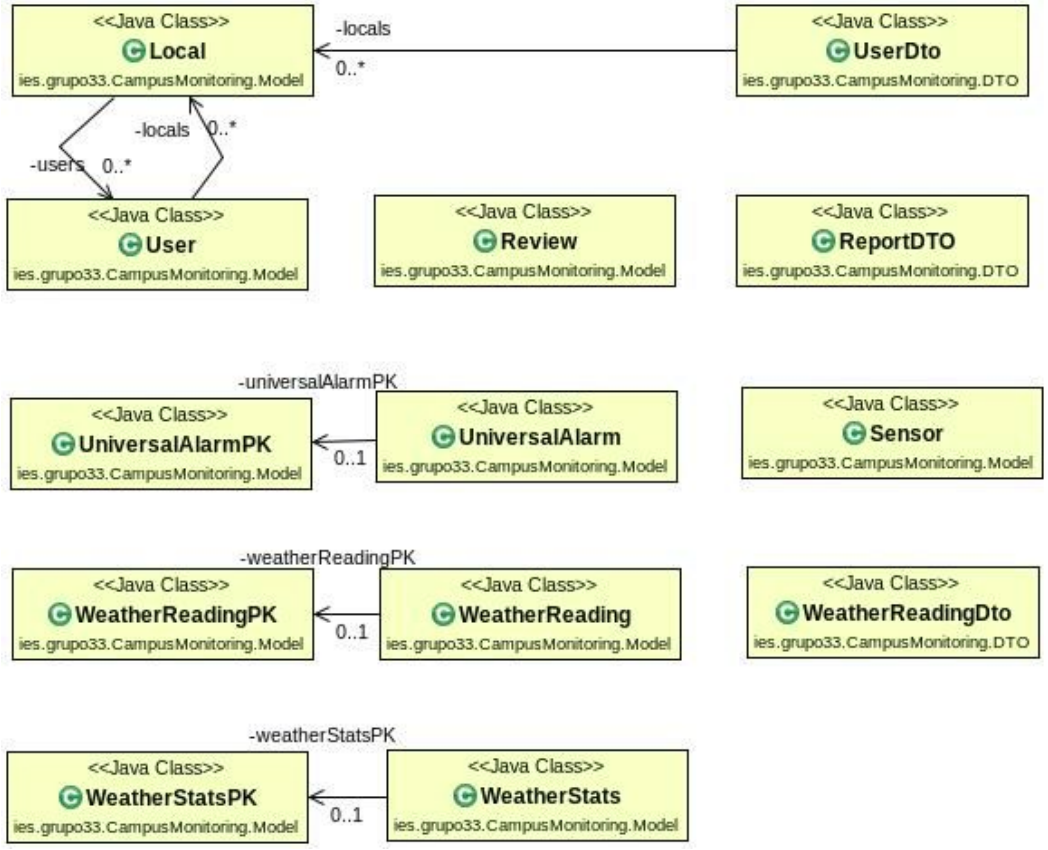


A camada de *backend* foi desenvolvida utilizando a *framework* **Spring Boot**. Esta *framework* é que gere a comunicação com a camada de persistência. Esta comunicação é executada através da interface Java Persistence Api (JPA) para mapeamento de objetos relacionais para classes e do Java Database Connectivity para o envio de instruções SQL. Nesta camada de persistência, nós utilizamos uma base de dados Postgresql. A comunicação entre o *frontend* e o *backend* é realizado através de pedidos http. Isto porque o *backend* de esta aplicação tomou a forma de uma REST API. Também é no *backend* que ocorre o envio de alertas para os emails dos utilizadores.


Client
Resource
Presentation Layer
RestControllers
Exceptions
BACKEND
Business Layer
Services
Data Access Layer
Repository
DTO
Entity
Data Streaming
Database

4 Information perspective

Diagrama do UML das classes do model



Estas são as classes que foram utilizadas no model do nosso projeto. Estas estão associadas diretamente a entidades do nosso modelo de base de dados que vais ser analisado de seguida. Como se pode observar para algumas entidades foi necessário criar duas classes, uma para a chave primária e outra para a junção da chave primária com os restantes atributos da entidade. Isto foi feito em todas as entidades que possuem chaves primárias com compostas. Também é de notar que foram criadas algumas classes que nó chamamos de DTO (Data Transfer Object) que servem para enviar as informações necessárias ao pedido realizado à API quando é necessário fazer algum processamento intermédio aos dados no backend e não é possível enviar uma classe do model na íntegra.

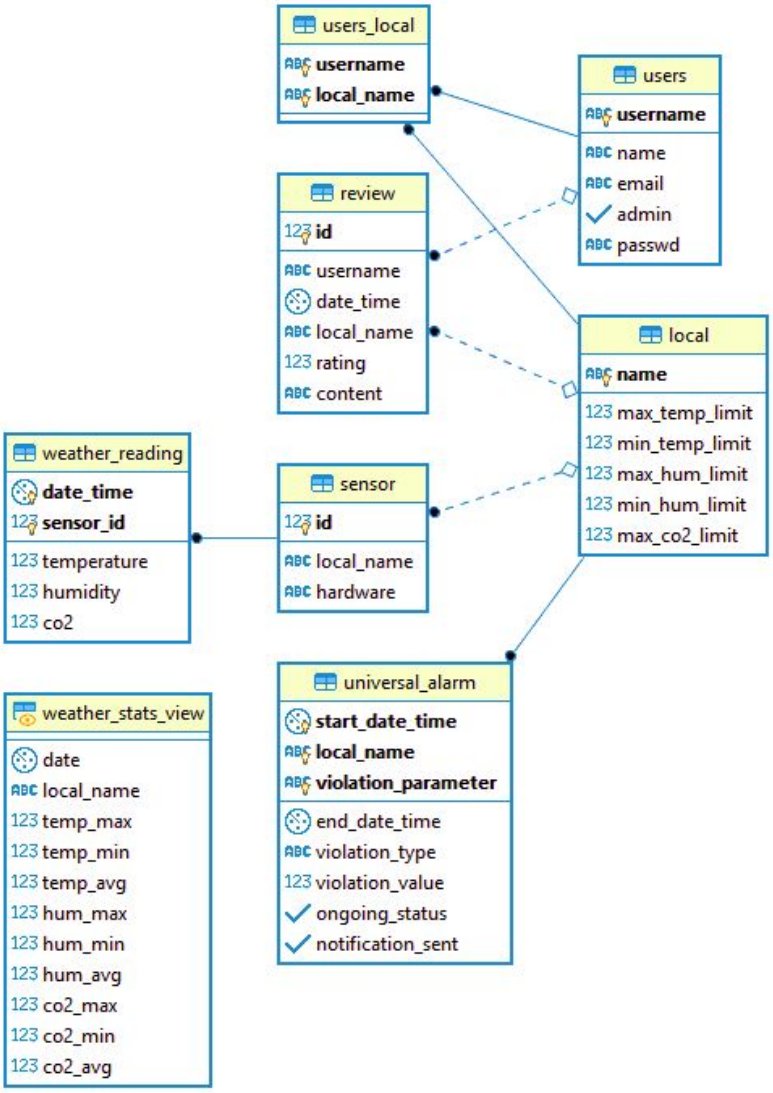


Diagrama Entidade-Relação da base de dados utilizada na camada de persistência.

Agora, iremos realizar uma breve análise de alguns aspetos do diagrama. Os sensores possuem várias leituras e cada sensor está associado a um local. No entanto este local pode ser alterado se o sensor mudar de sítio. Sendo que para obter o local de uma dada leitura é necessário realizar uma junção (JOIN) da tabela weather_reading com o sensor. Cada local tem as suas condições limite de acordo com os seus limites estipulados. De acordo com estes limites podem ser realizados vários alarmes caracterizados pelo local a que se referem, pelo tempo em que foram ativados e pelo parâmetro violado. Além disto, cada utilizador está associado a vários locais e vice-versa, sendo assim, foi necessário criar uma tabela que associa utilizadores aos locais a que pertence.

5 Análise de Progressão de Sprints

Sprint 1

Período: 31 de Outubro de 2019 a 13 de Novembro de 2019

Tarefas concluídas com sucesso:

1. Documentação da conceptualização do produto;
2. Criação do sistema de backlog do produto no Pivotal Tracker, com todas as user stories redigidas para representar as funcionalidades e dispostas hierarquicamente por ordem de prioridade;
3. Repositório oficial do projeto criado no GitHub;
4. Configuração do sensor para obtenção de dados;
5. Definição da arquitetura do produto:
    a. Documentação dos requisitos chave e limitações;
    b. Documentação da visão da arquitetura;
    c. Documentação das interações entre módulos.

Tarefas não terminadas:

1. Protótipos para a consulta das condições climatéricas em tempo real e histórico das condições climatéricas.

Sprint 2

Período: 14 de Novembro de 2019 a 27 de Novembro de 2019

Tarefas concluídas com sucesso:

1. Pipeline de data streaming completo. Contudo, ainda se encontram apenas disponíveis os dados coletados por sensores simulados;
2. Orquestração da Infraestrutura (PostgreSQL, RabbitMQ, adapters, collectors e aplicação Spring Boot);
3. Deployment de serviços por via de containers Docker na VM de toda a infraestrutura;
4. Protótipos para a consulta das condições climatéricas em tempo real e histórico das condições climatéricas;
5. Funcionalidade Weather Reading API

    Git Branch: Weather_Reading_Feature
6. Funcionalidade Weather Stats API

    Git Branch: weather_stats_business_db

Tarefas não terminadas:

1. O frontend da aplicação web ainda não foi integrado com a componente backend.
   Git Branch: frontend

Sprint 3

Período: 28 de Novembro de 2019 a 11 de Dezembro de 2019

Tarefas concluídas com sucesso:

1. Integração da geração dados reais de condições climatéricas na infraestrutura de data streaming, a partir de um Raspberry Pi B3 com sensor BME680;
2. Funcionalidade Universal Alarms API com notificação por e-mail;
   Git Branch: Universal_Alarm_Feature
3. Funcionalidade Report API, para geração de relatórios de desempenho dos espaços;
   Git Branch: Report_Feature
4. Funcionalidade Review API
   Git Branch: Review_Feature

Tarefas não terminadas:

5. O frontend da aplicação web ainda não foi integrado com a componente backend;
   Git Branch: frontend
6. Funcionalidade de Login de Utilizadores.
   Git Branch: Login_Feature

Sprint 4

Período: 12 de Dezembro de 2019 a 18 de Dezembro de 2019

Tarefas concluídas com sucesso:

1. Integração do frontend da aplicação web;
   Git Branch: frontend
2. Funcionalidade de Login de Utilizadores.
   Git Branch: Login_Feature

Síntese de Implementação de Funcionalidades

User Story	Estado	Git Branches	Observações
O trabalhador consulta as condições climatéricas com atualização em tempo real	Concluída	Weather_Reading_Feature frontend	-
O administrador consulta histórico das condições climatéricas	Concluída	weather_stats_business_db frontend	-
O administrador é notificado por e-mail e na página principal de um local de alarmes a avisar que as condições limite foram violadas	Concluída	Universal_Alarm_Feature frontend	-
O trabalhador consulta o seu perfil na aplicação	Concluída	Login_Feature frontend	Abrange frontend da web app e REST API
O administrador consulta estatísticas de desempenho de um espaço	Concluída	Report_Feature frontend	Mudança para consulta de estatísticas ao invés de consulta de histórico de alarmes que tinha sido inicialmente proposta
O trabalhador deixa uma crítica ao seu espaço de trabalho	Concluída	Review_Feature frontend	-
Os trabalhadores criam alertas personalizados consoante as suas preferências	Excluída	-	Excluída por motivos de repetição do mecanismo dos alarmes universais e encurtamento para uma semana do Sprint 4

6 Conclusão

No início do projeto, nomeadamente, na primeira iteração, houve um grande foco por parte de toda a equipa no planeamento das ações a tomar e também no desenvolvimento e integração imediata do backend.

Em relação à metodologia Agile, foram implementados princípios com sucesso, tendo também havido outros sem sucesso. Um dos pontos não implementados desta metodologia prende-se com o planeamento, isto é, esta fase inicial do projeto acabou por ser mais plan-driven do que o esperado, em parte, por nem todos os membros do grupo estarem familiarizados uns com os outros, em termos de práticas e pontos fortes de cada um. A iteratividade deste modelo foi aplicada com parcial sucesso, pois nem todos os objetivos de cada iteração foram alcançados logo no final dessa iteração.

Sem que pudesse ser planeado ou antecipado, o desenvolvimento do backend rapidamente se antecipou aos outros componentes, nomeadamente ao frontend, pelo que acabou por ocorrer uma integração atrasada destes componentes que foram sendo desenvolvidos e testados fora do ambiente de deployment.

Também procurámos adotar o XP release cycle, com a seleção de user stories por iteração, divisão em tarefas, desenvolvimento, release e avaliação. Mais uma vez, neste ponto o sucesso foi parcial, em parte devido ao atraso na integração do frontend.

O grupo tirou também partido das funcionalidades do GitHub, nomeadamente, com a adoção de um workflow Git em que cada user story era uma branch específica e após o desenvolvimento estar terminado, o product owner realizou code review para determinar se preenchia os requisitos da funcionalidade. Se sim, procedia-se à merge e disponibilização na master para deployment futuro na VM. Caso contrário, o developer era informado da necessidade de alterações.

Ao nível do frontend, foi adotada com sucesso a prática de Pair Programming, o que levou a uma perceção mais rápida das implementações necessárias, possíveis estratégias de implementação e decisão acertada no momento final. Apesar de não ser um dos pontos mais fundamentais do projeto, é uma área de grande interesse para alguns membros do grupo, pelo que acabou por se tornar numa espécie de side-project, por a sua integração não ter sido feita até perto da entrega do MVP.

Em termos de DevOps, existiu uma facilidade no facto da equipa de Developers ser também a de Operators. Foi adotado um esquema interessante, com um membro do grupo como “cabeça” da equipa de deployment e os restantes a representar o seu papel de Developers na adaptação às necessidades dos Operators.

7 Referências e recursos

Componentes para sistema montado com Raspberry Pi
breakout garden:

- https://github.com/pimoroni/breakout-garden

bme680 sensor:

- https://github.com/pimoroni/bme680-python

RabbitMQ
Tutorial:

- https://www.rabbitmq.com/tutorials/tutorial-one-python.html

Integração com Docker:

- https://hub.docker.com/r/bitnami/rabbitmq/

Documentação que auxiliou o uso da ferramenta Pika:

- https://pika.readthedocs.io/en/stable/examples.html

Docker para criar containers da aplicação e restante infraestrutura (scripts e db)

- https://www.callicoder.com/spring-boot-docker-example/
- https://www.callicoder.com/spring-boot-mysql-react-docker-compose-example/

Spring Boot

- https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation
- https://www.callicoder.com/hibernate-spring-boot-jpa-composite-primary-key-example/
- https://dzone.com/articles/pagination-in-springboot-applications
- https://www.baeldung.com/javax-validation

Frontend

- https://api.jquery.com/jquery.ajax/
- https://medium.com/better-programming/how-to-use-local-storage-with-javascript-9598834c8b72
- http://zetcode.com/articles/javascriptjsonurl/