

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Departamento de Informática



Relatório de Projeto

The Softshares

Lucas

João

Daniel

Cassamo

Francisco

Viseu, 2024

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu
Departamento de Informática

Relatório de Projeto
Curso de Licenciatura em
Engenharia Informática

The Softshares

Ano Letivo 2023/2024

Lucas
João
Daniel
Cassamo
Francisco

Viseu, 2024

Índice

1. Introdução	5
2. Ferramentas e Tecnologias	Erro! Marcador não definido.
3. Estrutura	Erro! Marcador não definido.
4. Requisitos	6
5. Desenvolvimento	11
5.1. Base de Dados	11
5.1.1. <i>Estrutura da Base de Dados</i>	11
5.1.2. <i>PgAdmin</i>	12
5.2. API.....	13
5.2.1. <i>Tecnologias Utilizadas</i>	13
5.2.2. <i>Arquitetura da Aplicação</i>	16
5.2.3. <i>Estrutura</i>	19
5.3. Web.....	21
5.3.1. <i>Estrutura do Projeto</i>	21
5.3.2. <i>Estrutura da Aplicação</i>	22
5.3.3. <i>Rotas</i>	22
5.3.4. <i>Componentes</i>	22
5.3.5. <i>Layout</i>	22
5.3.6. <i>Comunicação com a API</i>	22
5.4. Mobile.....	23

5.4.1. Estrutura do Projeto (João)	23
5.4.2. Detalhes dos conteúdos (João).....	23
5.4.3. Criar conteúdos (Daniel).....	29
5.4.4. Perfil (Daniel)	36
5.4.5. Editar perfil (Daniel).....	40
5.4.6. Calendário (Francisco).....	41
5.4.7. Fórum (Francisco)	41
5.4.8. Listagem por tipo (Francisco).....	41
5.4.9. Integração com o Google e Facebook (Cassamo/Francisco).....	42
5.4.10. Iniciar sessão (Cassamo).....	42
5.4.11. Comunicação com a API (Cassamo).....	42
6. Conclusão	43

Índice de tabelas

Índice de figuras

Figura 1-1. Exemplo de legenda.....**Erro! Marcador não definido.**

1. Introdução

2. Requisitos

Os requisitos funcionais descrevem as principais funcionalidades e comportamentos que o sistema deve ter. Especificam o que o sistema deve fazer para ir de encontro às necessidades dos utilizadores do mesmo. A tabela 1 demonstra os requisitos funcionais gerais e da parte web.

Requisitos Funcionais	
1	Criar Utilizadores
2	Ativar/Inativar utilizadores
3	Associar utilizadores a permissões da aplicação
4	Associar utilizadores a pelo menos um centro
5	Obrigar a atualização de password no primeiro login
6	Enviar email ao utilizador após o seu registo/criação na aplicação
7	Possibilidade de criação de áreas/categorias (Desporto, Formação, Transportes,...)
8	Possibilidade de criação de subáreas/atividades (Futebol, Boleias, Padle, ...)
9	Possibilidade de criação de centros
10	Possibilidade de criação de fóruns de discussão por categoria e/ou por atividade.
11	Associado a cada área/categoria possibilidade de criação de Álbuns.
12	Apresentação do calendário de eventos

13	Na área pessoal de um utilizador normal deve ser possível indicar/definir as áreas e subtópicos preferidos. Depois, sempre que forem publicados conteúdos nessas áreas e subtópicos, o utilizador deve ser notificado
14	Alterar o estado dos formulários a qualquer altura
15	Um administrador, em backoffice, apenas pode criar e moderar conteúdos relacionados com o centro em que é administrador. Os conteúdos moderados e publicados devem aparecer de imediato na aplicação móvel
16	Os Administradores podem editar os conteúdos dos utilizadores, quando estão a fazer a moderação. Por exemplo, podem mudar a área associada ao conteúdo
17	BackOffice adicionar os vários tipos de Categorias/Recomendações/Eventos/Atividades/Espaços. Cada tópico, terá subtópicos de acordo com as áreas de atuação.
18	Validação em backoffice de recomendações/eventos/comentários e outros conteúdos inseridos pelos utilizadores da aplicação móvel.
19	Visão geral no BackOffice em dashboard dividido por área de atuação, atividade mais comentada, atividades mais vistas, nº de tópicos abertos, e recomendações/eventos por validar, etc..
20	Um administrador só pode validar/aprovar/publicar conteúdos do seu centro
Requisitos Funcionais - APP Mobile	
21	Poder efetuar o login de 3 formas distintas
22	Possibilidade de guardar os dados de login, de forma a que não seja obrigado a efetuar o login sempre que utiliza a Aplicação

23	Possibilidade de recuperar password e de cancelar o processo de recuperar password
24	Página “Terminar Sessão” com o comportamento descrito no enunciado do trabalho.
25	Possibilidade de criação de recomendação
26	Possibilidade de criação de evento
27	Apresentação do calendário de eventos
28	Possibilidade de submeter mensagens em fóruns
29	Possibilidade de submeter fotos em álbuns
30	Possibilidade de fazer avaliações (podem ser parametrizáveis) de conteúdos. Por exemplo, de recomendações, comentários, etc.
31	Inserir comentários em cada um dos eventos disponíveis e visíveis
32	Possibilidade de denunciar comentários
33	Possibilidade de avaliar comentários
34	Visualizar eventos nos quais se encontra inscrito
35	Possibilidade de inscrever em eventos ou em grupos associados aos eventos
36	Possibilidade de partilha de uma recomendação/evento. (Exemplo, partilha de um restaurante com um colega/amigo).
37	Disponibilização de preços médios para cada recomendação partilhada
38	Na criação de eventos, recomendações, espaços ou edifícios relativos às atividades, associar geolocalização no Google Maps.

39	Disponibilizar hiperligação para partilha das atividades através das redes sociais ou na plataforma colaborativa (Teams)
40	Gerar notificações aquando de alterações de eventos ao qual se encontra inscrito
41	Gerar notificações quando são colocados comentários novos no painel dos eventos inscritos.
42	O utilizador que criou o anúncio de atividade (Ex: Jogo de Futebol, Encontro BTT, Jogo de Padle) deve receber notificação de qualquer interação que tenha sido efetuada no mesmo (inscrições, questões colocadas, etc.)
43	OS utilizadores normais podem colocar conteúdos/comentários/avaliações/mensagens em todos os centros
44	Visualizar lista de recomendações filtrada por áreas de interesse e por classificação
45	Permitir alterar os dados e acrescentar informação nos eventos do qual sou owner até esse conteúdo ser publicado por um administrador
46	Permitir inserir locais de interesse/recomendações por categoria ou área de atuação.
47	Na área pessoal de um utilizador normal deve ser possível indicar/definir as áreas e subtópicos preferidos. Depois, sempre que forem publicados conteúdos nessas áreas e subtópicos, o utilizador deve ser notificado
48	Possibilidade de alterar o estado dos formulários antes de eles serem publicados por um Administrador
Bónus	
49	A plataforma deve ter 3 línguas disponíveis (Português, Inglês e Espanhol);
50	Apresentar uma saudação ao utilizador conforme a hora

51	Disponibilizar área de Informações/Avisos: A página “Informações/Avisos” é a página onde os administradores podem criar, consultar e editar informações genéricas e avisos, que são disponibilizados imediatamente na Plataforma Web e App mobile na página “Informações/Avisos”
----	--

3. Desenvolvimento

Neste capítulo vão ser explicadas detalhadamente as aplicações mobile e Web e a infraestrutura que está por trás, Base de dados e API. Para a entrega do projeto utilizamos a plataforma render.com para hospedar as diferentes aplicações (Web, API e Base de Dados). Também com o intuito de facilitar a colaboração em equipa utilizamos o Github.

3.1. Base de Dados

3.1.1. Estrutura da Base de Dados

A base de dados foi desenhada para suportar de forma eficiente as principais funcionalidades da aplicação. Abaixo, apresento uma visão geral das principais tabelas e dos seus relacionamentos:

- **Perfil, Centro e Utilizador:** Estas tabelas armazenam as informações básicas sobre os perfis dos utilizadores, os centros aos quais podem estar associados e os próprios utilizadores. As relações entre estas tabelas permitem definir e gerir os diferentes tipos de utilizadores na aplicação.
- **Tópicos, Subtópicos e Interesses:** Estas tabelas organizam as áreas temáticas e os interesses dos utilizadores, permitindo uma personalização e filtragem de conteúdos relevantes para cada perfil.
- **Conteúdo:** Tabelas como conteúdo, álbum, comentário, e classificação armazenam e gerem todo o conteúdo gerado pelos utilizadores, incluindo uploads de imagens, comentários e avaliações.
- **Moderação e Gestão de Estado:** As tabelas estado, revisão, e denuncia são utilizadas para gerir o fluxo de trabalho de moderação, revisões de conteúdo, notificações enviadas aos utilizadores e o tratamento de denúncias.

3.1.2. PgAdmin

A implementação da base de dados foi realizada no pgAdmin, uma ferramenta robusta para administração de bases de dados PostgreSQL. As operações de criação, modificação e gestão das tabelas foram executadas através de scripts SQL, que permitem a criação de uma estrutura eficiente e escalável.

3.2. API

3.2.1. Tecnologias Utilizadas

3.2.1.1 Node.js

Node.js é uma plataforma de desenvolvimento backend baseada no motor V8 do Google Chrome, que permite a execução de JavaScript no lado do servidor. É uma tecnologia fundamental para a API, oferecendo um ambiente eficiente e escalável para o desenvolvimento de aplicações web. Node.js permite a construção de APIs que lidam com um grande número de conexões simultâneas, graças ao seu modelo assíncrono baseado em eventos.

3.2.1.2 Express.js

Express.js é um framework minimalista e flexível para Node.js, que facilita a criação de aplicações web robustas e APIs RESTful. Ele fornece uma estrutura sólida para lidar com requisições HTTP, roteamento, middlewares, e muito mais. Na tua API, o Express.js é utilizado para definir rotas, gerir middlewares (como autenticação e manipulação de erros), e estruturar a lógica de negócio de forma modular e eficiente.

3.2.1.3 Sequelize

Sequelize é um ORM (Object-Relational Mapping) para Node.js, que simplifica a interação com bases de dados relacionais, como PostgreSQL, MySQL, MariaDB, SQLite e MSSQL. Ele permite mapear modelos JavaScript a tabelas de bases de dados, facilitando operações como consultas, inserções, atualizações e exclusões de dados, tudo isso utilizando uma sintaxe simples e intuitiva. Na API, o Sequelize é utilizado para abstrair a camada de acesso a dados, permitindo que desenvolvedores trabalhem com objetos JavaScript ao invés de escrever diretamente SQL, o que acelera o desenvolvimento e reduz a probabilidade de erros.

3.2.1.4 JWT (JSON Web Token)

JSON Web Token (JWT) é uma tecnologia de autenticação que permite a comunicação segura entre cliente e servidor. É amplamente utilizada para gerir sessões de utilizadores, mantendo a segurança sem a necessidade de armazenar informações sensíveis no servidor. Na tua API, JWT é utilizado para gerar tokens que são enviados ao cliente após a autenticação bem-sucedida, permitindo que o cliente acesse rotas protegidas até que o token expire.

3.2.1.5 bcrypt

bcrypt é uma biblioteca de hashing que proporciona uma forma segura de encriptar senhas antes de armazená-las na base de dados. Ela aplica um algoritmo de hashing que é lento e resistente a ataques de força bruta, aumentando a segurança das credenciais dos utilizadores. Na API, o bcrypt é utilizado para encriptar senhas no momento da criação de utilizadores e para compará-las durante o processo de login.

3.2.1.6 Cloudinary

Cloudinary é uma solução de gestão de imagens e vídeos na nuvem, que facilita o upload, armazenamento e manipulação de arquivos multimédia. Na API, o Cloudinary é integrado para lidar com o upload e a gestão de imagens de forma eficiente, permitindo que estas sejam armazenadas na nuvem e servidas de forma otimizada para os clientes.

3.2.1.7 Crypto

Crypto é um módulo nativo do Node.js que fornece funcionalidades de criptografia e hashing. Na API, é utilizado para operações como geração de hashes, criação de tokens de segurança, e encriptação de dados sensíveis. Isso garante uma camada extra de segurança, protegendo informações críticas contra acessos não autorizados.

3.2.1.8 google-auth-library

google-auth-library é uma biblioteca utilizada para integrar autenticação e autorização via OAuth 2.0 e JWT com os serviços da Google. Na API, pode ser empregada para permitir que os utilizadores façam login utilizando as suas credenciais do Google, facilitando a autenticação através de um provedor de identidade confiável e amplamente utilizado.

3.2.1.9 multer

Multer é uma biblioteca de middleware para Express.js que facilita o tratamento de uploads de arquivos. É especialmente útil para manipular uploads de imagens, documentos e outros tipos de arquivos na API. O Multer permite definir a forma como os arquivos são armazenados e geridos, seja na memória ou diretamente em disco, garantindo uma integração eficiente com serviços de armazenamento como o Cloudinary.

3.2.1.10 nodemailer

Nodemailer é uma biblioteca para o envio de emails a partir de aplicações Node.js. Na API, é utilizado para enviar emails automatizados, como confirmações de registo, notificações, ou recuperação de senhas. O Nodemailer facilita a integração com diferentes serviços de email, permitindo que a API envie mensagens de forma confiável e segura.

3.2.1.11 pg e pgAdmin

pg é um cliente para Node.js que permite a interação direta com bases de dados PostgreSQL. Na API, o pg é utilizado para executar consultas SQL quando se prefere um acesso mais direto ou personalizado à base de dados, sem a necessidade de abstrações oferecidas por um ORM

como o Sequelize. Isso pode ser útil para operações específicas ou quando o desempenho é uma prioridade.

3.2.2. Arquitetura da Aplicação

3.2.2.1 Controllers

Os Controllers são uma parte fundamental na arquitetura de uma aplicação, pois atuam como intermediários entre a lógica de negócios e as requisições feitas pelos utilizadores. São responsáveis por processar as entradas, interagir com os serviços de negócio e devolver as respostas apropriadas aos clientes. Este capítulo descreve a implementação dos Controllers na nossa aplicação, com foco nas classes *Controller* e *BaseController*, que fornecem a estrutura necessária para criar controladores específicos para diferentes modelos de dados.

1. Estrutura Geral dos Controllers

Os Controllers foram implementados de forma a serem reutilizáveis e modulares, permitindo uma fácil extensão e manutenção. A estrutura básica de um *Controller* é definida na classe *Controller*, que serve como uma classe base, enquanto a classe *BaseController* fornece implementações concretas de métodos comuns, como operações *CRUD* (*Create*, *Read*, *Update*, *Delete*).

2. A Classe *Controller*

A classe *Controller* é a base sobre a qual todos os outros controladores são construídos. Ela encapsula a lógica comum de inicialização dos controladores, como a associação de um modelo de dados específico e a configuração de um serviço básico para operar com esse modelo. Cada controlador específico herda desta classe, garantindo que as operações básicas e o registo de logs sejam realizados de forma consistente em toda a aplicação.

Estrutura da Classe *Controller*

```
export class Controller {  
  constructor(model, identifier = Constants.DEFAULT_IDENTIFIER) {  
    this.model = model;  
    this.identifier = identifier;  
    this.service = new BaseService(model, identifier);  
    LogUtils.log(model.name, LogUtils.TIPO.CONTROLLERS);  
  }  
}
```

Figura 1 - Classe *Controller*

- Inicializa a classe com um *model* e um *identifier*, que são usados para definir o contexto de operação do controlador.
- Um serviço (*BaseService*) é instanciado para gerir as operações relacionadas ao modelo.
- A classe também regista informações de log para monitorizar a operação do controlador.

3. A Classe *BaseController*

A classe *BaseController* estende a classe *Controller* e implementa métodos pré-definidos para as operações *CRUD*. Esta classe foi projetada para ser uma solução genérica que pode ser facilmente adaptada para diferentes modelos, fornecendo uma implementação padrão para as operações mais comuns.

Estrutura da Classe *BaseController*

```
export class BaseController extends Controller {
  constructor(model, identifier = Constants.DEFAULT_IDENTIFIER) {
    super(model, identifier);
  }

  async obter(req, res) {
    try {
      const { id } = req.params;
      const response = await this.service.obter(id);
      return ResponseService.success(res, response);
    } catch (error) {
      return ResponseService.error(res, error.message);
    }
  }

  //...
}
```

Figura 2 - Classe *BaseController*

Métodos CRUD:

- obter: Busca um registo específico com base no identificador fornecido.
- simples_obter, criar, listar, atualizar, remover: Métodos adicionais que cobrem as operações básicas de uma API RESTful.
- Cada método segue uma estrutura padrão: tenta realizar a operação com a ajuda de um serviço e retorna uma resposta apropriada ao cliente.

3.2.2.2 Models

3.2.2.3 Routes

3.2.2.4 Services

3.2.2.5 Middleware

3.2.3. Estrutura

A API está estruturada da seguinte forma:

- **config:** Este diretório geralmente contém ficheiros de configuração, como variáveis de ambiente, configuração de base de dados e outras definições globais que a aplicação possa necessitar. Essas configurações são utilizadas por outras partes da aplicação para garantir que as mesmas informações e parâmetros sejam usados de forma consistente.
- **constants:** O diretório constants armazena constantes que são usadas em toda a aplicação, como códigos de erro, mensagens padrão, e quaisquer outros valores imutáveis que precisam estar disponíveis globalmente.
- **controllers:** Em controllers estão as funções responsáveis por manipular as requisições e respostas HTTP. Eles fazem a ponte entre as rotas e a lógica de negócio. Um controller normalmente chama métodos de serviços para realizar operações e devolver os resultados ao cliente.
- **exceptions:** Este diretório é destinado ao tratamento de exceções e erros personalizados. Aqui podem estar definidos tipos específicos de erros que são lançados e manipulados em diferentes partes da aplicação.

- **helpers:** helpers contém funções utilitárias que ajudam a realizar tarefas repetitivas ou comuns, como manipulação de datas, formatação de dados, entre outros. Essas funções são reutilizáveis e podem ser chamadas a partir de qualquer parte do código.
- **middlewares:** Em middlewares, encontram-se as funções que interceptam as requisições antes que elas cheguem aos controllers. Elas podem realizar tarefas como autenticação, validação de dados, logging, etc. Os middlewares são uma parte fundamental do fluxo de requisições em aplicações Express.
- **models:** O diretório models armazena as definições das entidades da aplicação, normalmente mapeando para tabelas de base de dados. Estes modelos representam a estrutura dos dados e contêm métodos para realizar operações de base de dados, como criar, ler, atualizar e eliminar registros.
- **routes:** Em routes estão definidos os endpoints da API. Cada rota está associada a um controller que executa a lógica necessária para responder às requisições. As rotas organizam como as requisições HTTP são tratadas, direcionando-as para os controllers adequados.
- **services:** O diretório services contém a lógica de negócio da aplicação. Enquanto os controllers lidam com a interface HTTP, os serviços realizam operações mais complexas que podem envolver múltiplos modelos ou interações externas, como chamadas a APIs de terceiros.
- **utils:** utils é semelhante a helpers, mas costuma armazenar funções utilitárias mais genéricas que não estão diretamente ligadas à lógica de negócio. Pode incluir, por exemplo, funções de manipulação de ficheiros, criptografia, etc.
- **app.js:** Este é geralmente o ponto de entrada da aplicação. O ficheiro app.js configura o servidor, carrega os middlewares globais, define as rotas principais e inicia a aplicação. É aqui que o servidor Express é configurado e inicializado.

3.3. Web

3.3.1. Estrutura do Projeto

A estrutura do projeto esta separada da seguinte forma:

- **api:** Neste diretório estão centralizadas todas as funções e serviços relacionados com a comunicação com a API.
- **assets:** O diretório assets é utilizado para armazenar recursos estáticos como imagens, ícones.
- **components:** O diretório components é onde estão armazenados os diversos componentes que são utilizados na aplicação, como por exemplo, botões, formulários, cartões entre outros.
- **context:** O uso do context sugere que a tua aplicação utiliza o React Context API para gerir o estado global ou partilhado entre diferentes partes da aplicação. Este é um bom padrão para evitar o “prop drilling” (passar props de um componente para outro de forma exagerada) e para centralizar a gestão do estado que é partilhado por múltiplos componentes.
- **data:** O diretório data provavelmente contém ficheiros JSON ou funções que gerem dados estáticos ou dinâmicos. Este pode ser um bom lugar para armazenar dados simulados (mock data) durante o desenvolvimento ou para armazenar constantes e configurações.
- **hooks:** Este diretório sugere que tens hooks personalizados (custom hooks) na tua aplicação. Hooks personalizados permitem encapsular e reutilizar a lógica de estado e efeitos que podem ser partilhados entre diferentes componentes, seguindo o princípio DRY (Don't Repeat Yourself).
- **layouts:** Os layouts são usados para definir a estrutura geral das páginas ou secções da aplicação. Podem incluir cabeçalhos, rodapés, barras laterais e áreas de conteúdo. Este padrão é útil para garantir uma consistência visual e estrutural na aplicação.
- **pages:** O diretório pages indica que a aplicação está dividida em várias páginas ou vistas. É comum em aplicações React organizadas de forma SPA (Single Page Application), onde cada componente dentro de pages representa uma rota ou secção principal da aplicação.

- **utils:** Funções utilitárias ou helpers que são usadas em múltiplos lugares da aplicação devem estar neste diretório. Isso ajuda a evitar a repetição de código e facilita a manutenção e testes dessas funções.

3.3.2. Estrutura da Aplicação

3.3.3. Rotas

3.3.4. Componentes

3.3.5. Layout

3.3.6. Comunicação com a API

3.4. Mobile

3.4.1. Estrutura do Projeto

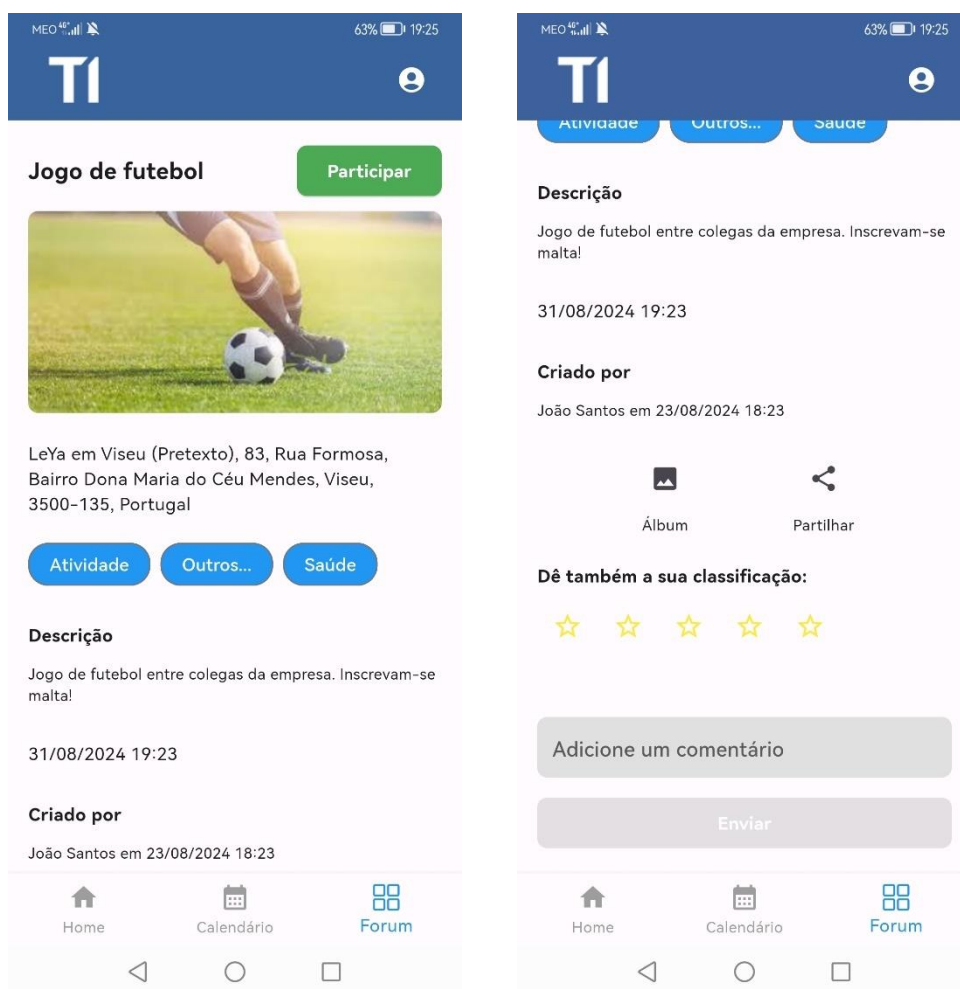
O projeto foi organizado em diferentes pastas para garantir uma estrutura organizada, que facilitasse o desenvolvimento contínuo pelos diferentes membros do grupo e a compreensão do código em melhorias futuras.

A estrutura está separada da seguinte forma:

- **assets** – este diretório foi usado para guardar imagens e logos.
- **backend** – esta pasta contém todos os ficheiros relacionados a funções de comunicação com a API.
- **widgets** – aqui encontram-se os ficheiros com o código dos widgets que foram sendo reutilizados em diferentes partes do projeto.
- **utils** – contém todas as funções uteis que são usadas em vários pontos da aplicação o que evita a repetição de código.
- **lib** – esta pasta está subdividida em vários diretórios. Cada um contém ficheiros correspondentes a funcionalidades, páginas ou conteúdos diferentes como por exemplo: atividade, espaço, recomendação, evento, calendário e perfil.

3.4.2. Detalhes dos conteúdos

A página dos detalhes dos conteúdos apresenta a seguinte estrutura base:

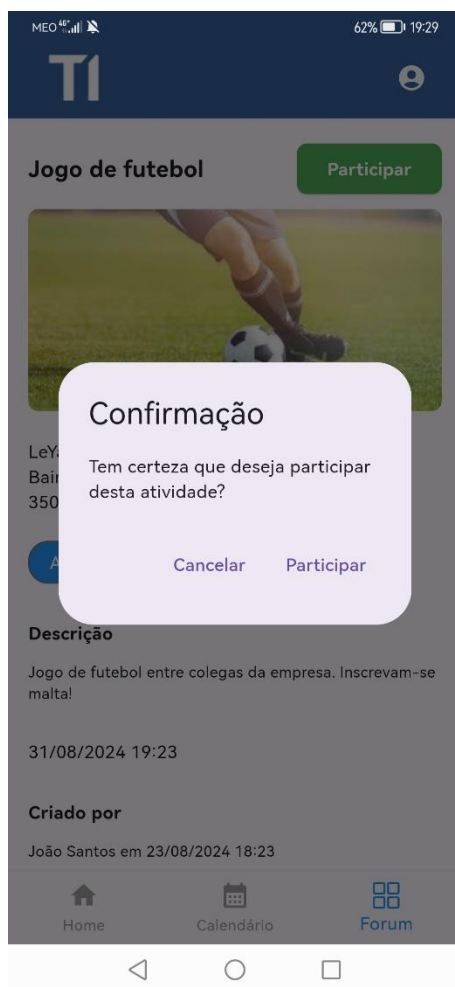


Nesta página são mostrados os detalhes de um conteúdo: título, imagem, local, tópico e subtópicos a que pertencem, descrição, data/hora, o utilizador que criou bem como a data e hora de criação. Tem também um botão de “participar”, um álbum de imagens, uma opção de partilha, classificação e uma secção de comentários. Estas funcionalidades são explicadas a seguir.

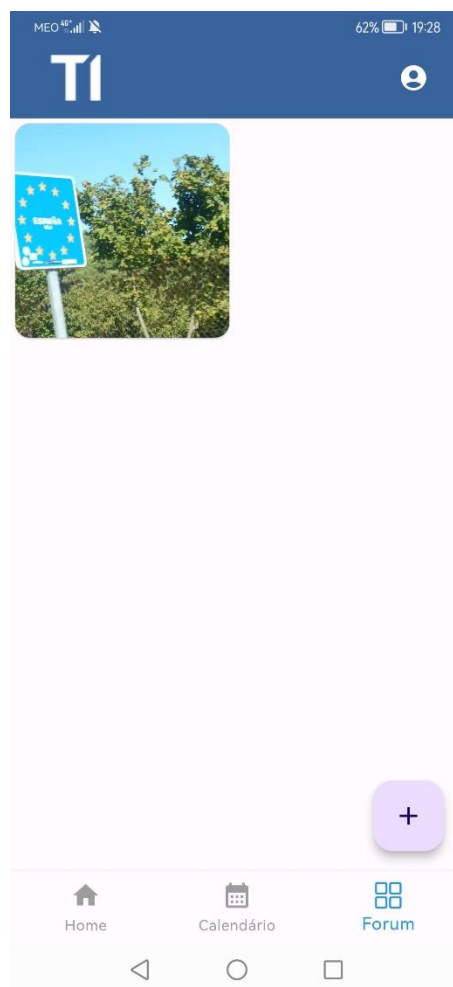
É de notar que de acordo com o tipo de conteúdo esta página será ligeiramente diferente, sendo mostrados ou escondidos campos como data/hora, preço.

Funcionalidades/Requisitos

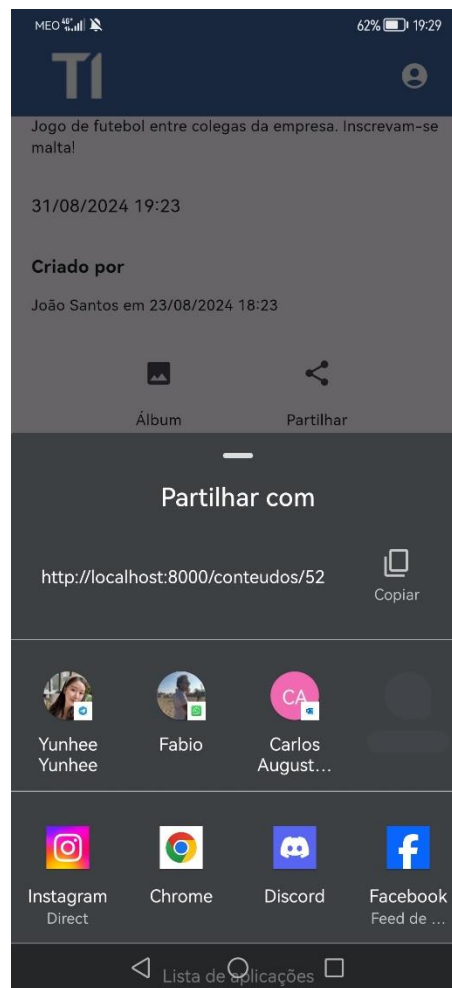
- **Participar na atividade** – Ao carregar no botão de “Participar” no topo da página o utilizador cria uma participação no conteúdo em questão. Este botão estará oculto para o caso de o conteúdo ser uma recomendação ou espaço.



- **Álbum de imagens** – Se o utilizador clicar no botão do “Álbum” será redirecionado para outra página onde poderá ver e adicionar mais imagens ao álbum.



- **Partilhar** – Por sua vez o botão de “Partilhar” copia para a área de transferência o link do conteúdo da página web. Ao mesmo tempo aparecem as aplicações do telemóvel onde o utilizador poderá partilhar esse link.

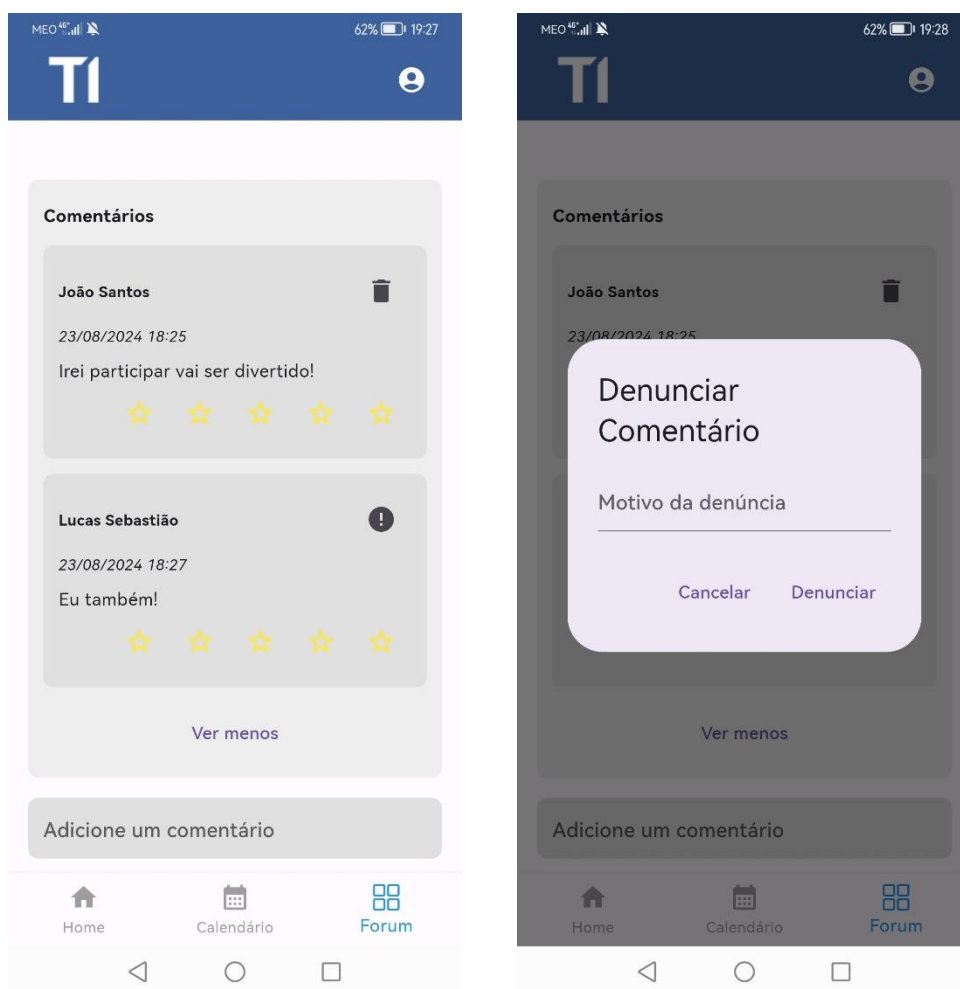


- **Classificação** – Tal como pretendido também é possível para um utilizador atribuir uma classificação de 1-5 (representado por estrelas) ao conteúdo.



Comentários – Existe também uma secção de comentários onde os utilizadores podem interagir acerca do conteúdo publicado.

O utilizador consegue adicionar comentários ao conteúdo e eliminá-los caso estes não tenham nenhuma denuncia ou classificação. Pode ainda denunciar e classificar comentários de outros users.



3.4.3. Criar conteúdos

Voltando agora para a página do “fórum”, temos um botão no canto superior direito que diz “Adicionar”. A partir dele, podemos criar um dos tipos de conteúdo que queremos, como “atividade”, “evento”, “recomendação” e “espaço”.

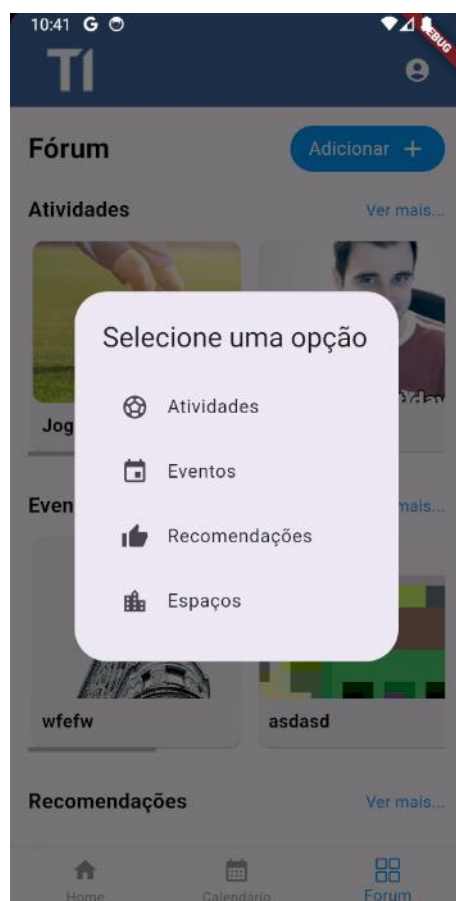


Figura 3 Criar conteúdo

- **Atividade**

Na figura ao lado (Figura 3), podemos ver as diferentes opções que podemos escolher. Primeiramente, se escolhermos “Atividade”, seremos redirecionados para a página de criação de atividade (Figura 4), onde temos várias opções para personalizar a atividade.

Primeiro, selecionamos uma imagem, o que abrirá a galeria do telemóvel e em seguida, escrevemos um título para a atividade. Depois, definimos o local que ao clicar no ícone, um mini mapa é aberto, permitindo que selecionemos qualquer ponto no mundo, para facilitar, o local e a morada selecionados aparecem diretamente acima do botão “Confirmar” (Figura 4).

Após selecionar o local, adicionamos a descrição e um subtópico, que pode ser escolhido a partir de uma lista de subtópicos existentes e em seguida, selecionamos o dia e a hora em que queremos realizar a atividade.

No final, há dois botões: um para cancelar, que exibe um aviso perguntando se realmente deseja cancelar ou continuar (Figura 5), e um botão “Criar Atividade”, que, como o nome sugere, finaliza a criação da atividade.

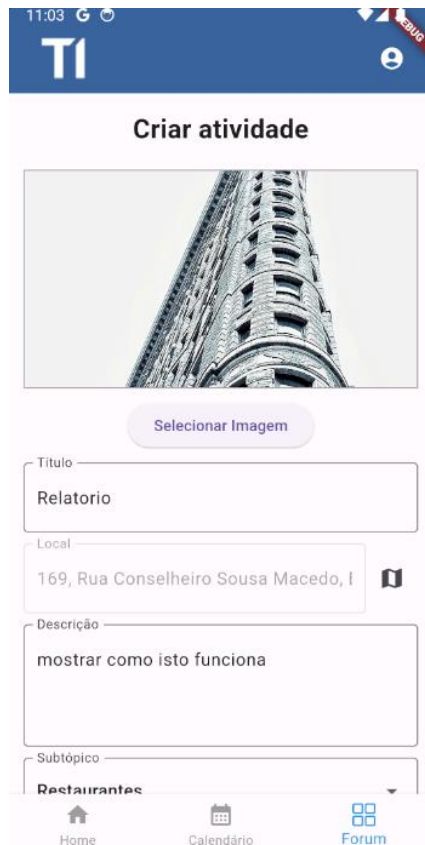


Figura 4 Criar atividade

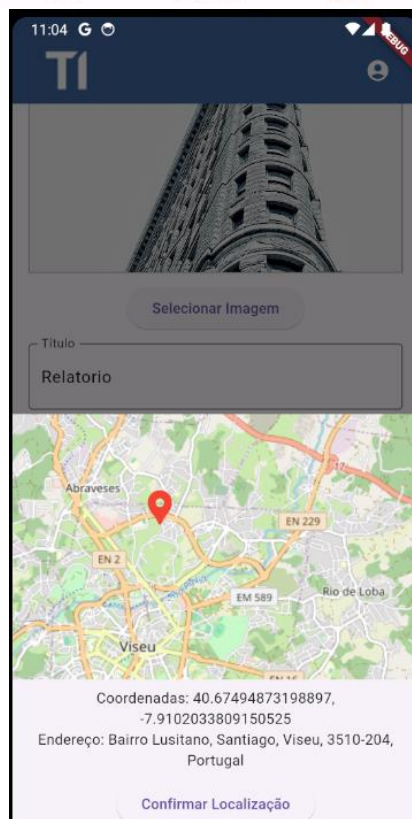


Figura 5 Mapa

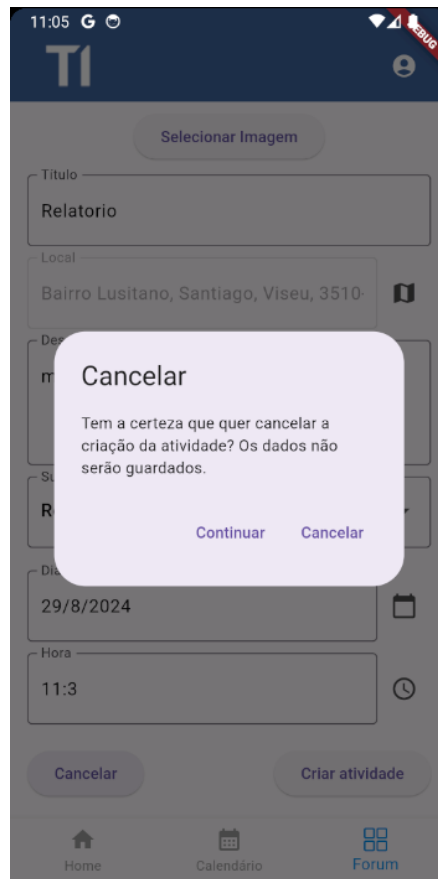


Figura 6 Botão cancelar

- **Evento**

Agora vamos para a opção evento que nos leva para a página de criar evento (Figura 7), aqui as opções para criar são exatamente iguais como na atividade. Aproveito para referir que todas as opções são necessárias preencher caso uma delas seja deixada, ao clicar em “criar evento” este não será continuado e aparecerá um aviso a vermelho sobre qual foi deixado. (figura 8).

The screenshot shows the 'Criar Novo Evento' screen in the T1 mobile application. At the top, there is a blue header with the 'T1' logo and a user profile icon. Below the header, the title 'Criar Novo Evento' is centered. A button labeled 'Selecionar Imagem' is positioned above a text input field for 'Título'. Below this is another text input field for 'Local', accompanied by a location pin icon. A larger text area for 'Descrição' follows. Below the description is a dropdown menu for 'Subtópico'. Further down are two more input fields: 'Dia' with a calendar icon and 'Hora' with a clock icon. At the bottom, there is a navigation bar with three icons: 'Home', 'Calendário', and 'Forum'.

Figura 7 Criar evento

This screenshot shows the same 'Criar Novo Evento' form as Figure 7, but with validation errors indicated by red borders and red text. The errors are as follows: 'Por favor, insira o nome do evento' above the 'Título' field; 'Por favor, insira o local da recomendação' above the 'Local' field; 'Por favor, selecione um subtópico' below the 'Subtópico' dropdown; 'Por favor, selecione o dia do evento' below the 'Dia' field; and 'Por favor, selecione a hora do evento' below the 'Hora' field. At the bottom, there are two buttons: 'Cancelar' and 'Criar evento'. The navigation bar at the very bottom remains the same with 'Home', 'Calendário', and 'Forum' icons.

Figura 8 Tópicos em falta

- **Recomendações**

Agora na opção recomendação as opções são muito parecidas com as outras exceto que agora temos uma opção em que o criador tem de por o preço nessa opção apenas aceita números e não letras (figura 9) e por baixo temos 5 estrelas em que se pode escolher quantas estrelas queremos dar a recomendação que estamos a fazer sendo que o mais baixo que se pode dar é 0 estrelas e o mais alto é 5 estrelas (Figura 10).



The screenshot shows a mobile application interface for creating a recommendation. The form is titled 'TI' and has a blue header. The fields are as follows:

- Título:** Relatorio
- Local:** Vildemoinhos, Repeses e São Salvado
- Descrição:** Teste para ele
- Subtópico:** Infantários
- Preço:** 12 (This field is highlighted with a purple border, indicating it is the focus of the current screen)
- Classificação:** 5 stars (indicated by five gray stars)

A numeric keypad is displayed at the bottom of the screen, with buttons for digits 1-9, 0, a decimal point, a minus sign, a backspace key, and a checkmark key.

Figura 9 Preço

The screenshot shows a mobile application interface for creating a recommendation. At the top, there is a status bar with the time 1:42, signal strength, and a battery icon. Below the status bar is a blue header with the logo 'T1' and a user profile icon. The main form is white and contains the following elements:

- A button labeled 'Selecionar Imagem' (Select Image) at the top.
- A text input field for 'Título' (Title) with the value 'Relatorio'.
- A text input field for 'Local' (Location) with the value 'Vildemoinhos, Repeses e São Salvado' and a location pin icon.
- A text input field for 'Descrição' (Description) with the value 'Teste para ele'.
- A dropdown menu for 'Subtópico' (Subtopic) with the value 'Infantários'.
- A text input field for 'Preço' (Price) with the value '12'.
- A star rating system labeled 'Classificação' (Rating) with five stars, four of which are yellow and one is gray.
- Two buttons at the bottom: 'Cancelar' (Cancel) and 'Criar recomendação' (Create recommendation).

At the very bottom, there is a navigation bar with three icons: a house icon for 'Home', a calendar icon for 'Calendário' (Calendar), and a grid icon for 'Forum'.

Figura 10 Estrelas

- **Espaço**

Em criar espaço as opções mais uma vez são iguais como nos outros criar, sendo que agora podemos por uma imagem, um título, o local, Descrição e um subtópico (figura 11).

The screenshot shows a mobile application interface for creating a new space. At the top, there is a status bar with the time 2:02, signal strength, and battery level. Below the status bar is a blue header with the logo 'TI' and a user profile icon. The main title is 'Criar Novo espaço'. Below the title is a button labeled 'Selecionar Imagem'. There are three text input fields: 'Título', 'Local', and 'Descrição'. To the right of the 'Local' field is a location pin icon. Below the 'Descrição' field is a dropdown menu labeled 'Subtópico' with the selected option 'Centros de Formação'. At the bottom of the form are two buttons: 'Cancelar' and 'Criar espaço'. The bottom navigation bar has three icons: 'Home', 'Calendário', and 'Forum'.

Figura 11 Criar Espaço

3.4.4. Perfil

O perfil está localizado no canto superior direito em todas as páginas (figura 12), ao clicar, este leva-nos para a página do perfil onde podemos encontrar a foto de perfil o nosso nome, a tag, o email, LinkedIn, facebook, Instagram, centro e o tipo de perfil (administrador ou utilizador) (figura 13). Em baixo ainda temos uma lista em que podemos escolher os nossos interesses que serão mostrados aos outros utilizadores (figura 14). No fundo temos 4 botões



Figura 12 Icon do perfil

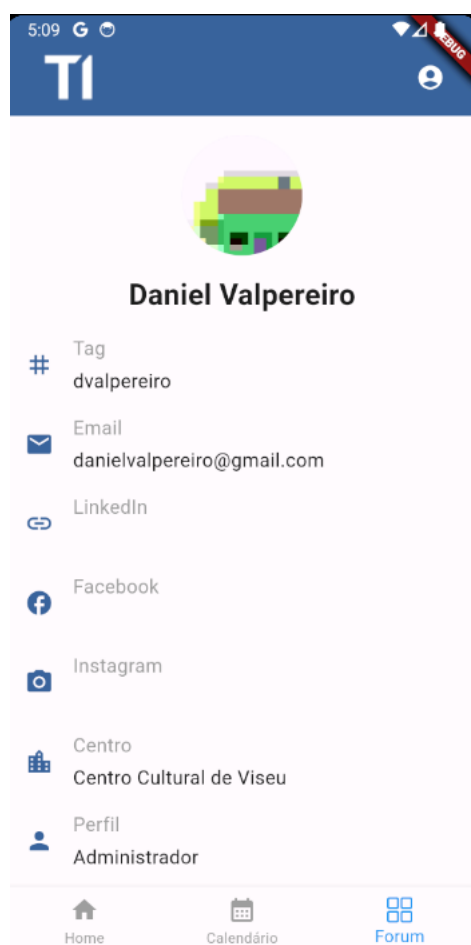


Figura 13 Página de perfil

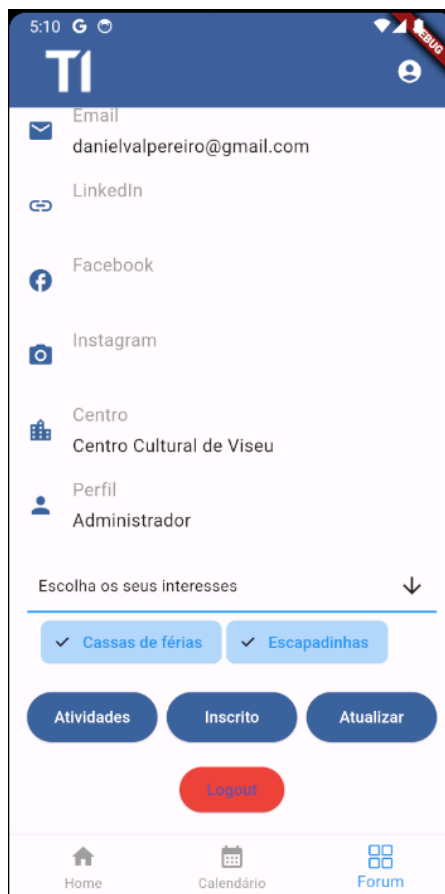


Figura 14 Interesses e botões

No fundo temos 4 botões (Figura 14) um para as atividades que nos mostra todas as atividades que nos criamos (figura 15), um para ver todos os conteúdos a quais estamos inscritos (figura 16) ao lado ainda temos um botão para atualizarmos o perfil, ou seja para mudar qualquer informação sobre ele (3.4.5 Editar perfil) e finalmente o botão a vermelho é utilizado para sair da conta e voltar para a pagina do login, antes disso ainda tem um aviso caso tenha carregado por erro (Figura 17).

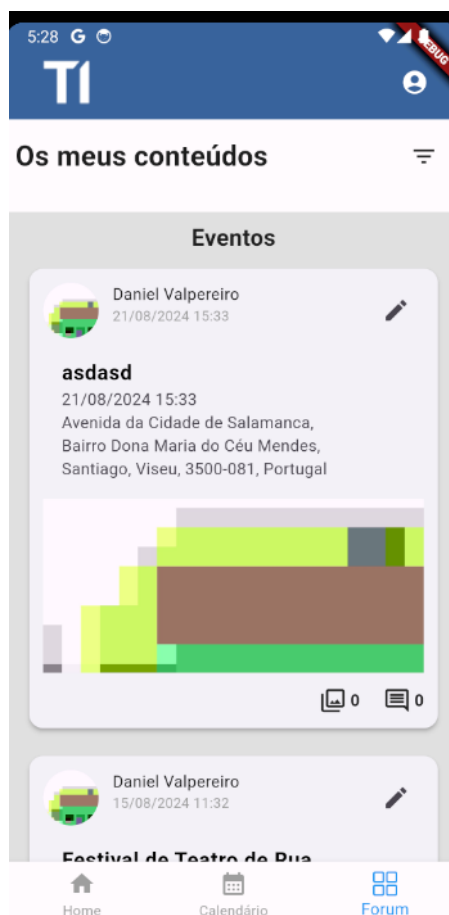


Figura 15 Conteúdos criados

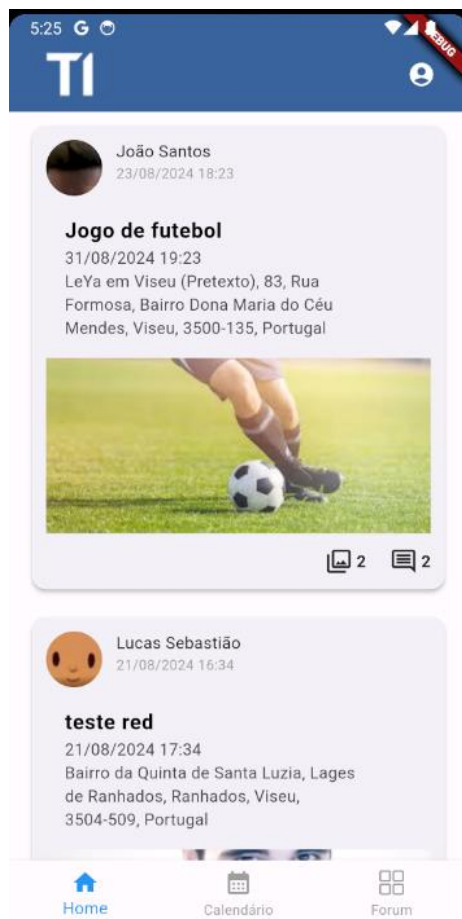


Figura 16 Inscritos

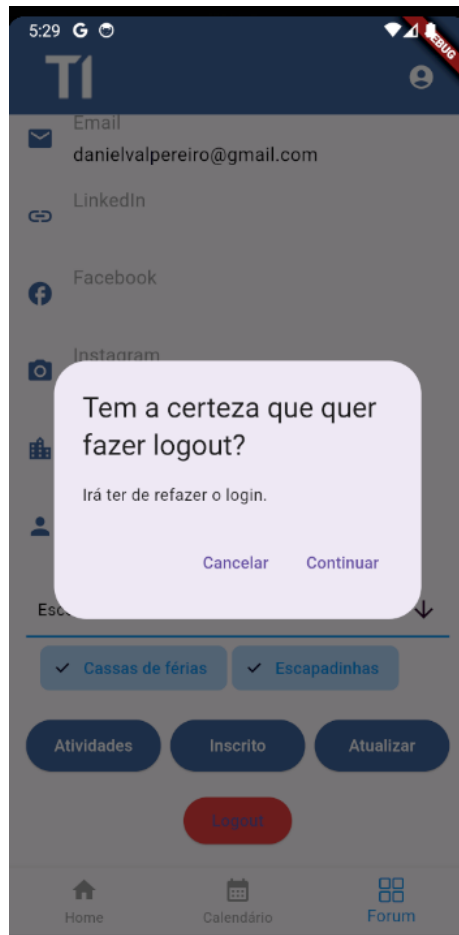
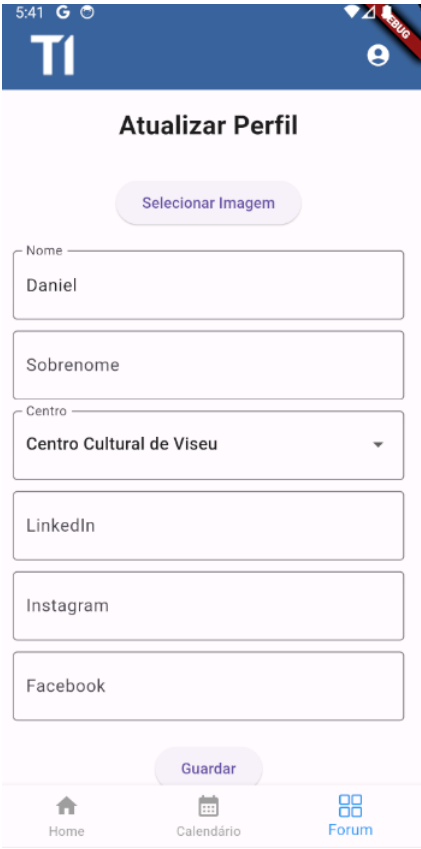


Figura 17 Aviso ao fazer logout

3.4.5. Editar perfil

Para aceder ao editar perfil basta clicar no botão de “atualizar” como vimos antes aqui podemos modificar tudo sobre o perfil, ou seja, imagem, nome, sobrenome, centro, linkedin, instagram e facebook, no final podemos simplesmente clicar em “guardar” (figura 18), importa referir que não é obrigatório escrever o linkedin, facebook ou instagram, mas, no entanto, é obrigatório ter imagem, nome e centro.



The screenshot shows a mobile application interface for updating a profile. At the top, there is a status bar with the time 5:41, signal strength, and battery level. Below the status bar is a blue header with the logo 'T1' and a user profile icon. The main title is 'Atualizar Perfil'. Below the title is a button labeled 'Selecionar Imagem'. The form contains several input fields: 'Nome' with the value 'Daniel', 'Sobrenome', 'Centro' with a dropdown menu showing 'Centro Cultural de Viseu', 'LinkedIn', 'Instagram', and 'Facebook'. At the bottom of the form is a button labeled 'Guardar'. The bottom navigation bar has three icons: 'Home', 'Calendário', and 'Forum'.

Figura 18 Editar perfil

3.4.6. Calendário (Francisco)

3.4.7. Fórum (Francisco)

3.4.8. Listagem por tipo (Francisco)

3.4.9. Integração com o Google e Facebook (Cassamo/Francisco)

3.4.10. Iniciar sessão (Cassamo)

Falar da comunicação com a api (não é preciso explicar o que acontece na api) como é feita (aproveitando e mencionando o capítulo comunicação com a api) e mencionar também o esqueceu passe

3.4.11. Comunicação com a API (Cassamo)

Explicar para que serve e como funcionam as classes AuthService, ApiService e MyHttp

4. Conclusão