Instituto Politécnico do Cávado e do Ave

Projeto Aplicado

***Relatório de Projeto Aplicado***

Licenciatura em Engenharia de Sistemas Informáticos

Diogo Oliveira

Nº 21111

Joshua Jones

Nº 21116

Leandro Matos

Nº 21124

**Conteúdo**

[Lista de Figuras 4](#_Toc124881780)

[1. Introdução 7](#_Toc124881781)

[1.1. Enquadramento 7](#_Toc124881782)

[1.2. Objetivos 7](#_Toc124881783)

[1.3. Estrutura do documento 7](#_Toc124881784)

[2. Contextualização 8](#_Toc124881785)

[3. Estado de Arte 9](#_Toc124881786)

[*3.1.* *Mapsted* 9](#_Toc124881787)

[*3.2.* *Nearmotion* 9](#_Toc124881788)

[*3.3.* *indoo.rs* 10](#_Toc124881789)

[*3.4.* *what3words* 10](#_Toc124881790)

[4. Proposta do Sistema 11](#_Toc124881791)

[5. Prototipagem 12](#_Toc124881792)

[6. Diagramas 14](#_Toc124881793)

[6.1. Diagrama Casos de Uso 14](#_Toc124881794)

[6.2. Diagrama de Atividades 16](#_Toc124881795)

[6.3. Diagrama de Sequência por Ecrã 17](#_Toc124881796)

[6.4. Diagrama de Entidade Relação 21](#_Toc124881797)

[6.5. Diagrama de Atividades 22](#_Toc124881798)

[7. Mockups 23](#_Toc124881799)

[8. Implementação API 25](#_Toc124881800)

[8.1. Source 25](#_Toc124881801)

[8.1.1. Config 25](#_Toc124881802)

[8.1.2. Errors 27](#_Toc124881803)

[8.1.3. Routes 27](#_Toc124881804)

[8.1.4. Services 28](#_Toc124881805)

[8.2. Services/utilizador.js 29](#_Toc124881806)

[8.3. Routes/utilizadores.js 30](#_Toc124881807)

[8.4. Middlewares.js 31](#_Toc124881808)

[8.5. Passport.js 31](#_Toc124881809)

[8.6. Router.js 32](#_Toc124881810)

[8.7. Config/routes.js 33](#_Toc124881811)

[9. Aplicação Movel 34](#_Toc124881812)

[10. Conclusão 45](#_Toc124881813)

[11. Bibliografia: 46](#_Toc124881814)

# Lista de Figuras

[Figura 1 - Mapsted 9](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881815)

[Figura 2 - Nearmotion 9](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881816)

[Figura 3 - indoo.rs 10](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881817)

[Figura 4 - what3words 10](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881818)

[Figura 5 - Ecrã Mapa 12](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881819)

[Figura 6 - Ecra Login 12](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881820)

[Figura 7 - Ecrã Horário 13](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881821)

[Figura 8 - Ecrã Rota 13](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881822)

[Figura 9 - Diagrama Casos de Uso 14](#_Toc124881823)

[Figura 10 - Comportamento Normal Diagrama Casos de Uso 14](#_Toc124881824)

[Figura 11 - Diagrama Caso de Uso Login 15](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881825)

[Figura 12 - Diagrama de Atividades 16](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881826)

[Figura 13 - Diagrama de Sequência Ecrã Registo 17](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881827)

[Figura 14 - Diagrama de Sequência Ecrã Login 18](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881828)

[Figura 15 - Diagrama de Sequência Reset Password 18](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881829)

[Figura 16 - Diagrama de Sequência Ecrã Perfil 19](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881830)

[Figura 17 - Diagrama de Sequência Ecrã Horário 19](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881831)

[Figura 18 - Diagrama de Sequência Ecrã Pesquisa 20](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881832)

[Figura 19 - Diagrama de Entidade Relação 21](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881833)

[Figura 20 - Diagrama de Atividades Login 22](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881834)

[Figura 21 - Diagrama de Atividades Percurso 22](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881835)

[Figura 22 - Mockup Reset Password 23](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881836)

[Figura 23 - Mockup Login 23](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881837)

[Figura 24 - Mockup Register 23](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881838)

[Figura 25 - Mockup Encontrar Sala 24](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881839)

[Figura 26 - Mockup Ecrã Inicial 24](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881840)

[Figura 27 - Mockup Perfil 24](#_Toc124881841)

[Figura 28 - Mockup Horário 24](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881842)

[Figura 29- source -config 25](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881843)

[Figura 30 - source - errors 27](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881844)

[Figura 31 - source - routes 27](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881845)

[Figura 32 - source - services 28](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881846)

[Figura 33 - Services/utilizador 29](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881847)

[Figura 34 - Routes/utilizadores 30](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881848)

[Figura 35 - Middlewares 31](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881849)

[Figura 36 - Passport 31](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881850)

[Figura 37 - Router 32](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881851)

[Figura 38 - Config/routes 33](file:///C:\Users\ossie\Desktop\Projeto-Aplicado\Relatorio_Projeto_Aplicado.docx#_Toc124881852)

[Figura 39 - Código database.kt 35](#_Toc124881853)

[Figura 40 - Código Sala Dao 36](#_Toc124881854)

[Figura 41- Código LoginResponse e LoginRequest 37](#_Toc124881855)

[Figura 42 - Código Classe Dados Sala 37](#_Toc124881856)

[Figura 43 - Código LoginActivity PT1 38](#_Toc124881857)

[Figura 44 - Código LoginActivity PT2 38](#_Toc124881858)

[Figura 45 - Código ClassroomFinderActivity PT 1 39](#_Toc124881859)

[Figura 46 - Código ClassroomFinderActivity PT 2 40](#_Toc124881860)

[Figura 47 – Códigg Classe RoutingActivity 41](#_Toc124881861)

[Figura 48 - Código PerfilActivity 42](#_Toc124881862)

[Figura 49 - Login XML PT 1 43](#_Toc124881863)

[Figura 50 - Login XML PT 2 44](#_Toc124881864)

[Figura 51 - Activity Login 44](#_Toc124881865)

**Lista de Acronimos:**

* API: Application Programming Interface;
* UI: User Interface;
* IPCA: Instituto Politécnico do Cávado e do Ave;
* IPS: Indoor Positioning System;
* SiGES: Sistema de Gestão de Ensino Superior;
* SIG: Sistema de Informação Geográfica.

# Introdução

Este projeto consiste num IPS com localização exata dos utilizadores dentro das infraestruturas do Instituto Politécnico do Cávado e do Ave. Esta aplicação permitirá, com acesso a dados facultados pelos Serviços Administrativos, saber o horário do utilizador e fazer a rota otimizada de modo que este chegue ao seu destino.

## Enquadramento

A ideia deste projeto surgiu no primeiro dia de aulas onde foi proposto a dois elementos deste grupo guiar os alunos novos dentro do IPCA. Tendo as cadeiras de Projeto aplicado e também Programação de Dispositivos Móveis decidimos que seria uma mais-valia, tanto para o nosso percurso académico, como também para o IPCA criarmos uma aplicação que conseguisse certificar que nenhum aluno ou docente teria algum problema a encontrar o seu destino no IPCA.

## Objetivos

O objetivo deste projeto é conseguirmos ter as capacidades para criarmos uma aplicação que satisfaça os objetivos que temos para a mesma. Ao concretizarmos esses objetivos iremos ganhar vários conhecimentos, tanto a nível da programação, como competências para o nosso futuro profissional. Estas competências a nível profissional serão adquiridas trabalhando em grupo, e desempenhado papeis associados ao mundo do trabalho e trocando papeis entre elementos do grupo para cada um sair deste projeto com as competências desse papel.

## Estrutura do documento

O documento encontra-se organizado em 5 capítulos sendo estes a Introdução, Estado de Arte, Conteúdo dos Ficheiros, Realização do Trabalho, Conclusão e Referencias.

# Contextualização

Quanto à contextualização será abordado o cliente, utilizadores, melhorias no dia a dia dos utilizadores e as partes interessadas e não interessadas neste projeto.

1. Cliente:

O cliente do projeto é o IPCA.

1. Utilizadores:

Os utilizadores do projeto são os alunos, docentes e visitantes do IPCA.

1. Melhorias no dia a dia dos utilizadores:

O utilizador desta aplicação nunca terá problemas em encontrar a sala onde pretende estar e consoante o tipo de utilizador poderá saber exatamente onde tem de estar em especificas horas.

1. Partes interessadas:

As partes interessadas deste projeto é o IPCA, financiador do mesmo e único cliente visto que o projeto é feito para o Campus de Barcelos.

# Estado de Arte

No presente capítulo será apresentado o estado da arte, relativo a tecnologias, aplicações e recursos que atualmente se encontram disponíveis para a realização/implementação de metodologias colaborativas.

## *Mapsted*

Figura - Mapsted

A Mapsted é uma empresa inovadora de tecnologia baseada no Canadá que tem clientes em várias indústrias tais como centros comerciais, universidades, hospitais, estações de comboio, resorts, etc., que usam uma tecnologia avançada para a localização dentro e fora de edifícios e que não necessita de *hardware* sendo so necessário um *smartphone* sem custos acrescentados.

## *Nearmotion*

Figura - Nearmotion

A Nearmotion é uma empresa baseada na Arabia Saudita que em parceria com a *Saudi Aramco Entrepeneurship Ventures* para providenciar soluções de confiança que capacitam organizações governamentais e setores empresariais com ferramentas pioneiras que levam o envolvimento e experiência dos clientes a um novo nível.

Edifícios inteligentes que interagem com os visitantes de acordo com a sua localização, dão-lhes as boas-vindas à chegada, orientam-nos passo a passo através da orientação digital até ao seu destino e recompensam-nos com ofertas e cupões de acordo com as suas preferências. É isso que o NEARMOTION oferece por meio de uma plataforma fácil de usar que permite que os locais ofereçam níveis avançados de experiência aos visitantes e tornem a sua visita uma viagem inesquecível.

## *indoo.rs*

Figura - indoo.rs

A indoo.rs foi fundada em 2010 com o nome de *CustomLBS* por 2 alunos, *Bernd Gruber* e *Markus Krainz*.

A ideia nasceu quando, numa escala extremamente longa num aeroporto, *Bernd* achou mais difícil do que deveria ser localizar os edifícios e salas certas.

Desde então, a empresa cresceu para 20 funcionários, com sede e um escritório de desenvolvimento em Viena e um escritório de vendas em San Francisco.

Em fevereiro de 2019, a empresa foi adquirida pela *Esri*, fornecedora líder internacional de *software* SIG (sistema de informação geográfica).

## Uma imagem com texto Descrição gerada automaticamente*what3words*

Figura - what3words

Endereços de ruas não são precisos o suficiente para especificar locais precisos, como entradas de prédios, e não existem para parques e muitas áreas rurais.

Isso dificulta a localização de lugares e impede que as pessoas descrevam exatamente onde a ajuda é necessária em caso de emergência.

Para resolver esses problemas foi criado o what3words, que consiste em dividir o mundo em 3 metros quadrados e dá a cada quadrado uma combinação única de três palavras. É a maneira mais fácil de encontrar e partilhar localizações exatas.

# Proposta do Sistema

1. Requisitos Funcionais:
   1. O utilizador devia de ser capaz de conseguir chegar ao seu destino, independentemente de onde se situar dentro do campus;
   2. O utilizador deve ser capaz de inserir as suas credenciais para aceder às funcionalidades da aplicação;
   3. O utilizador deve ser capaz de aceder ao seu horário de aulas;
   4. O sistema deve ser capaz de obter posição do utilizador;
   5. O sistema deve ser capaz de atualizar a posição atual do utilizador;
   6. O sistema deve ser capaz de obter os horários de todos os cursos e anos do IPCA;
   7. O sistema deve ser capaz de obter e reconhecer a planta do IPCA.
2. Requisitos Não Funcionais
   1. Usabilidade: Um utilizador deverá conseguir operar o sistema sem necessitar de um guia do mesmo;
   2. Eficiência: O programa deve ser capaz de conseguir atualizar as rotas sem perder o destino;
   3. Confiabilidade: O sistema terá de estar sempre operacional, tendo as exceções de raras pausas de manutenção;
   4. Portabilidade: O programa deverá conseguir correr em todos os dispositivos *Android*;
   5. Implementação: O programa deverá ser desenvolvido na linguagem *Kotlin*;
   6. Interoperabilidade: O sistema deverá obter as suas informações através do *Maps* da *Google* e com o *SiGES* (Sistema de Gestão de Ensino Superior) para obter os horários do docentes e alunos;
   7. Legais: O programa deverá atender às normas legais, tais como padrões, leis, etc.

# Prototipagem

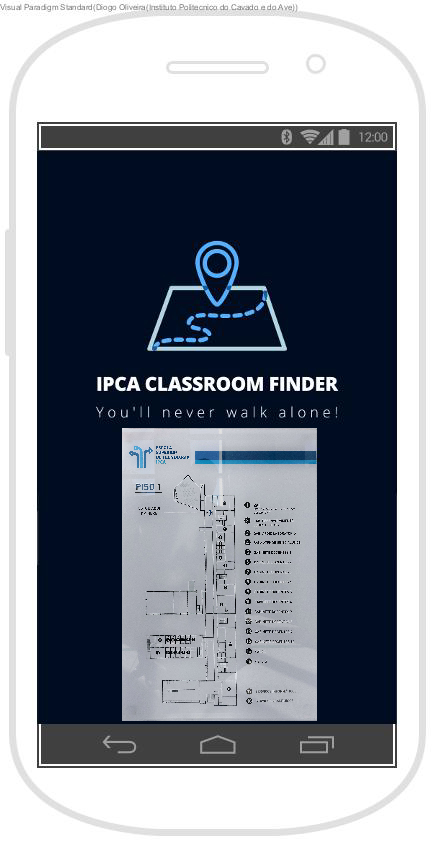


Figura - Ecrã Mapa

Figura - Ecra Login

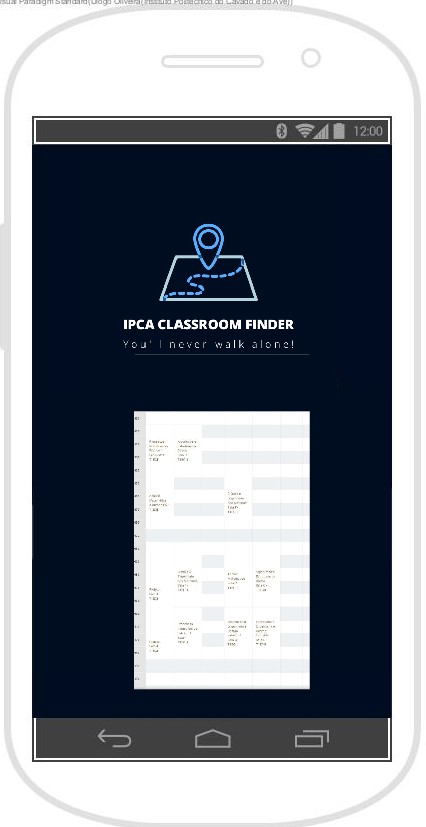


Figura - Ecrã Horário

Figura - Ecrã Rota

# Diagramas

## Diagrama Casos de Uso

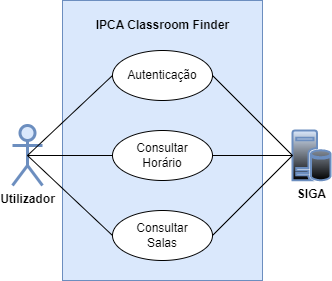


Figura - Diagrama Casos de Uso

Uma imagem com mesa

Descrição gerada automaticamente

Figura - Comportamento Normal Diagrama Casos de Uso

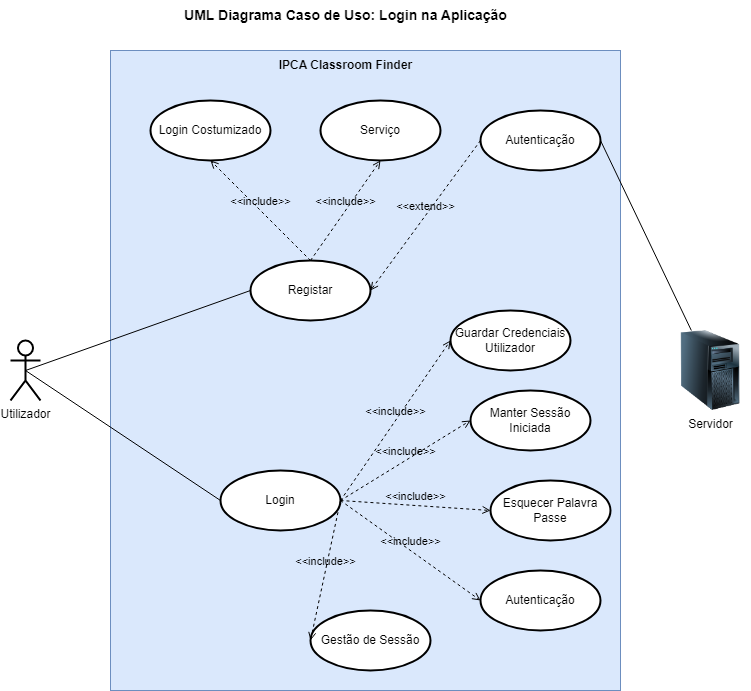


Figura - Diagrama Caso de Uso Login

## Diagrama de Atividades

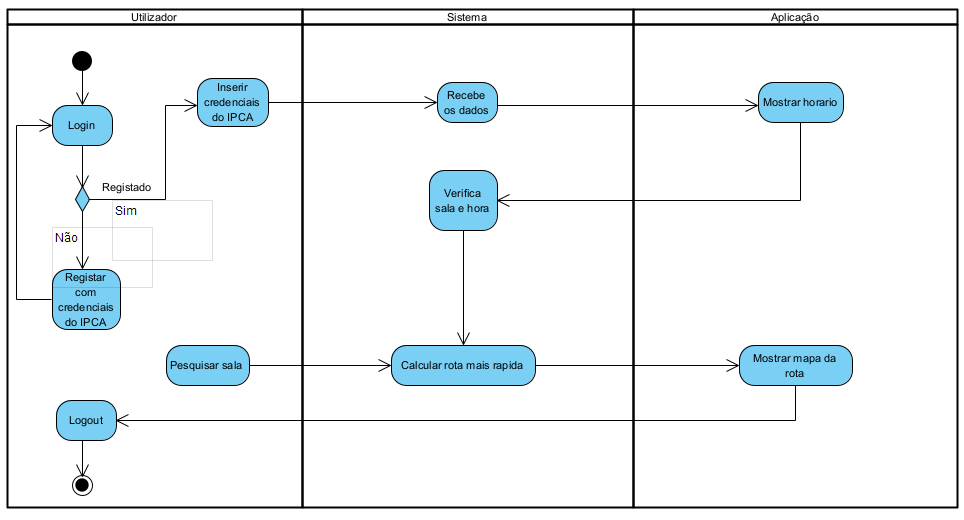


Figura - Diagrama de Atividades

## Diagrama de Sequência por Ecrã

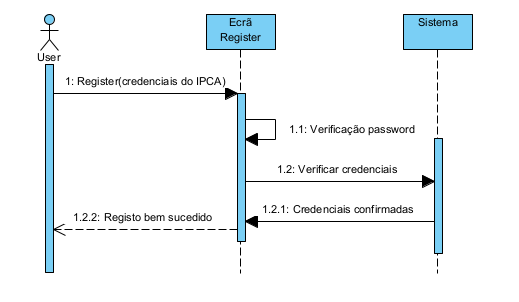


Figura - Diagrama de Sequência Ecrã Registo

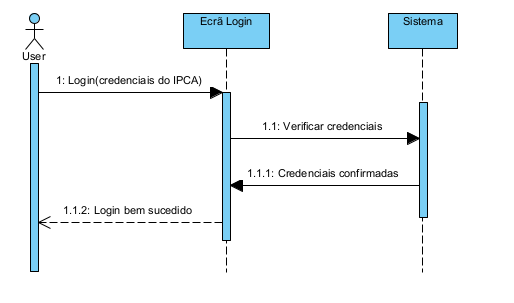


Figura - Diagrama de Sequência Ecrã Login

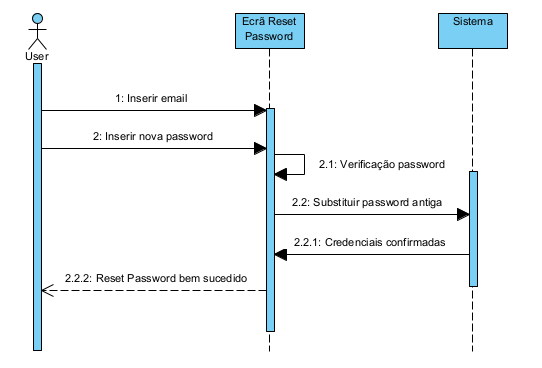


Figura - Diagrama de Sequência Reset Password

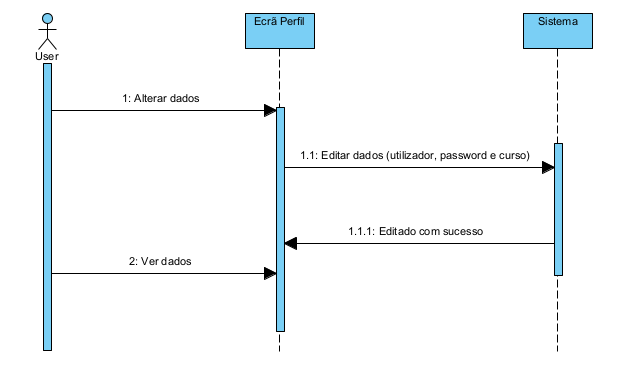


Figura - Diagrama de Sequência Ecrã Perfil

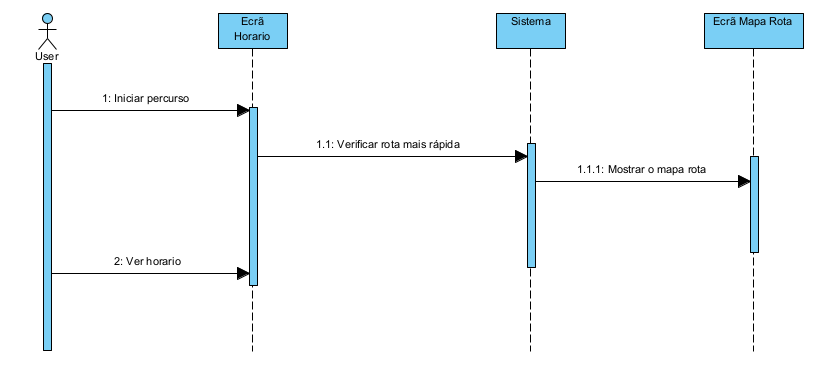


Figura - Diagrama de Sequência Ecrã Horário

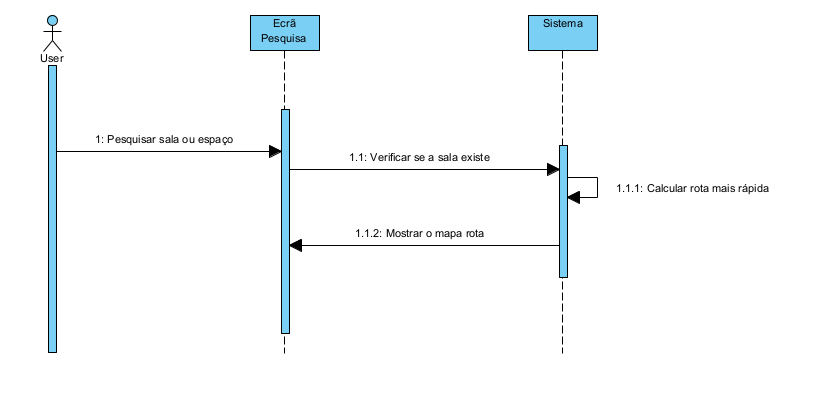


Figura - Diagrama de Sequência Ecrã Pesquisa

## Diagrama de Entidade Relação

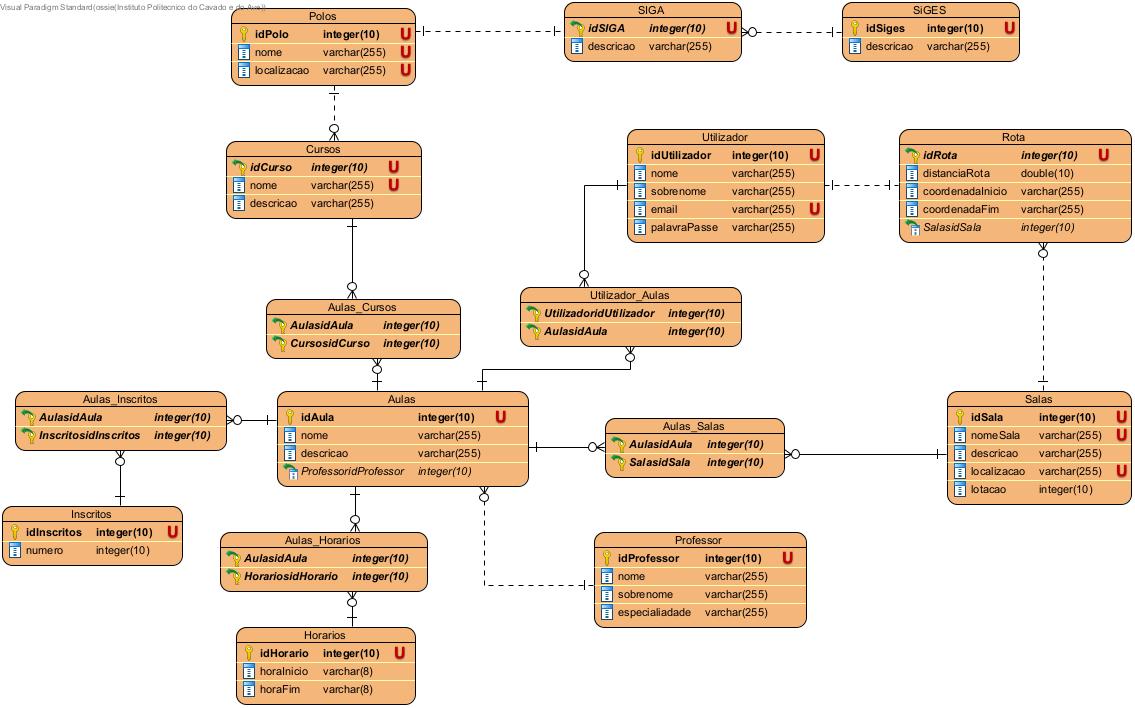


Figura - Diagrama de Entidade Relação

## Diagrama de Atividades

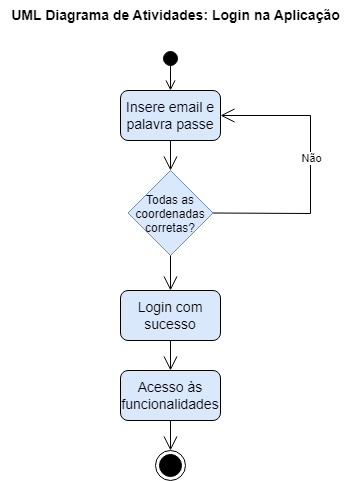


Figura - Diagrama de Atividades Login

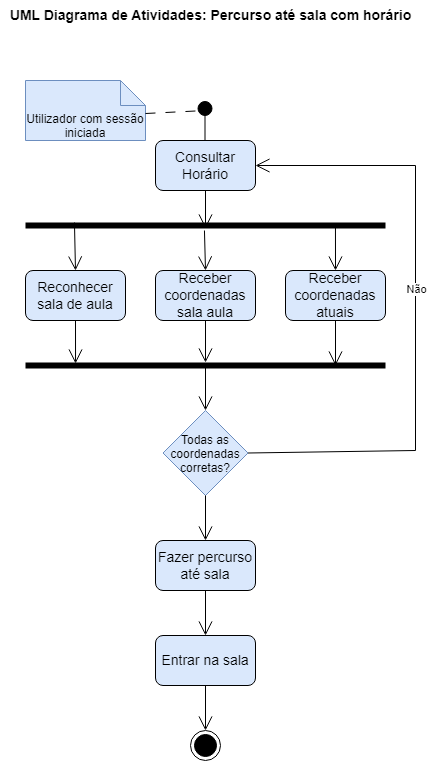


Figura - Diagrama de Atividades Percurso

# Mockups

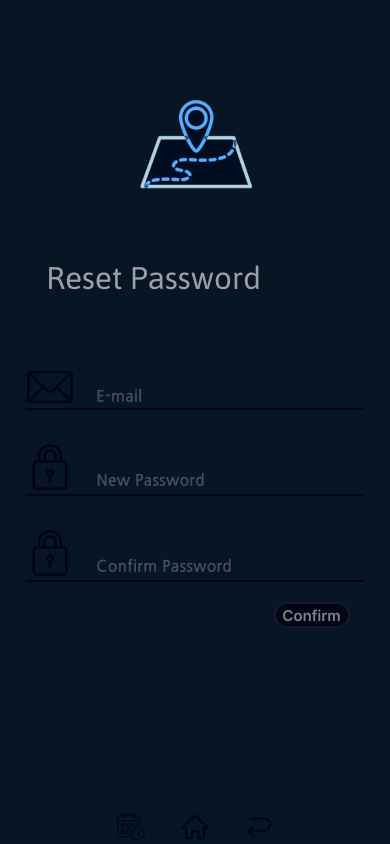


Figura - Mockup Reset Password

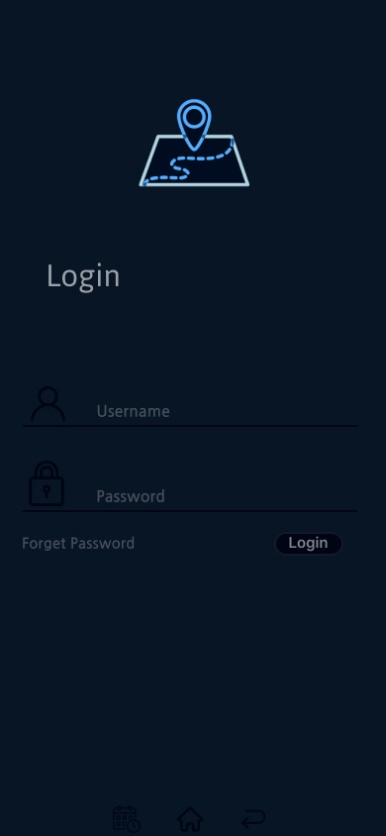


Figura - Mockup Login

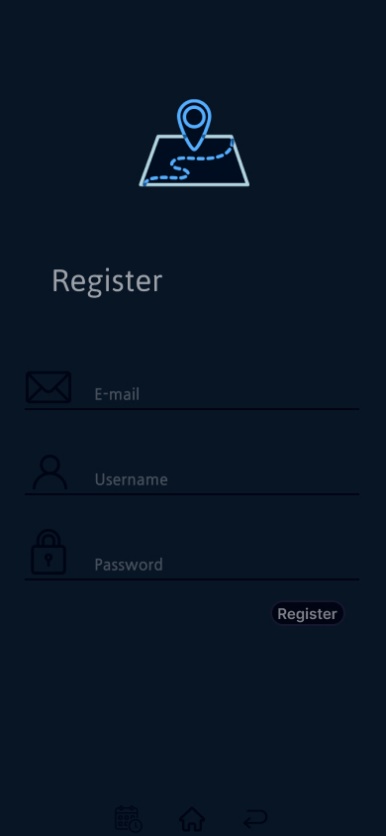
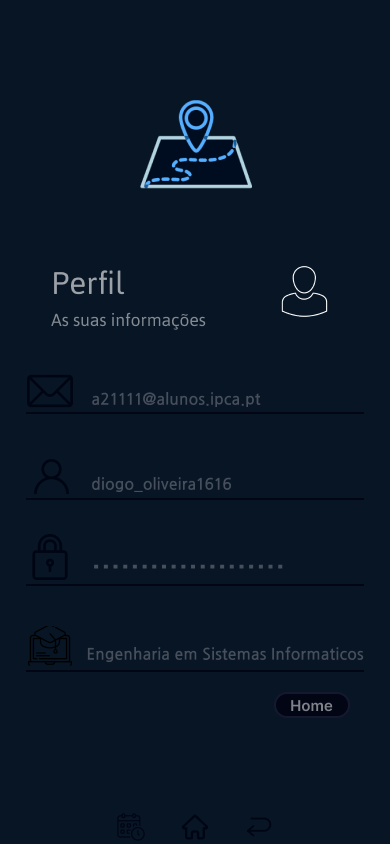


Figura - Mockup Register



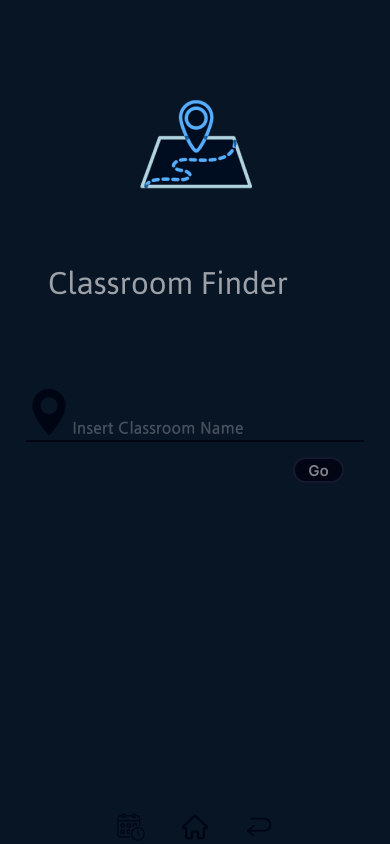


Figura - Mockup Encontrar Sala

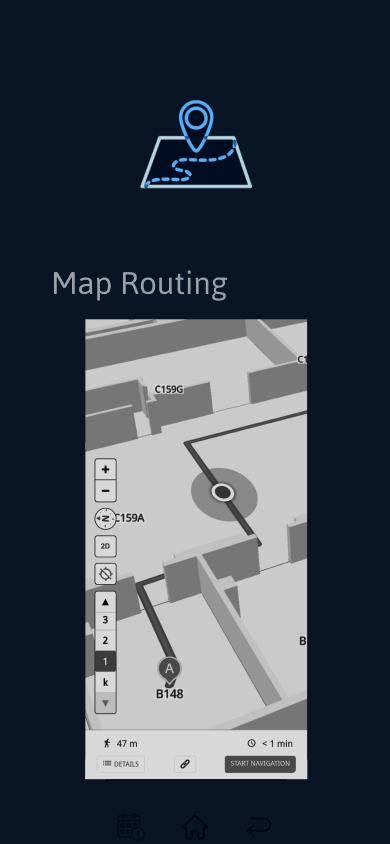
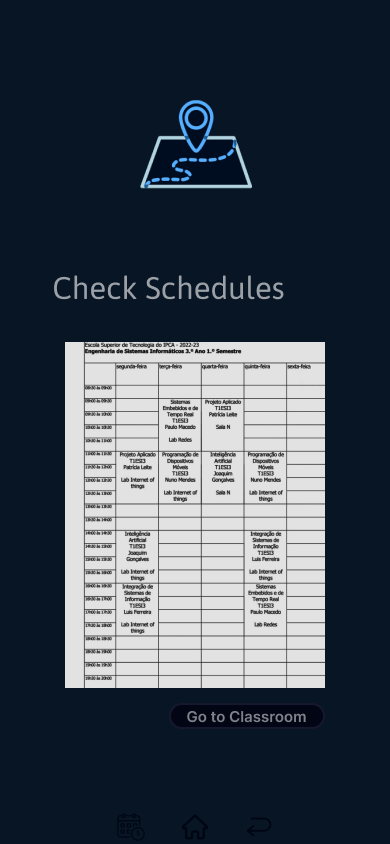


Figura - Mockup Ecrã Inicial

Figura - Mockup Perfil

Figura - Mockup Horário



# Implementação API

## Source

### Config



Figura - source -config

A pasta config guarda os ficheiros *.js* que configuram a API

**Middlewares:**

No contexto do desenvolvimento web, *middleware* refere se ao software que fica entre o sistema operacional e uma aplicação, realizando uma função específica ou um conjunto de funções. Numa aplicação web, as funções de *middleware* são funções que têm acesso aos objetos de solicitação e resposta e podem realizar uma variedade de tarefas, como autenticação, log, servir arquivos estáticos, entre outros.

No contexto desta aplicação JavaScript web, as funções de *middleware* são escritas num arquivo chamado middleware.js e usado em conjunto com um framework web, como o Express.js. Essas funções de *middleware* podem ser consideradas como um *request* que tem a habilidade de executar código, tomar decisões e realizar tarefas antes que a solicitação seja tratada pela aplicação principal.

**Passport:**

Passport.js é um middleware de autenticação para o Node.js que oferece uma maneira simples e flexível de autenticar utilizadores e proteger rotas. Ele usa "estratégias" para autenticar solicitações, que são funções que verificam as credenciais de um usuário e fornecem um objeto de usuário após a autenticação bem-sucedida.

O Passport.js suporta muitos mecanismos de autenticação diferentes, chamadas “estratégias”, incluindo OAuth, SAML e OpenID Connect. Ele também fornece uma API simples para autenticar solicitações e suporta mais de 500 mecanismos de autenticação.

**Router:**

Router.js é um módulo de software que é comumente usado em aplicações web para gerir rotas (caminhos para diferentes páginas ou recursos da aplicação). Ele pode ser usado para mapear rotas para diferentes funções de manipulação de requisições HTTP (como GET, POST, PUT e DELETE), que podem ser usadas para ler e escrever dados em um banco de dados ou para realizar outras operações no lado do servidor.

**Routes:**

Um arquivo routes.js é um arquivo de configuração de rotas em uma aplicação web. Ele é usado para definir as rotas e as funções de manipulação de requisições HTTP (como GET, POST, PUT e DELETE) que estão associadas a elas.

### Errors

Uma imagem com texto

Descrição gerada automaticamente

Figura - source - errors

O ficheiro *“Errors”* é constituído por ficheiros *.js* usados para apresentar mensagens de erro sempre que um erro for encontrado num pedido à API.

### Routes

Uma imagem com texto

Descrição gerada automaticamente

Figura - source - routes

A pasta de *Routes* é constituída por todos os ficheiros *JavaScript* com o código que executa o *CRUD* da API, fazendo estas consoantes as instruções criadas nos *Services*.

### Services

Uma imagem com texto

Descrição gerada automaticamente

Figura - source - services

A pasta de *Services* é constituída por código que executa operações de CRUD para cada tabela especifica da base de dados.

## Uma imagem com texto Descrição gerada automaticamenteServices/utilizador.js

Figura - Services/utilizador

Os Services do Utilizador.js fazem 6 operações:

* *findOne:* Aqui, através de um filtro vindo do *frontend* da *API,* o utilizador consegue encontrar outro utilizador na base de dados a partir de um filtro que ele queira usar para pesquisar.
* *getAll:* Código de uma operação GET para o utilizador receber os dados todos de todos os utilizadores que estão inseridos na base de dados.
* *getAllID:* Código de uma operação GET para o utilizador receber os dados todos de um certo utilizador através do seu ID.
* *create:* Código de uma operação POST onde um utilizador insere os dados de um novo utilizador. O campo *idUtilizador* é auto-incrementado, o restante campo tem de ser inserido senão o programa retorna uma mensagem de erro a pedir os dados em falta.
* *update:* Código de uma operação PUT que, a partir de novos dados inseridos, atualiza o utilizador selecionado através de um ID.
* *remove:* Código de uma operação REMOVE que apaga um utilizador a partir de um ID inserido, isto claro se coincidir.

## Routes/utilizadores.js

Uma imagem com texto

Descrição gerada automaticamente

O *Routes/Utilizadores.js* define o que cada

rota faz, consoante as operações criadas nos

*Services* do mesmo.

Figura - Routes/utilizadores

## Middlewares.js

Uma imagem com texto

Descrição gerada automaticamente

Figura - Middlewares

## Passport.js

Uma imagem com texto

Descrição gerada automaticamente

Figura - Passport

## Router.js

Uma imagem com texto

Descrição gerada automaticamente

Figura - Router

O *Router.js* é onde são definidas as rotas que no serviço web vão executar as funções do código dos *Services.*

## Uma imagem com texto Descrição gerada automaticamenteUma imagem com texto Descrição gerada automaticamenteUma imagem com texto Descrição gerada automaticamenteUma imagem com texto Descrição gerada automaticamenteConfig/routes.js

Figura - Config/routes

O código acima tem as rotas todas da API, com as operações que conseguem executar e também as mesmas em cenários onde a pesquisa é por ID.

# Aplicação Movel

***Room:*** *O Room é um framework do Android Studio que facilita a criação de aplicativos com banco de dados SQLite. Ele fornece uma abstração de camada de persistência que permite aos desenvolvedores trabalhar com objetos Java simples ao invés de lidar diretamente com o banco de dados SQLite. O Room também fornece recursos como geração automática de código SQL, validação de consultas e suporte para transações.*

***Retrofit:*** Retrofit GSON é uma extensão do Retrofit, ele é utilizado para realizar requisições HTTP e converter as respostas em objetos Java, especificamente para o formato JSON. Ele utiliza a biblioteca GSON para serializar e desserializar os objetos Java automaticamente. GSON é uma biblioteca de código aberto criada pela Google, que é usada para converter objetos Java para formato JSON e vice-versa. A combinação do Retrofit e GSON permite que os desenvolvedores trabalhem com objetos Java simples ao invés de lidar diretamente com o formato JSON, facilitando a implementação de operações de rede e acesso a dados em aplicativos Android que se comunicam com APIs que retornam dados em formato JSON.

***SQLite:*** SQLite é um banco de dados relacional de código aberto que é usado para armazenar e recuperar dados estruturados. Ele é embutido no sistema operacional Android, o que significa que ele não precisa ser instalado ou configurado separadamente. O SQLite é utilizado para gerenciar dados de aplicativos Android, tais como armazenamento de configurações, usuários, mensagens e outros tipos de dados. Ele é uma ferramenta útil para aplicativos que precisam armazenar dados localmente e trabalhar offline, sem a necessidade de acesso a uma conexão de rede. O Android Studio fornece suporte nativo para trabalhar com o SQLite, incluindo a capacidade de criar e gerenciar tabelas, consultas e transações.

Link do ilustre vídeo de utilização da aplicação: <https://youtu.be/nK-TcMnJOjQ>

**Desenvolvimento do código:**

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código database.kt

Este código cria uma classe de dados para um utilizador, com propriedades para armazenar informações como nome, sobrenome, email e palavra-passe, e uma interface para interagir com a tabela de utilizadores na base de dados. A classe MyDatabase é anotada como uma base de dados Room e fornece uma instância única da base de dados, "utilizador.db", que é construída usando o contexto passado. A interface utilizadorDao fornece métodos para buscar, inserir, atualizar e deletar utilizadores na tabela.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código Sala Dao

Este código fornece uma interface chamada SalaDao que possui métodos para interagir com a tabela sala na base de dados, e uma classe chamada MyDatabaseSala que é anotada como uma base de dados Room, e que possui uma instância única da base de dados chamada "sala.db". A classe MyDatabaseSala também define um método abstrato SalaDao() que retorna uma instância da interface SalaDao e o objeto companheiro fornece um operador invoke() para acessar a instância da base de dados criada.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código LoginResponse e LoginRequest

Este código define duas classes de dados, LoginResponse e LoginRequest. A classe LoginResponse contém dois campos, token e user, enquanto a classe LoginRequest contém dois campos, email e password.

A interface LoginResponseDao contém um método chamado checkLogin que é anotado com @POST e tem uma url especificada("v1/auth/signin"). Este método espera um objeto LoginRequest como parâmetro e retorna uma chamada do tipo LoginResponse.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código Classe Dados Sala

Este código define uma classe de dados chamada sala que contém informações como idSala, nomeSala, descricao, localizacao e lotacao. Esta classe é anotada com @Entity, indicando que será usada para criar uma tabela no banco de dados e o nome da tabela é especificado como "sala". A chave primária para a tabela é idSala.

A interface SalasDAO contém dois métodos, um para buscar todas as salas(getSalas) e outro para buscar salas por id(getSalasByID). Ambos os métodos são anotados com @GET e tem url específica("v1/sala" e "v1/sala/{id}") respectivamente. Eles retornam chamadas do tipo List<sala>.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código LoginActivity PT1

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código LoginActivity PT2

Este código define uma classe chamada LoginActivity que é uma atividade da AppCompatActivity. Ele possui um método onCreate que é chamado quando a atividade é criada. Também é definido um botão de login, os campos de email e password. Quando o botão é clicado, é chamado o método checkLogin que passa os valores dos campos email e password e uma intent para a próxima atividade.

O método checkLogin cria uma conexão com a base de dados e verifica as credenciais passadas, se as credenciais estiverem corretas, ele insere o usuário na base de dados e inicia a próxima atividade, caso contrário, exibe uma mensagem de erro.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código ClassroomFinderActivity PT 1

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código ClassroomFinderActivity PT 2

Este código define uma classe chamada ClassroomFinderActivity que é uma atividade da AppCompatActivity. Ele possui um método onCreate que é chamado quando a atividade é criada. São definidos dois botões, um para ir para a atividade de horários e outro para buscar salas pelo nome. Também é definido um botão para ir para a atividade de perfil. Quando o botão de buscar salas é clicado, é chamado o método checkSala que passa o nome da sala para ser buscada.

O método checkSala cria uma conexão com a base de dados e busca as salas pelo nome passado, se encontrar, insere as salas na base de dados e inicia a próxima atividade, caso contrário, exibe uma mensagem de erro.

Uma imagem com texto

Descrição gerada automaticamente

Figura – Códigg Classe RoutingActivity

Este código define uma classe chamada RoutingActivity que é uma atividade da AppCompatActivity. Ele possui um método onCreate que é chamado quando a atividade é criada. São definidos os dados da sala, como nome, descrição, localização e lotação, para serem exibidos na tela.

Uma imagem com texto

Descrição gerada automaticamente

Figura - Código PerfilActivity

Este código define uma classe chamada PerfilActivity, que é uma atividade da AppCompatActivity. Ele possui um método onCreate que é chamado quando a atividade é criada. É definido os dados do utilizador como nome e email. Também existem dois botões, um para mudar a senha e outro para voltar à tela inicial. Quando esses botões são clicados, são iniciadas novas atividades.

O XML do frontend da nossa aplicação é igual ao das Mockups, tendo exportado os mesmos do *Figma*. Um exemplo de código é o seguinte:

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login XML PT 1

Uma imagem com texto

Descrição gerada automaticamente

Figura - Login XML PT 2

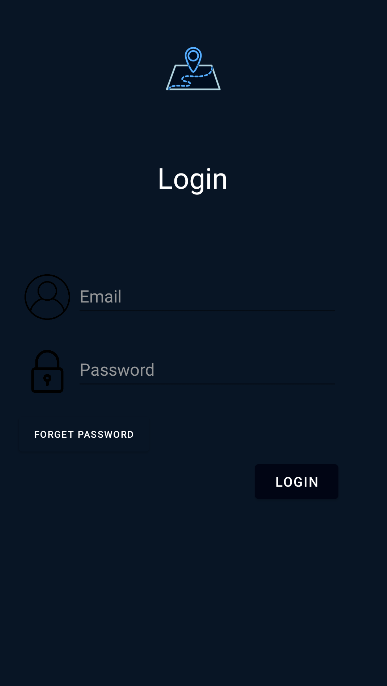


Figura - Activity Login

# Conclusão

Ao longo do desenvolvimento deste trabalho, foi possível consolidar conceitos sobre integração de sistemas de informação, projetar arquiteturas de integração de sistemas, explorar ferramentas de suporte ao desenvolvimento de serviços web, investigar novas tecnologias e frameworks para implementação de serviços web, potenciar a experiência no desenvolvimento de aplicações, explorar funcionalidades de programação em android e aprofundar os conhecimentos adquiridos na unidade curricular. Todo o grupo achou este projeto e esta cadeira uma mais-valia na medida em que englobou diversas unidades curriculares em simultâneo e em concordância para que fosse possível chegar à implementação final.  
 Como grupo tivemos as nossas dificuldades e momentos mais frágeis uma vez que algumas das unidades curriculares e os seus objetivos abordados eram novidade. Foi necessário bastante entreajuda e pesquisa aprofundada para a conclusão deste trabalho prático.  
 Em suma pensamos que como programadores e como pessoas obtivemos valores bastantes positivos para o mundo de trabalho e desenvolvimento pessoal.

# Bibliografia:

Reportório Aulas: <https://elearning2.ipca.pt/>

GitHub Professor Lufer: <https://github.com/luferIPCA/ISI/tree/master/ESI>

Microsoft Azure: <https://azure.microsoft.com/pt-pt/>

SQL Workbench: <https://www.mysql.com/products/workbench/>

Postman: <https://www.postman.com/>

NodeJS: <https://nodejs.org/en/>

Express: <https://www.expresstecnologia.com/>