Intrusion Tracker

Sistema de gerenciamento de acessos de uma empresa

Universidade de Aveiro

Licenciatura em Engenharia Informática

UC 40384 - Introdução à Engenharia de Software

Aveiro, 18 de dezembro de 2019

Projeto desenvolvido por:

89348 - Diogo Gonçalves Silva

88808 - João Miguel Nunes de Medeiros e Vasconcelos

88886 - Tiago Carvalho Mendes

88931 - Vasco António Lopes Ramos

universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Índice

1. Introdução

O presente documento tem como principal objetivo descrever detalhadamente o projeto desenvolvido no âmbito da unidade curricular de Introdução à Engenharia de Software da Universidade de Aveiro.

O principal objetivo pretendido com a realização deste projeto seria o de os estudantes aplicarem diversas práticas de engenharia de software, lecionadas durante o semestre, tais como:

- Desenvolver a especificação de um produto através de uma análise de requisitos e casos de uso.
- Projetar e implementar uma arquitetura de software, baseada em diversas tecnologias e *frameworks*.
- Aplicar práticas de trabalho colaborativo e processos de desenvolvimento de software numa pequena equipa.

De seguida iremos explicar todas as decisões tomadas em cada etapa do projeto.

2. Acessos

Para aceder à nossa plataforma web basta aceder ao url http://192.168.160.220:3000/ e é possivel transferir a nossa aplicação através do link, Intrusion Tracker - APP.

Além disso, é possível aceder à nossa REST API, através do url http://192.168.160.220:8080/api/v1/.

Para iniciar sessão na nossa plataforma web e mobile temos os seguintes acessos:

Username	Password	Role
j.vasconcelos99@ua.pt	pwd	ADMIN
diogo04@ua.pt	pwd	SECURITY ENFORCER
tiagocmendes@ua.pt	pwd	TEAM LEADER

3. Funções da equipa

De modo a coordenar o nosso trabalho em equipa, decidimos atribuir uma determinada função a cada um dos elementos. Essa atribuição foi feita de seguinte forma:

Funções:

- **Product Owner:** Diogo Silva
- **Team Manager:** Vasco Ramos
- **Architect:** Tiago Mendes
- **DevOps Master:** João Vasconcelos

Para além disto, decidimos segmentar o trabalho a fazer. Para isto, optámos por dividir a nossa equipa em 2 sub-equipas, cada uma com um foco principal. Isto é, uma focou-se mais no desenvolvimento do Backend, DevOps e CI; enquanto que a outra ficou responsável por todo o desenvolvimento do Frontend:

Equipa de desenvolvimento:

- **Frontend Developers:** Diogo Silva e Tiago Mendes
- **Backend Developers:** João Vasconcelos e Vasco Ramos

4. Metodologias e Processos

4.1 Gestão de *Backlog*

Para a gestão do desenvolvimento do sistema, decidimos utilizar a Metodologia Agile Scrum e, como *software* de suporte, decidimos utilizar Jira. Esta escolha deveu-se, principalmente, ao facto do Jira ser uma ferramenta usada mundialmente, uma das mais conceituadas para gestão de ciclos agile, e, portanto, vimos, neste projeto, uma excelente oportunidade para nos familiarizarmos com a mesma.

4.2 Desenvolvimento com *Feature-Branches*

Tendo em conta experiências passadas, decidimos estruturar o desenvolvimento da nossa solução através do Git, usando um *workflow* de *features* por *branch*. Com isto, comprometemo-nos com um desenvolvimento mais organizado, de forma a permitir que toda a equipa facilmente percebesse, a cada momento, em que fase do desenvolvimento estávamos.

Para a nomeação das branches seguimos um standard bastante utilizado. As novas features seriam desenvolvidas em *branches* nomeadas **feature/nome_da_feature**. Por outro lado, sempre que era necessário corrigir algum *bug*, esta correção seria desenvolvida numa branch nomeada **hotfix/fix_name**.

4.3 DevOps e CI

Decidimos escolher o GitLab como repositório principal do nosso projeto por 2 motivos principais.

Primeiro, porque todos os elementos da equipa estavam habituados a utilizar o GitHub e assim tivemos oportunidade de explorar e aprender uma ferramenta nova.

Segundo, porque um dos objetivos do projeto era implementar ***Continuous Integration*** e o GitLab permitiu-nos fazer isso.

Neste sentido, para cumprir o objetivo de implementar **CI**, decidimos usar a pipeline fornecida pelo GitLab. Para tal, tivemos que criar um ficheiro do tipo **.gitlab-ci.yml** para gerir o processo de compilação e inicialização do nosso projeto .

Esta funcionalidade foi muito importante nos *merge requests* para perceber se o novo código poderia ser integrado com o atual de forma correta. Caso a pipeline realizasse o build com sucesso, podíamos avançar com a integração do novo código.

Para cumprirmos o objetivo de dar deploy à nossa aplicação em containers, decidimos usar o Docker. Esta escolha deveu-se ao facto de já termos utilizado e explorado esta tecnologia durante as aulas práticas de IES. Ao usar *containers* como a nossa estratégia de deployment, conseguimos iniciar e terminar a nossa aplicação com mais facilidade, o que facilitou o desenvolvimento da pipeline e o deploy para a máquina fornecida pelos docentes.

5. Conceito do produto

5.1 Visão

A ideia para o nosso produto surgiu de uma conversa com o professor José Luis Oliveira onde nos foi sugerido desenvolver um sistema em que fosse possível monitorizar acessos dentro de uma empresa.

Decidimos adotar a ideia e foi nesse momento que surgiu o Intrusion Tracker. O nosso sistema é um software de gestão de acessos de empresas onde é possível monitorizar os acessos em tempo real dos funcionários nas várias salas e divisões da empresa. Ao receber esses acessos, o nosso produto analisa e conclui se se trata de um acesso indevido ou autorizado.

No caso de ser um acesso não autorizado, os administradores e a equipa de segurança são notificados através de emails e de push notifications para a aplicação associada ao nosso produto.

Além disso, propusemo-nos a permitir reservas de salas, visualização de todos os acessos na empresa, visualização de gráficos com análise dos dados de acessos, edição de equipas, departamentos e salas e criação e eliminação destas entidades.

Pretendíamos, também, implementar algumas funcionalidades extra além das mencionadas em cima como geração de relatórios automáticos, revogação de acessos e geração de hotmaps que permitissem visualizar onde se localizam a maioria dos acessos dentro da empresa; contudo, por motivos de tempo decidimos não implementar estas funcionalidades e focámo-nos no desenvolvimento das funcionalidades base, de forma correta e funcional.

5.2 Personas

Na modelação do nosso produto, definimos 3 Personas distintas:

- **Administradores:**

Os administradores pretendem ter uma visão geral da empresa. Por isso podem visualizar os acessos dos colaboradores da empresa, salas, departamentos, atividades suspeitas entre outros tipos de dados. Assim, conseguem ter toda esta informação concentrada numa única plataforma.

- **Equipa de segurança:**

A equipa de segurança tem como objetivo monitorizar as atividades dos funcionários da empresa. Por isso, conseguem visualizar os funcionário suspeitos, os acessos às várias salas dentro da empresa, são notificados através de email e push notification quando uma pessoa suspeita realiza um acesso indevido, entre outras funcionalidades.

- **Team leader:**

O team leader tem como motivação principal poder gerir a sua equipa de uma forma intuitiva e simples. Deste modo, consegue reservar salas, ver os funcionários que constituem a sua equipa, entre outras funcionalidades.

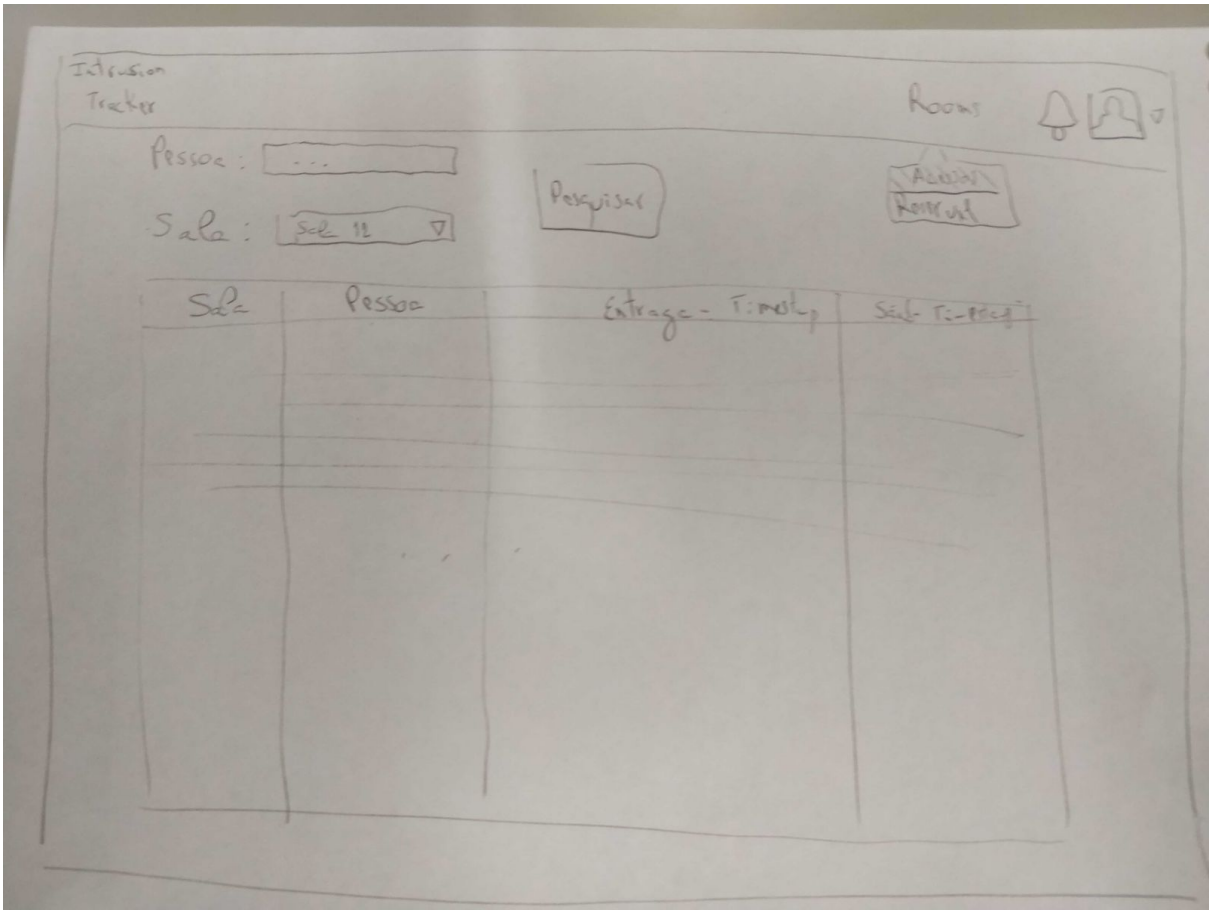
5.3 Cenários principais

Ao definir as personas que iriam utilizar a nossa aplicação, decidimos definir um conjunto de cenários de utilização associados a protótipos em papel.

Dois dos cenários principais são:

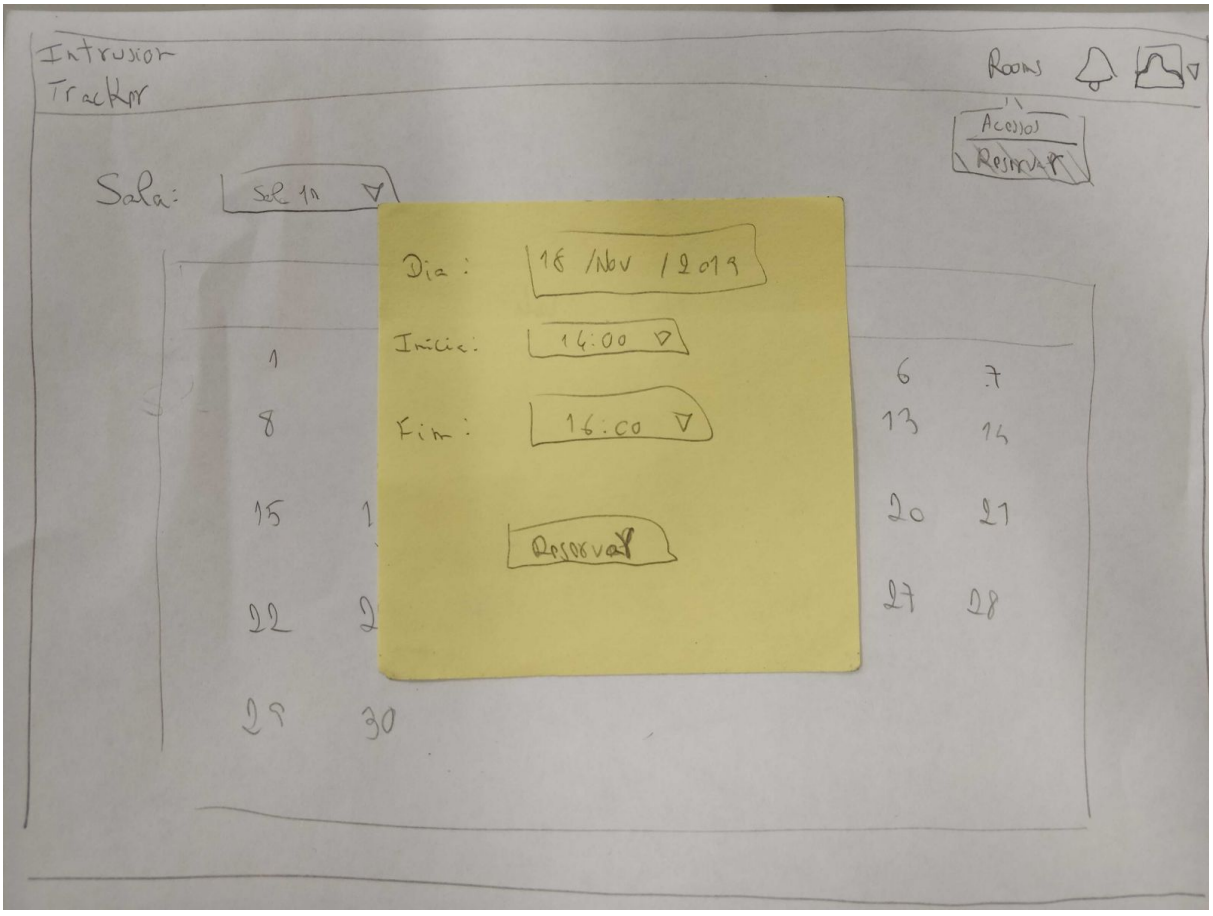
- **A equipa de administração, equipa de segurança e team leader conseguem monitorizar os acessos:**

A equipa de administração, equipa de segurança e team leader querem ter acesso aos logs de entrada e saída, para poderem detetar atividades suspeitas.



- **A equipa de administração, equipa de segurança e team leader conseguem reservar uma sala específica para uma data e hora específica:**

A equipa de administração, equipa de segurança e team leader podem reservar uma sala para uma data e hora específica, não permitindo o acesso a qualquer pessoa que não faça parte da reunião.



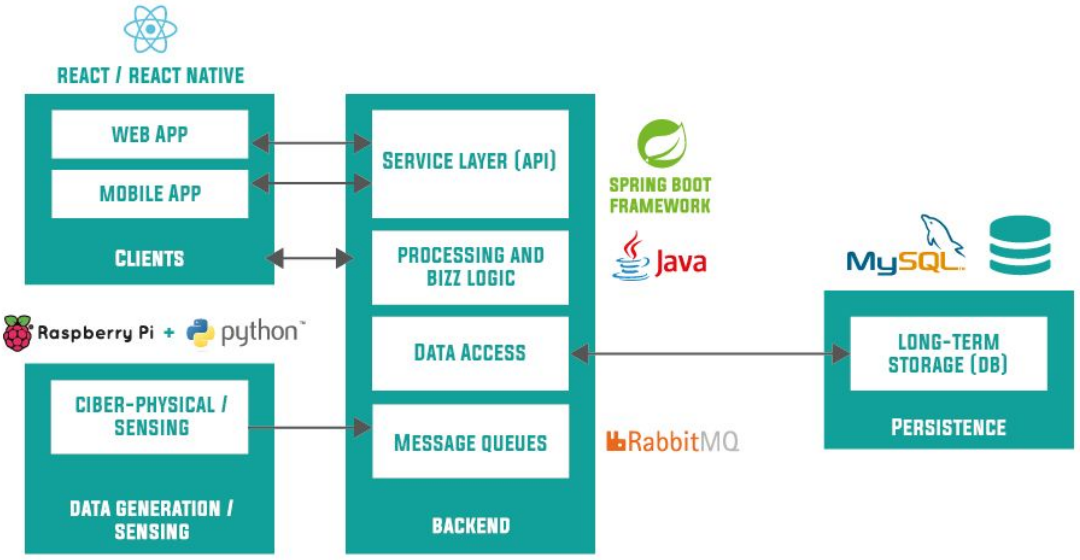
6. Arquitetura

6.1 Necessidades e Restrições

Quando pensámos na modelação do nosso sistema, deparámo-nos com alguns problemas que teriam de ser solucionados. Dois dos principais problemas que foram encontrados vão ser agora analisados em detalhe:

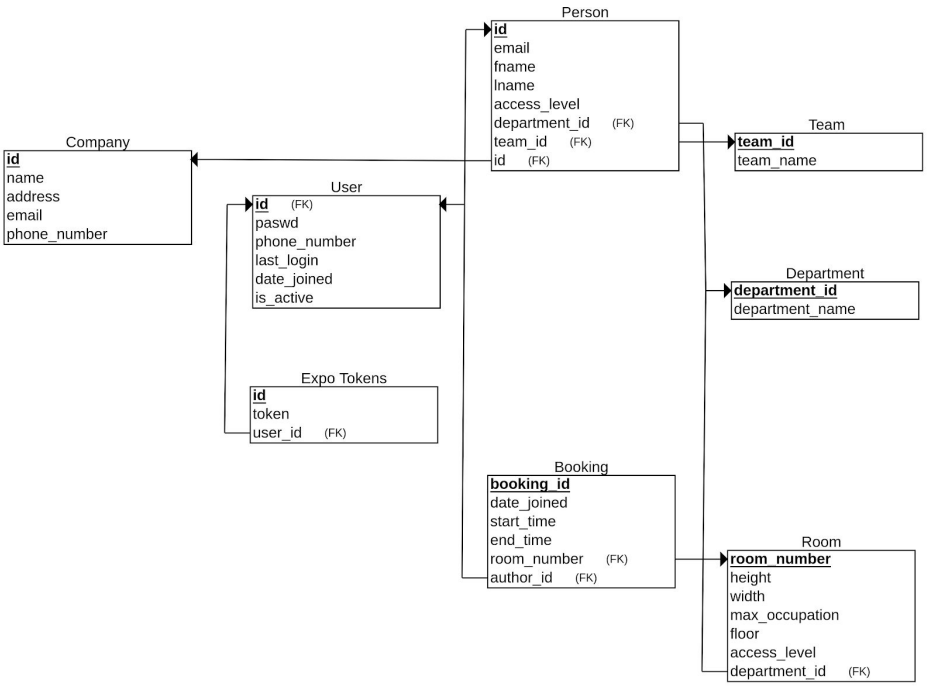
- **Serviço de Emails:** devido a experiências passadas, sabíamos de antemão que não iríamos conseguir enviar emails dentro da rede da UA, devido a restrições de segurança implementadas pelos STIC. Assim, para manter esta funcionalidade, optámos por utilizar uma API externa, SendGrid, que nos permitiu implementar o serviço de emails e ultrapassar esta restrição de meios.
- **Tokens para Push Notifications:** desde o início que queríamos ter um sistema de push notifications para uma aplicação mobile. Contudo, para isto, usámos uma api de notificações onde era necessário uma tabela específica para guardar estes tokens, que acaba por não ter grande ligação com o sistema. Esta necessidade de modelação teve de ser suprida de forma a podermos implementar esta funcionalidade.

6.2 Visão da Arquitetura



6.2.1 Persistência

Para persistência em bases de dados, decidimos usar uma base de dados relacional, **MySQL**. Inicialmente ponderámos utilizar uma base de dados não relacional, mais especificamente MongoDB; contudo, devido a termos entidades com bastantes relações e uma lógica maioritariamente transacional, percebemos que a solução mais adequada para o nosso problema seria mesmo uma base de dados relacional.



6.2.2 Backend

Data Access: para o layer de acesso aos dados em persistência usámos **Java Persistence API**, uma api simples de tradução de queries alto nível para queries SQL.

Lógica de Negócio + Serviços: neste layer usámos Spring. Inicialmente, a equipa pensou em usar Django, sendo a principal razão dessa escolha o facto de um dos elementos ter conhecimento extenso nesta framework pela realização de um estágio e do serviço de backend para o DETI4devs realizado no DETI em Outubro. Contudo, acabámos por optar por Spring para aprendermos uma nova tecnologia e uma nova abordagem às REST API que é, na realidade, um dos objetivos da cadeira e deste projeto.

Message Queues: para lidar com as streams de dados decidimos usar RabbitMQ, um dos mais populares *message brokers* open source, em que alguns dos principais benefícios são: mensagens assíncronas, preparação para *cloud* e sistemas de gestão e monitorização.

6.2.3 Geração de Dados

Para simularmos os nossos streams de dados, decidimos utilizar um **Raspberry3**, onde correm todos os nossos scripts de simulação e geração de dados.

6.2.4 Clientes

De forma a fornecermos aos clientes uma forma fácil e apelativa de aceder aos nossos serviços, criámos duas interfaces - uma Web-App e uma Mobile-App -, sendo a primeira desenvolvida em **React** e a segunda em **React Native**. Estas plataformas foram desenhadas para criar uma ligação limpa e suave entre as nossas funcionalidades de backend e o utilizador, tendo sido dada prioridade ao design minimalista, funcional, e intuitivo, utilizando (e modificando) alguns templates.

7. Implementação (Extras)

7.1 Aplicação mobile e push notifications

Quando formulámos a ideia inicial, decidimos fazer um esforço para implementar o serviço de push notifications, pois seria essencial para notificar de forma rápida e intuitiva os administradores e equipa de segurança da empresa.

Com esse objetivo em mente, e tal como mencionamos anteriormente, decidimos desenvolver uma aplicação mobile em React Native onde fosse possível visualizar alguns dados essenciais para a empresa, tais como, número de departamentos, salas, equipas, acessos indevidos, funcionários e funcionários suspeitos.

Após o desenvolvimento da base da aplicação, decidimos fazer um esforço para conseguirmos que a nossa app recebesse push notifications. Para isso, usámos o serviço fornecido por uma framework de React Native, Expo.

Quando o utilizador realiza o login na nossa aplicação mobile, guardamos um token associado ao seu user id na base de dados. Quando precisamos de enviar uma notificação, vamos buscar os tokens dos users para os quais pretendemos enviar uma notificação e consumimos um endpoint fornecido pelo Expo que irá tratar do envio da mesma.

Quando o utilizador realiza o logout da aplicação mobile, eliminamos o seu token da base de dados e assim ele não irá receber notificações quando não tem sessão iniciada.

7.2 Documentação da REST API

Sempre foi nosso objetivo termos uma documentação do serviço disponível na nossa Rest Api. Para isso, decidimos usar o sistema do OpenApi do Swagger. Inicialmente pensámos fazer isto num processo manual; contudo, por falta de tempo, decidimos investigar mais um pouco e percebemos que, através de anotações próprias no Spring conseguíamos criar toda a documentação de forma bastante automática e eficiente em consumo de tempo.

A nossa documentação está disponível online no seguinte website: http://192.168.160.220:8080/api/v1/swagger-ui.html.

8. Conclusão

Após a realização deste trabalho, concluímos que os objetivos propostos foram, de uma forma geral, alcançados com sucesso.

A execução deste projeto foi, sem dúvida, um dos melhores momentos de aprendizagem e autonomia até hoje no nosso percurso académico. Desde início, decidimos que não nos queríamos acomodar e acabámos por nunca escolher a abordagem mais simples ou rápida, num esforço de uma aprendizagem mais profunda de diferentes tecnologias. Um exemplo deste esforço foi a implementação da REST API em Spring. Apesar de já existir conhecimento prévio na framework Django, decidimos usar Spring e foi, definitivamente, um desafio. Começando pela estrutura de linguagem mais verbosa e documentação menos organizada, sentimos algumas dificuldades iniciais.

Contudo, e para conclusão final, consideramos que a execução deste trabalho foi um caso de sucesso e que, sempre que houve problemas ou dificuldades, conseguimos trabalhar para os superar e, no fim, conseguimos produzir um sistema funcional e que vai de encontro à nossa ideia inicial.