A picture containing text

Description automatically generated



INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR

DE ENGENHARIA

DE COIMBRA

DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

**Sistema de Informação para Gestão de Parcerias de Estágio e Projetos do ISEC**

Projeto Final de Licenciatura em Engenharia Informática

Autor

**Bruno Gabriel Tavares Pinto**

Orientador

**César Paulo das Dores Páris**

Coimbra, julho e 2025

Índice

[Índice i](#_Toc207822581)

[Índice de figuras iii](#_Toc207822582)

[Lista de siglas e acrónimos iv](#_Toc207822583)

[Agradecimentos 1](#_Toc207822584)

[Resumo 2](#_Toc207822585)

[1 Introdução 3](#_Toc207822586)

[1.1 Estrutura do relatório 3](#_Toc207822587)

[2 Estado Pré-existente do Projeto 5](#_Toc207822588)

[2.1 Arquitetura e tecnologias 5](#_Toc207822589)

[3 Planeamento e Conceção do Sistema 7](#_Toc207822590)

[3.1 Objetivos 7](#_Toc207822591)

[3.2 Arquitetura do sistema 11](#_Toc207822592)

[3.3 Tecnologias usadas 12](#_Toc207822593)

[4 Implementação da API 13](#_Toc207822594)

[4.1 Modelo de Dados 13](#_Toc207822595)

[4.2 Implementação da API 15](#_Toc207822596)

[4.3 Organização do projeto 15](#_Toc207822597)

[4.4 Funcionamento da API 16](#_Toc207822598)

[4.5 Geração Automática de Documentos 16](#_Toc207822599)

[4.6 Tarefas assíncronas 17](#_Toc207822600)

[5 Interface e Design visual 19](#_Toc207822601)

[5.1 Conceção visual do sistema 19](#_Toc207822602)

[5.1.1 Cores 19](#_Toc207822603)

[5.1.2 Tipografia 20](#_Toc207822604)

[5.1.3 Espaçamentos 20](#_Toc207822605)

[5.1.4 Componentes 20](#_Toc207822606)

[5.2 *Design* da Aplicação 21](#_Toc207822607)

[5.3 Implementação da Interface 21](#_Toc207822608)

[5.3.1 Organização do projeto 21](#_Toc207822609)

[5.4 Arquitetura da Interface 22](#_Toc207822610)

[6 Interação com o Sistema 25](#_Toc207822611)

[6.1 Autenticação de um utilizador (*login*) 25](#_Toc207822612)

[6.2 Filtragem de Listas 27](#_Toc207822613)

[6.3 Gestão de Docentes 27](#_Toc207822614)

[6.4 Gestão de Calendários 30](#_Toc207822615)

[6.5 Submissão de uma Proposta 32](#_Toc207822616)

[6.6 Exportação da Proposta em PDF 35](#_Toc207822617)

[7 Desenvolvimento Futuro 37](#_Toc207822618)

[8 Conclusão 40](#_Toc207822619)

[Anexos I](#_Toc207822620)

Índice de figuras

[Figura 1 – Listagem de Funcionalidades 8](#_Toc207822621)

[Figura 2 – Arquitetura do Sistema 11](#_Toc207822622)

[Figura 3 – Diagrama Entidade-Relação 13](#_Toc207822623)

[Figura 4 – Classe *Accounts* 14](#_Toc207822624)

[Figura 5 – Classe Aluno 14](#_Toc207822625)

[Figura 6 – Estrutura padrão dos endpoints da API 16](#_Toc207822626)

[Figura 7 - Agendamento da tarefa principal 17](#_Toc207822627)

[Figura 8 - Tarefa assíncrona 18](#_Toc207822628)

[Figura 9 – Representação de uma cor com as suas nove variações 19](#_Toc207822629)

[Figura 10 – Página de *Login* 25](#_Toc207822630)

[Figura 11 – Página inicial com as permissões padrão 26](#_Toc207822631)

[Figura 12 – Página inicial com todas as permissões 26](#_Toc207822632)

[Figura 13 – Filtragem de listas 27](#_Toc207822633)

[Figura 14 – Listagem de docentes 28](#_Toc207822634)

[Figura 15 – Visualização de um docente 28](#_Toc207822635)

[Figura 16 – Edição de um docente 29](#_Toc207822636)

[Figura 17 – Listagem de com um docente removido 30](#_Toc207822637)

[Figura 18 – Listagem de Calendários 30](#_Toc207822638)

[Figura 19 – Formulário de criação de calendário 31](#_Toc207822639)

[Figura 20 – Visualização de um Calendário 32](#_Toc207822640)

[Figura 21 – Mensagem de erro do formulário 33](#_Toc207822641)

[Figura 22 – Formulário de submissão de uma Proposta 33](#_Toc207822642)

[Figura 23 – Seleção de curso, ramos e calendário no formulário de submissão 34](#_Toc207822643)

[Figura 24 – Criação de um Orientador no formulário de submissão 34](#_Toc207822644)

[Figura 25 - Botão para geração do PDF 35](#_Toc207822645)

[Figura 26 - Documento PDF da proposta gerado 36](#_Toc207822646)

[Figura 27 - Estado final das funcionalidades 37](#_Toc207822647)

Lista de siglas e acrónimos

|  |  |
| --- | --- |
| API | *Application Programming Interface* |
| CSS | *Cascading Style Sheets* |
| HTML | *HyperText Markup Language* |
| HTTP | *Hypertext Transfer Protocol* |
| IDE | *Integrated Development Environmen* |
| ISEC | Instituto Superior de Engenharia de Coimbra |
| JSON | *JavaScript Object Notation* |
| JWT | JSON *Web Token* |
| MVT | *Model-View-Template* |
| ORM | *Object Relational Mapping* |
| PDF | *Portable Document Format* |
| SQL | *Structured Query Language* |
| UI | *User Interface* |
| VSCode | Visual Studio Code |

Agradecimentos

Agradeço ao Professor César Páris, orientador do projeto, e ao Professor João Costa, responsável pela unidade curricular, pela orientação, disponibilidade e apoio prestados ao longo do semestre, que foram essenciais para a concretização deste trabalho.

Resumo

Palavras Chave: Estágios, …

# Introdução

Este relatório descreve o trabalho desenvolvido no âmbito da Licenciatura em Engenharia Informática no ISEC (Instituto Superior de Engenharia de Coimbra). O projeto realizado intitula-se "Desenvolvimento de um Sistema de Informação para Gestão de Parcerias de Estágio e Projetos do ISEC".

O projeto teve como objetivo desenvolver uma plataforma centralizada para a gestão toda a informação relativa às colaborações entre o ISEC e empresas externas, no âmbito de estágios e projetos de investigação. Através desta solução, pretendeu-se facilitar a gestão, comunicação e acompanhamento destas parcerias, promovendo maior eficiência administrativa e transparência nas interações entre os vários intervenientes: alunos, docentes e entidades parceiras.

Este projeto surge da necessidade de colmatar uma lacuna no sistema do ISEC, onde não existe atualmente uma ferramenta centralizada para gerir e monitorizar as colaborações com empresas no âmbito de estágios e projetos de investigação. Com este sistema pretende-se organizar e documentar essas parcerias, facilitando a comunicação, o acompanhamento e a avaliação das oportunidades, promovendo uma gestão mais eficiente e transparente entre alunos, docentes e entidades parceiras.

O desenvolvimento do projeto foi realizado ao longo do semestre com reuniões quinzenais, que serviram para apresentar o trabalho desenvolvido e receber *feedback* dos docentes envolvidos. Estas reuniões permitiram assegurar que o desenvolvimento seguia na direção correta, possibilitando corrigir erros e melhorar certos aspetos da aplicação.

O projeto contou com a orientação do Professor César Páris e com o acompanhamento do Professor João Costa, que atuou como cliente do sistema.

## Estrutura do relatório

O relatório organiza-se em oito capítulos, que abordam as diferentes fases e componentes do projeto:

* **Estado Pré-existente do Projeto:** analisa o sistema antes do desenvolvimento, identificando funcionalidades implementadas, limitações e problemas encontrados.
* **Conceção do Sistema:** apresenta o planeamento e as decisões de conceção do sistema, incluindo a definição dos objetivos do projeto, a arquitetura adotada, as tecnologias utilizadas e os critérios que orientaram o desenvolvimento da aplicação.
* **Implementação da API:** detalha o desenvolvimento da API (*Application Programming Interface*), explicando a modelação dos dados, a estrutura dos *endpoints*, os métodos utilizados e as questões relacionadas com autenticação e segurança.
* **Interface e Design visual:** apresenta as decisões e conceção da interface, abordando aspetos de usabilidade, acessibilidade, organização visual e a implementação prática.
* **Interação com o Sistema:** ilustra o funcionamento através de casos de uso e cenários de interação complementados com a visualização dos ecrãs correspondentes.
* **Desenvolvimento Futuro:** detalha funcionalidades pendentes, potencias melhorias e o planeamento de evoluções futuras do sistema, destacando oportunidades para expansão e otimização do sistema.

# Estado Pré-existente do Projeto

A aplicação existente resultava de uma fase inicial de desenvolvimento, cujo objetivo era criar um sistema que apoiasse este processo. No início do desenvolvimento deste projeto, o ISEC não dispunha de uma ferramenta centralizada que permitisse gerir de forma eficiente as parcerias estabelecidas com empresas no âmbito de estágios e projetos de investigação. A ausência desta solução dificultava o acompanhamento das propostas e candidaturas, a comunicação entre os vários intervenientes e a organização da informação, obrigando a uma gestão fragmentada e manual. Embora já fosse possível realizar algumas operações básicas, como a submissão de propostas por parte das empresas ou a consulta de oportunidades pelos alunos, a solução encontrava-se incompleta e pouco madura. O projeto assumiu, assim, como ponto de partida a análise desta versão inicial, com o intuito de compreender a sua estrutura, identificar limitações e estabelecer uma base sólida para a reestruturação e evolução do sistema.

No início do projeto, a aplicação encontrava-se parcialmente desenvolvida, pelo que foi necessário realizar testes práticos para compreender o seu funcionamento e identificar as funcionalidades implementadas. A partir desta análise, foi possível identificar as seguintes funcionalidades:

* **Administrador:**
  + Gestão de entidades: O administrador tinha acesso a um painel onde podia criar, editar e remover empresas, cursos, calendários, alunos e propostas.
  + Importação de alunos via ficheiro Excel: Existia uma funcionalidade que permitia importar listas de estudantes para o sistema a partir de um ficheiro Excel.
* **Empresa:**
  + Submissão de propostas: As empresas registadas tinham a possibilidade de criar propostas de estágio ou projeto, preenchendo um formulário.
  + Consulta do estado das propostas: Após a submissão, as empresas podiam consultar as suas propostas e verificar o estado de aprovação pelo administrador, mas não podiam editá-las caso fosse necessário atualizar ou corrigir dados.
* **Aluno:** 
  + Edição de perfil: Cada aluno tinha acesso a um perfil pessoal onde podia atualizar dados académicos (médias, curso, ano letivo) e anexar o seu currículo.
  + Consulta de propostas: A plataforma disponibilizava uma listagem com todas as propostas aprovadas, permitindo que os alunos navegassem pelas oportunidades de estágio ou projeto.
  + Submissão e acompanhamento de candidaturas: Os alunos podiam candidatar-se às propostas disponíveis e acompanhar o estado da sua candidatura (pendente, aceite ou rejeitada).

Durante a avaliação, foram identificadas algumas limitações, nomeadamente: a necessidade de atualizar manualmente a página para visualizar novas propostas, a impossibilidade por parte das Empresas editarem as suas próprias propostas, restrições de acesso do administrador a candidaturas e propostas aprovadas, e validação insuficiente de alguns campos. Posteriormente, foi realizada uma análise da arquitetura e da estrutura do código, que revelou uma organização pouco estruturada, baixa modularidade e falta de documentação.

Apesar de o sistema já apresentar uma separação clara entre *backend*, desenvolvido em Django [1], e *frontend*, implementado em React [2], a análise inicial do código revelou um conjunto significativo de limitações técnicas que dificultavam a evolução da aplicação. A organização das pastas e dos módulos não seguia convenções ou boas práticas de desenvolvimento, resultando numa baixa modularidade e tornando difícil a reutilização de componentes e a implementação de novas funcionalidades. A ausência de documentação dificultou ainda mais a compreensão do fluxo e organização da aplicação, obrigando a uma análise detalhada do código para identificar as responsabilidades de cada componente e a lógica de negócio associada.

Foram igualmente identificados problemas ao nível da experiência de utilização. Algumas operações exigiam recarregamentos manuais das páginas para atualizar dados, existiam restrições desnecessárias de acesso a determinadas funcionalidades e a validação de campos de entrada era insuficiente, aumentando a probabilidade de erros ou inconsistências na base de dados. Do ponto de vista arquitetural, a API apresentava uma estrutura pouco escalável, sem uma camada clara de abstração entre os modelos, controladores e *endpoints*, o que dificultava a manutenção e a integração de novos módulos.

Estas fragilidades tornaram evidente a necessidade de uma reestruturação profunda da aplicação, tanto a nível técnico como funcional, para garantir um sistema robusto, seguro e preparado para evoluções futuras e acomodar de forma adequada os objetivos definidos.

# Planeamento e Conceção do Sistema

A análise do sistema pré-existente revelou limitações na modularidade, organização do código e documentação. Estas limitações dificultavam a manutenção e tornavam a expansão do sistema uma tarefa bastante complexa.

Face a este cenário, tornou-se necessário refazer o sistema do zero, aproveitando apenas componentes e estruturas do sistema anterior que se mostrassem estáveis e reutilizáveis. Esta abordagem criou uma base estruturada e documentada, preservando o conceito geral da arquitetura original, que respeitava boas práticas de separação de responsabilidades entre cliente e servidor. Adicionalmente, foram mantidas as tecnologias anteriores, por serem soluções consolidadas, escaláveis e compatíveis com futuras evoluções. Desta forma, as novas funcionalidades puderam ser implementadas de forma eficiente e sustentável, garantindo que o sistema se mantivesse escalável e fácil de manter a longo prazo.

## Objetivos

A análise do sistema existente e as reuniões realizadas com os docentes envolvidos no projeto, permitiram definir os objetivos que orientaram o desenvolvimento da nova versão da aplicação. Estes requisitos respondem às necessidades identificadas, com foco na criação de uma arquitetura robusta, modular e estruturada, de forma a facilitar a evolução e manutenção do sistema a longo prazo.

Para facilitar a organização e priorização dos requisitos, foi utilizada a notação MoSCoW [3], que classifica cada funcionalidade de acordo com a sua importância:

* ***Must*** – Funcionalidade essencial que deve ser incluída no sistema para que este funcione corretamente e cumpra os objetivos principais.
* ***Should*** – Funcionalidade importante, mas não crítica; deve ser incluída se possível, mas o sistema ainda funciona sem ela.
* ***Could*** – Funcionalidade desejável ou opcional; melhora a experiência do utilizador ou adiciona valor, mas não é essencial.
* ***Won’t*** – Funcionalidade que não será implementada nesta fase do projeto, podendo ser considerada para versões futuras.

Figura 1 – Listagem de Funcionalidades

|  |  |  |  |
| --- | --- | --- | --- |
| # | Funcionalidade | Descrição | Prioridade |
| Geral | | | |
| F01 | Exportação de atribuições | Gerar ficheiro Excel com a listagem final de alunos, empresas e estágios/projetos atribuídos. | Should |
| F02 | Exportação de propostas | Gerar um ficheiro Excel contendo a listagem final das propostas registadas no sistema, organizada de acordo com o calendário definido. | Should |
| F03 | Automação das atribuições | Desenvolvimento de um algoritmo que realiza automaticamente a distribuição dos alunos pelas propostas, garantindo maior rapidez no processo | Could |
| F04 | Criação automática do documento da proposta | Preenchimento automático do PDF da proposta, utilizando um *template* predefinido | Could |
| F05 | Redesign da interface | Redesenhar a interface gráfica melhorando a acessibilidade, responsividade e usabilidade. | Could |
| F06 | Sistema de comunicação centralizado | Espaço único no sistema para troca de mensagens entre alunos, empresas, docentes e administradores, centralizando a comunicação. | Could |
| F07 | Preenchimento automático de Protocolos | Gerar o protocolo com dados preenchidos automaticamente, prontos para assinatura pelas partes envolvidas. | Could |
| F08 | Preferência de orientadores | Empresas e alunos podem indicar as suas preferências relativamente aos docentes, que são depois consideradas por um algoritmo responsável por definir a atribuição final de orientadores aos estágios/projetos. | Could |
| F09 | Filtragem de Listas | Possibilidade de filtrar todas as listas (propostas, candidaturas, docentes, alunos, etc.) na aplicação. | Should |
| Autenticações e Registos | | | |
| F10 | Autenticação | Login na plataforma. | Must |
| F11 | Registo de Docentes | Registo de um Docente no sistema | Must |
| F12 | Registo de Alunos | Registo de um Aluno no sistema | Must |
| F13 | Registo de Empresas | Registo de uma Empresa no sistema | Must |
| F14 | Registo de Representantes | Registo de um Representante no sistema | Must |
| Gestão do Sistema | | | |
| F15 | Alterar Password do Admin | Redefinir a Password do Administrador do Sistema | Should |
| F16 | Alterar Password por defeito dos Alunos | Redefinir a Password por defeito na criação de um Aluno pelo Admin | Should |
| F17 | Alterar Password por defeito dos Docentes | Redefinir a Password por defeito na criação de um Docente pelo Admin | Should |
| Gestão de Áreas Científicas | | | |
| F18 | Criar | Criar uma Área Científica | Must |
| F19 | Editar | Mudar o nome de uma Área Científica | Must |
| F20 | Remover | Remover uma Área Científica do sistema | Must |
| Gestão de Cursos | | | |
| F21 | Criar | Criar um Curso | Must |
| F22 | Editar | Editar a informação de um Curso | Must |
| F23 | Remover | Remover um Curso do sistema | Must |
| Gestão de Calendários | | | |
| F24 | Criar | Criar um Calendário associado a um Curso | Must |
| F25 | Editar | Editar as datas e informação de um Calendário | Must |
| F26 | Remover | Remover um Calendário do sistema | Must |
| Gestão de Docentes | | | |
| F27 | Criar | Criar um Docente | Must |
| F28 | Editar | Editar a informação de um Docente | Must |
| F29 | Desativar | Desativar um Docente | Must |
| Gestão de Alunos | | | |
| F30 | Criar | Criar um Aluno | Must |
| F31 | Editar | Editar a informação de um Aluno | Must |
| F32 | Desativar | Desativar um Aluno | Must |
| Gestão de Empresas | | | |
| F33 | Editar | Editar a informação da Empresa | Must |
| F34 | Desativar | Desativar uma Empresa | Must |
| Gestão do Representantes | | | |
| F35 | Editar | Editar a informação de um Representante | Must |
| F36 | Desativar | Desativar um Representante | Must |
| Gestão de Propostas | | | |
| F37 | Editar | Editar os dados da Proposta | Must |
| F38 | Remover | Remover uma Proposta | Must |
| Gestão de Candidaturas | | | |
| F39 | Editar | Editar o estado de uma Candidatura | Must |
| Submissão de Propostas | | | |
| F40 | Submissão por parte de uma Empresa | Uma Empresa submeter uma Proposta | Must |
| F41 | Submissão por parte de um Docente | Um Docente submeter uma Proposta | Must |
| Fluxo do Representante | | | |
| F42 | Orientação | Um Representante ter acesso às Propostas às quais é orientador | Must |
| F43 | Gestão de Propostas | Gestão igual à das empresas de Propostas de Projeto | Must |
| F44 | Convidar Representantes | Um Representante poder convidar, via email, um utilizador para o sistema, ficando o mesmo já associado à Empresa. | Could |
| F45 | Edição do Perfil | Um Representante poder editar a informação relativa ao seu perfil | Must |
| Fluxo do Docente | | | |
| F46 | Permissões | Cada Docente tem um conjunto de permissões (visualizar, editar, remover) em cada Módulo do sistema. Estas permissões dão acesso ao Docente a ter uma gestão similar à do Administrador | Must |
| F47 | Orientação | Um Docente ter acesso às Propostas às quais é orientador | Must |
| F48 | Gestão de Propostas | Gestão igual à das empresas de Propostas de Projeto | Must |
| F49 | Edição do Perfil | Um Docente poder editar a informação relativa ao seu perfil | Must |
| Fluxo do Aluno | | | |
| F50 | Listar Propostas disponíveis | O Aluno poder listar todas as propostas, após a sua divulgação, do calendário a qual pertence | Must |
| F51 | Submeter Candidatura | Submeter uma Candidatura com um mínimo e um máximo de Propostas, definidas pelo Calendário | Must |
| F52 | Estado da Candidatura | Acompanhamento do estado da Candidatura realizada | Must |
| F53 | Edição do Perfil | Um Aluno poder editar a informação relativa ao seu perfil | Must |

## Arquitetura do sistema

A aplicação segue uma arquitetura de cliente-servidor, onde a interface (cliente) comunica com uma API que funciona como camada intermediária com a base de dados (servidor). O servidor processa toda a lógica de dados enquanto o cliente é responsável por apresentar os dados e recolher os *inputs* do utilizador. Esta separação permite uma maior escalabilidade, organização dos diferentes componentes do sistema.

Uma imagem com texto, captura de ecrã, diagrama, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 2 – Arquitetura do Sistema

O sistema utiliza Django para gestão de dados e lógica da aplicação (*backend* ou servidor) e React para a interface com o utilizador (*frontend* ou cliente).

Django é uma *framework* em Python que segue o modelo MVT (*Model-View-Template*). O *Model* é responsável pela representação dos dados e pela interação com a base de dados. O *View* gere o fluxo de dados e lida com os pedidos do utilizador. Já o *Template* descreve a apresentação dos dados.

No contexto deste sistema, Django foi utilizado para construir a API, através da qual o *frontend* pode aceder aos dados e funcionalidades da aplicação. Esta API expõe os dados em formato JSON (*JavaScript Object Notation*) e permite operações como consulta, criação, atualização e eliminação de dados, garantindo uma separação clara entre a lógica de dados e a interface. As interações entre o React e Django são feitas através de chamadas HTTP (*HyperText Transfer Protocol*).

React é a ferramenta principal para a construção da interface do utilizador. Esta *framework* JavaScript permite criar componentes dinâmicos e reutilizáveis que interagem com fontes externas — neste caso, a API criada em Django. Desta forma, o conteúdo da página atualiza-se em tempo real, sem ser necessário recarregar a página por completo. De maneira a facilitar a construção da interface, e permitindo também a sua responsividade, foi usado Bootstrap, uma *framework* de código aberto desenvolvida com o objetivo de facilitar a organização de conteúdo usando classes e estruturas definidas de modo a acelerar o processo de desenvolvimento [4].

## Tecnologias usadas

Durante o desenvolvimento do sistema, foram utilizadas algumas ferramentas que apoiaram a implementação e organização da aplicação. Durante esta fase foram usados dois IDEs distintos, cada um escolhido com base nas necessidades especificas do *frontend* e do *backend*.

As tecnologias e ferramentas utilizadas incluem:

* **Visual Studio Code (VSCode)** [5]: IDE usado para o desenvolvimento do *frontend*. Foi escolhido por ser leve e altamente extensível, com diversas extensões de comunidade que melhoram a produtividade no desenvolvimento [6]. Embora não seja ideal para *debug[[1]](#footnote-1)*, não representa uma limitação significativa, no caso do *frontend*, pois o *debug* é realizado diretamente no *browser*.
* **PyCharm** [7]: IDE usado para o desenvolvimento da API em Django. Oferece suporte avançado a Python, *debug* gráfico, testes unitários integrados, gestão de dependências e integração com sistemas de controlo de versões [8].
* **GitHub** [9]: Plataforma usada para controlo de versões onde foi possível manter todo o código desenvolvido guardado de uma maneira segura.
* **Figma** [10]: Ferramenta utilizada para planear e visualizar o *design* da interface gráfica. Serviu de apoio ao desenvolvimento do *frontend*, permitindo definir layouts e testar a organização visual antes da implementação.

# Implementação da API

Uma vez definidos os objetivos do sistema, foi necessário reestruturar o modelo de dados e refazer a API de forma a acomodar as mudanças introduzidas. A estrutura foi pensada de maneira a respeitar as regras de negócio identificadas. A API foi desenvolvida com o objetivo de ser modular, facilitando a organização do código, tornando o de fácil manutenção e permitindo futuras adaptações.

## Modelo de Dados

Para responder às necessidades da aplicação, procedeu-se à reconstrução do modelo de dados. O modelo de dados foi pensado para representar todas as entidades e informação relevantes, incluindo utilizadores e outros dados essenciais.

A aplicação contempla quatro tipos de utilizadores, cada um com permissões e responsabilidades específicas: Administrador, Docente, Representante[[2]](#footnote-2) e Aluno.

Uma imagem com texto, file, diagrama, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Figura 3 – Diagrama Entidade-Relação

Todos os utilizadores são geridos através da classe central ***Accounts***, que estende o modelo ***AbstractUser*** do Django. Esta classe adiciona campos personalizados, como o tipo de utilizador e a fotografia de perfil, mantendo compatibilidade com o sistema de autenticação do Django.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 4 – Classe *Accounts*

Cada tipo de utilizador é depois especializado através de modelos associados por uma relação de um para um com *Accounts*. Por exemplo, para os alunos:

Uma imagem com texto, cartão de visita, Tipo de letra, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Figura 5 – Classe Aluno

## Implementação da API

A API foi desenvolvida em Django, uma framework em Python que oferece uma abstração para a interação com a base de dados através de ORM (*Object Relational Mapping*), isto é, as operações sobre os dados podem ser feitas diretamente no código, sem a necessidade de escrever *queries* em SQL manualmente.

A arquitetura da API segue o padrão *RESTful* [11], permitindo a comunicação entre cliente e servidor através de métodos HTTP como GET, POST, PUT e DELETE. Toda a troca de informação é feita em JSON, serializando os dados de maneira a garantir compatibilidade e simplicidade na comunicação.

Para assegurar a segurança e o controlo de acesso, cada pedido deve ser acompanhado de um JWT (JSON *Web Token*), que é obtido no momento de *login*. Este *token* identifica o utilizador perante o sistema, que, por sua vez, verifica se o utilizador tem permissões para aceder ao recurso solicitado. A ausência do *token*, ou envio de um *token* inválido, resulta na rejeição do pedido.

## Organização do projeto

A estrutura da API segue a organização típica de uma aplicação em Django, com responsabilidades bem definidas em ficheiros distintos:

* ***controllers***–pasta que contém subpastas com os ficheiros *views.py* e *urls.py* organizados por áreas funcionais, dividindo a lógica da aplicação em partes distintas e promovendo uma melhor modularidade.
* ***tasks*** – pasta onde se encontram todas as tarefas assíncronas do sistema. Cada tarefa está organizada em ficheiros separados, de forma a facilitar a manutenção e modularidade. Existe ainda um ficheiro *base.py*, responsável por centralizar as chamadas dessas tarefas
* ***templates*** – pasta que contém os *templates* de documentos oficiais utilizados pelo sistema. Inclui, por exemplo, ficheiros Word para propostas e ficheiros Excel para listagens exportadas. Estes *templates* permitem gerar documentos automaticamente, preenchendo-os com os dados de forma consistente e padronizada.
* ***models.py*** – ficheiro onde é definida a estrutura de dados da aplicação, como os Cursos, Propostas ou tipos de utilizador, representando a estrutura da base de dados.
* ***views.py*** –ficheiroonde estão todas as funções de manipulação de dados, tais como listar, visualizar, editar, eliminar e outras operações necessárias. Estas funções são diretamente chamadas quando uma chamada é feita à API.
* ***urls.py***–ficheiro onde são definidos todos os caminhos (*endpoints*) para cada funcionalidade da API.

## Funcionamento da API

O funcionamento da API assenta numa lógica onde cada recurso (como alunos, propostas ou cursos) tem um conjunto de endpoints associados que seguem uma estrutura padronizada. Esta organização facilita tanto a manutenção do código como facilita a utilização da API.

Cada controlador está associado a um modelo da aplicação e define os endpoints responsáveis por um conjunto de operações. Os caminhos definidos para cada operação seguem o seguinte padrão:

Figura 6 – Estrutura padrão dos endpoints da API

|  |  |  |
| --- | --- | --- |
| Método | Caminho[[3]](#footnote-3) | Descrição |
| GET | api/students/ | Lista os Alunos |
| GET | api/student/{id} | Retorna os dados de um Aluno |
| POST | api/student/create | Cria um Aluno novo |
| PUT | api/student/{id}/edit | Atualiza os dados de um Aluno |
| DELETE | api/student/{id}/delete | Elimina um Aluno |

Esta estrutura é aplicada a todos os modelos principais da aplicação, como Alunos, Docentes, Cursos, entre outros.

Todos os endpoints protegidos exigem que o token de autenticação seja enviado no cabeçalho do pedido. Caso o token esteja ausente ou inválido, a API responde com o código[[4]](#footnote-4) 401. Em caso de sucesso, é devolvido o código apropriado (200 ou 201) acompanhado dos dados solicitados, quando aplicável.

No total, foram criados 56 endpoints para suportar todas as funcionalidades da aplicação. A lista completa dos endpoints encontra-se disponível no Anexo A.

## Geração Automática de Documentos

A API disponibiliza a funcionalidade F0X, responsável pela geração dinâmica de um PDF de uma proposta, a partir dos dados armazenados no sistema. Este processo ocorre inteiramente no servidor, garantindo consistência, segurança e formatação padronizada.

Para implementar esta funcionalidade foram utilizadas várias bibliotecas: a **docxtpl** permite preencher templates *.docx*[[5]](#footnote-5) com os dados da proposta, substituindo marcadores por campos como curso, ano letivo, semestre, título, descrição e objetivos da proposta [12]; a **docx2pdf** realiza a conversão do ficheiro Word para PDF [13]; e a biblioteca **tempfile** gere a criação de ficheiros temporários, evitando armazenamentos desnecessários no servidor [14].

O *template* utilizado encontra-se na pasta *templates/docs* com o nome “proposal\_template.docx”. Este ficheiro utiliza uma sintaxe semelhante à do Jinja2[[6]](#footnote-6), permitindo a definição de *placeholders* do tipo *{{nome\_do\_conteudo}}*, que correspondem aos campos da proposta no sistema. Para preencher estes *placeholders*, a API define um dicionário no código, onde cada chave corresponde a um marcador no documento e cada valor ao conteúdo a inserir.

Uma imagem com texto, Tipo de letra, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Figura 7 - Exemplo de *placeholders* no template e correspondência no código

Durante a execução da funcionalidade, todos os *placeholders* são substituídos pelos valores correspondentes, gerando um documento Word completo. Este documento é posteriormente convertido em PDF e enviado diretamente ao utilizador, sendo que o título do ficheiro é gerado automaticamente com base em informações da proposta, como o ano letivo, o ID da proposta, o semestre e o título da proposta.

## Tarefas assíncronas

Para permitir a execução de tarefas em *background* de forma assíncrona, foi integrado o Celery na API. O Celery [15] é uma biblioteca Python que possibilita o agendamento e execução de tarefas fora do fluxo principal da aplicação, garantindo que operações demoradas ou periódicas não bloqueiem o processamento de pedidos do utilizador.

O agendamento das tarefas foi configurado diretamente no ficheiro *settings.py* do projeto Django, usando o Redis [16] como *broker*[[7]](#footnote-7):

Uma imagem com texto, captura de ecrã, cartão de visita, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 8 - Agendamento da tarefa principal

A função agendada está definida no ficheiro *base.py* dentro da pasta *tasks*. Esta função percorre todos os calendários registados e, dependendo da data atual, invoca tarefas específicas para gerir orientações e colocações, garantindo que ações periódicas importantes são processadas automaticamente.

Uma imagem com texto, captura de ecrã, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 9 - Tarefa assíncrona

Apesar de a infraestrutura para tarefas assíncronas estar configurada e pronta para utilização, não foi possível, devido a limitações de tempo, colocar em funcionamento os requisitos **F03** e **F08**, que estavam planeadas como automações recorrentes do sistema.

A implementação atual estabelece uma base sólida para a integração futura de tarefas assíncronas, permitindo a execução de operações periódicas e de longa duração sem comprometer o desempenho do sistema principal.

# Interface e Design visual

A interface da aplicação não serve apenas para interagir com as funcionalidades disponíveis — ela representa a imagem do instituto. Sendo uma aplicação com o objetivo de ser usada tanto internamente, por alunos e docentes, como externamente por empresas e os seus representantes, foi necessário planear uma nova “face” para o sistema.

Com esse objetivo em mente, foi criado design que respeita princípios de simplicidade, clareza e consistência visual, garantindo uma experiência de utilização mais acessível para todos os perfis de utilizador.

## Conceção visual do sistema

A estrutura visual da aplicação baseia-se num *design system* com o objetivo de garantir consistência visual em toda a interface da aplicação, facilitar a reutilização de componentes e acelerar o desenvolvimento. Neste sistema foram definidas as cores, tipografia, espaçamentos e componentes visuais a utilizar em todas as páginas.

### Cores

A paleta de cores foi pensada para refletir a identidade visual do instituto, escolhendo como cor de destaque a cor tijolo, representativa do ISEC. Para garantir legibilidade e conforto visual, foram incluídas cores neutras destinadas a fundos e textos. Além disso, uma cor secundária foi adicionada para criar contrastes e trazer dinamismo ao design.

Cada cor da paleta é composta por nove tonalidades, organizadas numa escala que vai do tom mais claro ao mais escuro. Esta estrutura permite utilizar diferentes intensidades da mesma cor conforme o contexto facilitando a aplicação coerente e intuitiva das cores em toda a interface.

Uma imagem com texto, captura de ecrã, design, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 10 – Representação de uma cor com as suas nove variações

### Tipografia

A tipografia da aplicação utiliza duas fontes distintas, com papéis complementares na hierarquia visual da interface.

* **Montserrat** é aplicada em títulos, subtítulos e textos de maior destaque.
* **Inter** é utilizada no corpo de texto.

Os tamanhos tipográficos foram definidos com base numa escala proporcional do tipo *Minor Third*, que utiliza uma progressão harmónica para garantir consistência entre os diferentes níveis de texto.

O cálculo é feito a partir de um tamanho base — por exemplo, 16px — e os restantes tamanhos são obtidos multiplicando esse valor pelo fator da escala (aproximadamente 1,2). Assim, os níveis seguintes seriam aproximadamente 19px, 23px, 28px, e assim sucessivamente. Este sistema assegura uma evolução coerente entre títulos, subtítulos e corpo de texto, reforçando a hierarquia visual da interface.

### Espaçamentos

Para garantir uma experiência visual harmoniosa e adaptada a diferentes dispositivos, foram definidos dois sistemas de espaçamentos distintos: um para dispositivos com ecrãs de maiores dimensões e outro para dispositivos móveis, como telemóveis. Em ambos os casos, aplicou-se uma escala modular semelhante à utilizada na tipografia, mas com fatores de escala diferentes.

Nos dispositivos com ecrãs maiores, utilizou-se a escala *Golden Ratio* (fator de escala 1,618), que proporciona uma progressão suave e visualmente agradável. Para dispositivos móveis, adotou-se a escala *Major Second* (fator de escala 1,125), oferecendo uma progressão mais subtil e adequada ao espaço limitado disponível.

### Componentes

Com todos os estilos definidos, avançou-se para a criação dos componentes da interface, como a barra de navegação, botões, cartões, entre outros. Estes componentes foram desenvolvidos de forma modular e reutilizável, podendo ser aplicados em diferentes páginas e contextos, adaptando-se aos diversos perfis de utilizador.

Esta abordagem assegura uma experiência visual coerente e ao mesmo tempo simplifica a manutenção e permite a expansão do sistema.

## *Design* da Aplicação

Com a identidade visual do sistema definida, avançou-se para o desenho dos ecrãs da aplicação, desenvolvidos a partir das funcionalidades requeridas. A estrutura da interface foi organizada em quatro fluxos principais, correspondentes aos diferentes perfis de utilizador: Administrador, Docente, Aluno e Representante.

Para assegurar flexibilidade e facilitar a manutenção, as páginas foram projetadas de maneira modular e reutilizável, permitindo o compartilhamento de componentes comuns entre os diferentes fluxos. Essa abordagem modular também possibilita a adaptação das interfaces conforme as permissões específicas de cada utilizador, incluindo o acesso avançado de alguns docentes, similar ao do administrador, conforme definido nos requisitos do sistema.

Adicionalmente, a navegação foi estruturada tendo em conta a experiência do utilizador, garantindo rapidez e clareza no acesso às diversas funcionalidades, minimizando a curva de aprendizagem e melhorando o uso da aplicação. Foram criados 80 ecrãs que abrangem todas as funcionalidades previstas para os diferentes perfis, assegurando uma cobertura completa das necessidades do sistema.

A documentação completa relativa ao design encontra-se no Anexo B, onde estão descritas todas as decisões tomadas ao nível visual.

## Implementação da Interface

A implementação da interface do sistema foi feita utilizando React, o que possibilitou estruturar o frontend em componentes modulares e reutilizáveis. O CSS foi usado para personalizar os estilos. Além disso, o Bootstrap foi incorporado para acelerar o desenvolvimento de layouts responsivos e consistentes, facilitando a adaptação da aplicação a diferentes dispositivos. Essa combinação de tecnologias resultou numa interface flexível e de fácil manutenção.

### Organização do projeto

A estrutura do projeto seguiu a organização típica de uma aplicação desenvolvida em React, com uma separação clara entre os diferentes tipos de elementos: componentes, páginas, estilos, serviços, entre outros. Esta divisão garante um código limpo, modular e fácil de manter. Na raiz do projeto temos:

* ***assets*** – Contém os recursos estáticos utilizados na aplicação, como imagens, ícones e ficheiros gráficos.
* ***styles*** –Contém os ficheiros CSS responsáveis pela aparência global da aplicação.
* ***componentes*** –Reúne os componentes reutilizáveis da interface, como botões, cabeçalhos e cartões.
* ***pages*** –Contém todas as páginas da aplicação.
* ***services*** – Contém todas as funções que centralizam a comunicação com a API.
* ***utils*** – Inclui funções auxiliares que suportam a lógica no *frontend*.

## Arquitetura da Interface

#### Navegação e gestão de rotas

A navegação da aplicação foi implementada utilizando a biblioteca *react-router-dom*, com o controlo centralizado no ficheiro *App.js*. Este ficheiro define os caminhos que permitem o acesso às várias páginas e módulos do sistema, assegurando que cada utilizador visualiza apenas as secções adequadas ao seu perfil e respetivas permissões. Esta abordagem garante uma navegação fluida e intuitiva, facilitando a utilização diária.

#### Componentes reutilizáveis e flexibilidade

Os componentes da interface foram organizados na pasta *components* e construídos com base nos elementos do *design system*. Estes componentes são reutilizáveis e flexíveis, permitindo a passagem de estilos adicionais através da propriedade *className*, o que possibilita a adaptação do seu estilo conforme o contexto de utilização.

#### Estrutura modular das páginas

A estrutura das páginas segue uma lógica modular, com cada módulo funcional organizado na pasta *pages/Modules*. Aqui cada módulo possui as suas páginas específicas de listagem, visualização e edição. Esta organização facilita a escalabilidade do sistema e mantém o código organizado e de fácil gestão.

Um Módulo típico é estruturado da seguinte forma:

* ***View*** – pasta que contém os ficheiros *.jsx* [[8]](#footnote-8) e *.css [[9]](#footnote-9)* da página de visualização.
* ***Edit*** – pasta que contém os ficheiros *.jsx* e *.css* da página de edição/criação.
* ***List*** – pasta que contém os ficheiros *.jsx* e *.css* da página de listagem.
* ***Index,js*** – ponto de exportação de todas as páginas do módulo.

#### Definição e organização dos estilos

A gestão dos estilos na aplicação foi pensada de forma modular e organizada. As variáveis CSS que sustentam a identidade visual do sistema estão definidas em ficheiros separados dentro da pasta *styles*, com cada ficheiro dedicado a uma responsabilidade específica – por exemplo, cores (*\_colors.css*) ou tipografia (*\_fonts.css*). Estes ficheiros são todos importados através de um único ponto central, o ficheiro *imports.css*, garantindo que todas as definições ficam disponíveis globalmente na aplicação.

A nomenclatura das variáveis segue uma convenção coerente, baseada nas diretrizes do *design system*. Os nomes utilizam uma estrutura padronizada com valores dinâmicos indicados entre chavetas {}*[[10]](#footnote-10)*:

* **Cores:** *--{*nome*}-color-{***valor***}* – onde o valor vai de 100 a 900 (em saltos de 100), representando diferentes intensidades da mesma cor.
* **Espaçamentos:** *--spacing-{***tamanho***}* – com valores em *t-shirt size* (xs, s, m, l, xl, 2xl, 3xl, 4xl).
* **Tamanhos de texto[[11]](#footnote-11):** *--fs--{***valor***}* – de 100 a 950, com saltos de 100.
* **Tamanhos de ícones:** *--icons-size-{***tamanho***}* – também seguindo o padrão *t-shirt* size (xs a l).

Adicionalmente, foram definidas cores utilitárias que seguem uma convenção padronizada para facilitar o seu uso em diferentes contextos da interface. Estas cores não estão associadas diretamente à identidade visual da aplicação, mas servem como um conjunto de apoio para elementos funcionais.

Cada cor utilitária possui três variantes principais:

* *{***cor***}* – a cor base.
* {**cor**}*-lighter* – uma versão mais clara da cor.
* {**cor**}*-darker* – uma versão mais escura da cor.

O conjunto inclui cores como vermelho, verde, azul, roxo, rosa, turquesa, castanho, laranja e amarelo, assegurando diversidade suficiente para múltiplas necessidades visuais sem comprometer a coerência do design.

Por sua vez, o ficheiro *App.css* aplica estilos base e genéricos à interface, definindo comportamentos comuns para elementos HTML e componentes globais. Este ficheiro funciona como camada de normalização visual da aplicação, aplicando regras consistentes que usam as variáveis importadas a partir do *imports.css*.

#### Suporte para *Markdown*

Durante o desenvolvimento do *frontend*, surgiu a necessidade de permitir que os utilizadores tivessem algum controlo sobre a formatação do texto em determinados campos (por exemplo, a descrição de propostas ou cursos). Como o foco da aplicação não é fornecer um editor de texto avançado, criar um sistema completo de edição de texto seria demasiado complexo e desnecessário para o objetivo do projeto.

Para resolver este problema, optou-se por utilizar Markdown [17], uma linguagem de marcação simples que possibilita a formatação de texto de forma intuitiva e leve. O texto é armazenado na base de dados em formato literal, ou seja, contendo todos os símbolos que definem o estilo[[12]](#footnote-12).

No frontend, a biblioteca “react-markdown” é responsável por interpretar este conteúdo e renderiza-lo como HTML. Para garantir que o resultado respeita as definições visuais da aplicação, o componente *<Markdown>* é sempre envolvido por uma *div* de classe *markdown*.

Uma imagem com texto, captura de ecrã, Tipo de letra, cartão de visita

Os conteúdos gerados por IA podem estar incorretos.

Figura 11 - Estrutura *Markdown*

# Interação com o Sistema

Entre os diversos fluxos implementados durante o desenvolvimento, apresentam-se a seguir aqueles que melhor representam as funcionalidades-chave e os objetivos do sistema. Estes fluxos foram selecionados por refletirem processos essenciais e por permitirem uma compreensão mais clara do funcionamento global da aplicação. Através da análise destes exemplos, será possível observar tanto a interação do utilizador com a interface como a lógica subjacente que suporta as operações mais relevantes do sistema.

## Autenticação de um utilizador (*login*)

O processo de autenticação de um utilizador segue o procedimento habitual, no qual são solicitadas as credenciais (email e palavra-passe). Estas são enviadas à API para validação. Em caso de sucesso, a API retorna um token de autenticação juntamente com informações adicionais relevantes, como o tipo de utilizador e as respetivas permissões (caso seja um docente). Esta informação é então utilizada para apresentar a página inicial (*dashboard*) correta, adaptada ao perfil do utilizador autenticado.

Além disso, o sistema verifica, no momento do acesso, se já existe um token armazenado localmente. Caso este token seja considerado válido – após uma verificação junto à API – o utilizador é automaticamente redirecionado para a sua *dashboard*, evitando assim a necessidade de efetuar o login manualmente.

Uma imagem com céu, captura de ecrã, texto, edifício

Os conteúdos gerados por IA podem estar incorretos.

Figura 12 – Página de *Login*

**Visão do Docente**

Um dos casos em que a página inicial apresenta maior variação é na visão do docente. Para além das funcionalidades básicas, como consultar as suas orientações, submeter propostas e acompanhar as suas propostas submetidas, o sistema adapta-se para oferecer acessos adicionais semelhantes aos de um administrador, dependendo das permissões atribuídas ao utilizador.

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 13 – Página inicial com as permissões padrão

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 14 – Página inicial com todas as permissões

## Filtragem de Listas

Considerando que o sistema lida com grandes volumes de informação, a funcionalidade de filtragem revela-se essencial para facilitar a localização rápida e eficiente dos dados desejados. Sem esta ferramenta, seria complexo e demorado navegar pelas diversas listas.

A filtragem funciona de forma intuitiva: o utilizador pode clicar no cabeçalho da lista para inserir termos de pesquisa diretamente, ou utilizar menus *dropdown* que permitem selecionar critérios específicos para refinar a informação apresentada. Esta abordagem combina simplicidade e flexibilidade, tornando a busca mais eficaz.

Uma imagem com texto, captura de ecrã, file, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 15 – Filtragem de listas

Embora exista ainda margem para melhorias e otimizações futuras, a filtragem implementada já garante uma usabilidade adequada, permitindo ao utilizador restringir a visualização conforme vários critérios e encontrar rapidamente o que procura.

## Gestão de Docentes

Este fluxo ilustra a gestão dos docentes pela visão do administrador, abrangendo as funcionalidades de listagem, visualização, adição/edição e remoção.

A listagem dos docentes apresenta todos os docentes registados no sistema, com a possibilidade de filtragem (como referido anteriormente) para facilitar a consulta rápida. Nesta vista, o administrador tem acesso direto às opções de visualizar, editar, remover e adicionar um novo docente.

Uma imagem com texto, captura de ecrã, número, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 16 – Listagem de docentes

O sistema permite a criação de novos registos de docentes, bem como a edição das informações existentes. Esta funcionalidade é essencial para manter os dados atualizados, incluindo contactos, áreas de especialização e outros detalhes relevantes.

A funcionalidade de visualização de um docente permite consultar informação relevante sobre o docente.

Uma imagem com texto, captura de ecrã, software, Página web

Os conteúdos gerados por IA podem estar incorretos.

Figura 17 – Visualização de um docente

O mesmo ecrã é utilizado tanto para a criação de um novo docente como para a edição de um registo existente. A diferença principal está no preenchimento prévio dos campos: no caso da edição, os dados do docente são carregados por defeito, enquanto na criação os campos estão vazios. Esta abordagem visa reutilizar código, dado que ambas as operações partilham a mesma interface e estrutura de dados.

Uma imagem com texto, captura de ecrã, software, Página web

Os conteúdos gerados por IA podem estar incorretos.

Figura 18 – Edição de um docente

No sistema, a remoção pode assumir formas distintas conforme o contexto. Em casos específicos, como a gestão de docentes a remoção não elimina os registos definitivamente, mas desativa-os. Esta abordagem garante que a informação permaneça preservada para futuras consultas, mantendo a integridade e consistência dos dados. Em outros contextos onde não existe esta necessidade, a remoção poderá ser efetiva, eliminando totalmente o registo do sistema.

Uma imagem com texto, captura de ecrã, software, Página web

Os conteúdos gerados por IA podem estar incorretos.

Figura 19 – Listagem de com um docente removido

Os docentes removidos aparecem na lista com um fundo escuro, como se pode ver na Figura 12.

## Gestão de Calendários

A gestão de calendários no sistema segue um fluxo simples e intuitivo, permitindo ao utilizador visualizar, criar e consultar calendários associados a cada curso.

Para aceder à listagem de calendários, é necessário navegar até à página de um curso específico. Nessa página, o utilizador encontra a lista completa dos calendários criados para o curso selecionado. Os calendários ativos são exibidos normalmente, enquanto os calendários inativos, cujo período já terminou, são apresentados com um fundo escuro, à semelhança dos docentes inativos.

Uma imagem com texto, Tipo de letra, file, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Figura 20 – Listagem de Calendários

#### Criação de um Calendário

Para criar um Calendário, o utilizador seleciona a opção “Adicionar calendário” disponível na página de listagem. O formulário de criação permite definir:

* Ano letivo e semestre;
* Datas importantes do processo: início e fim de submissões, divulgação de propostas, candidaturas e colocações;
* Limites mínimo e máximo de propostas por candidatura.

Uma imagem com texto, captura de ecrã, número, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 21 – Formulário de criação de calendário

#### Visualização de um Calendário

Para consultar um calendário, o utilizador clica no botão “Ver” presente em cada linha da listagem. O sistema apresenta então uma página detalhada com todas as informações associadas ao calendário:

* Datas definidas para submissão, divulgação, candidaturas e colocações;
* Limites mínimo e máximo de propostas por candidatura;
* Listagens relacionadas:
  + Propostas associadas ao calendário;
  + Alunos inscritos neste calendário;
  + Candidaturas submetidas.

Uma imagem com texto, captura de ecrã, software, Ícone de computador

Os conteúdos gerados por IA podem estar incorretos.

Figura 22 – Visualização de um Calendário

## Submissão de uma Proposta

O processo de submissão de Propostas pode ser realizado tanto por um Representante quanto por um Docente. Este exemplo será focado no fluxo do ponto de vista do **Representante**.

Ao aceder à *dashboard*, o representante encontra a opção “Submeter Proposta”, que direciona para o formulário de submissão.

Se não existir nenhum calendário ativo para submissão, o botão estará desativado, impedindo o acesso ao formulário. Caso o utilizador consiga aceder à página por outro meio, é apresentada uma mensagem de erro, informando que não é possível submeter propostas naquele momento.

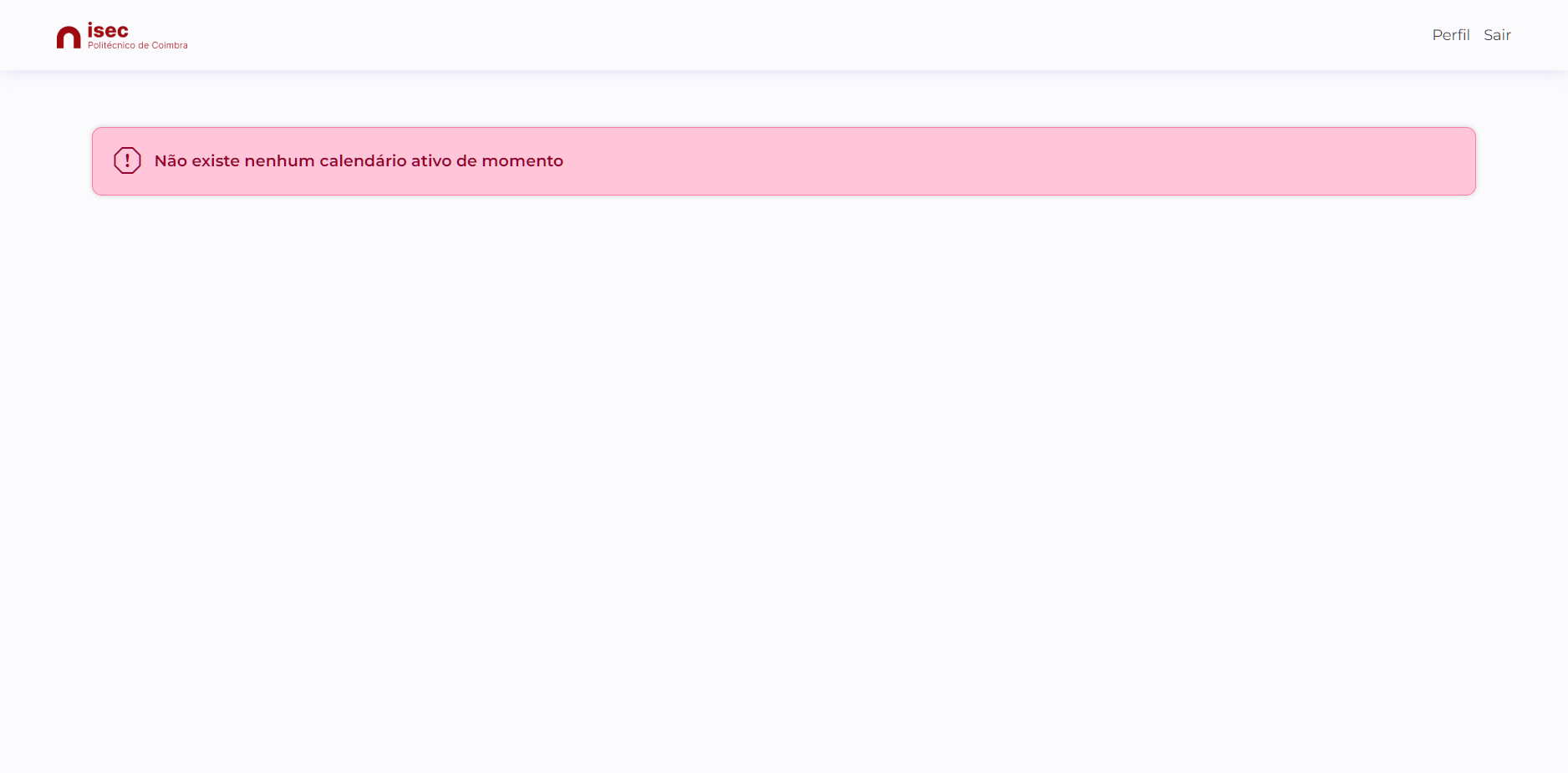


Figura 23 – Mensagem de erro do formulário

Uma imagem com texto, captura de ecrã, número, Paralelo

Os conteúdos gerados por IA podem estar incorretos.

Figura 24 – Formulário de submissão de uma Proposta

No formulário, o representante deve escolher o curso e o calendário ao qual pretende submeter a proposta. Se o curso tiver ramos, será possível selecionar aqueles que são mais adequados à proposta.

Após a seleção do curso, o formulário ajusta dinamicamente os campos apresentados, de acordo com as definições específicas do curso.

Uma imagem com texto, captura de ecrã, file, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 25 – Seleção de curso, ramos e calendário no formulário de submissão

Durante o preenchimento da proposta, o representante deve indicar um Orientador da empresa responsável pela supervisão da proposta. Existem duas opções:

1. Escolher um Representante já registado na empresa.
2. Criar um Orientador, selecionando a opção “Novo Representante” e fornecendo um nome e um email. Esta ação cria automaticamente uma conta de representante associada à empresa.

Uma imagem com texto, captura de ecrã, file, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 26 – Criação de um Orientador no formulário de submissão

## Exportação da Proposta em PDF

O sistema disponibiliza a funcionalidade F04, que permite criar automaticamente um ficheiro PDF com os detalhes de uma proposta.

Na página de visualização da proposta, o utilizador encontra o botão “Proposta”. Ao clicar neste botão:

1. É enviado um pedido à API para gerar o documento.
2. A API processa os dados da proposta e devolve um ficheiro PDF pronto para download.
3. O utilizador recebe o ficheiro com um nome formatado de acordo com o ano letivo, identificador e título da proposta.

Uma imagem com texto, captura de ecrã, Página web, software

Os conteúdos gerados por IA podem estar incorretos.

Figura 27 - Botão para geração do PDF

Uma imagem com texto, captura de ecrã, documento, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.

Figura 28 - Documento PDF da proposta gerado

# Desenvolvimento Futuro

Tendo em conta que este projeto foi desenvolvido no âmbito de uma unidade curricular com duração limitada, não foi possível implementar na íntegra todas as funcionalidades idealizadas. Ainda assim, o desenvolvimento foi orientado por uma visão completa do sistema, e todas as decisões tomadas – tanto a nível de estrutura como de implementação – tiveram em conta o conjunto completo das funcionalidades previstas.

Algumas funcionalidades foram concluídas, outras ficaram parcialmente desenvolvidas (ficando pela interface gráfica), e há ainda elementos que se encontram apenas definidos ou planeados, mas não implementados.

Figura 29 - Estado final das funcionalidades

|  |  |  |  |
| --- | --- | --- | --- |
| # | Funcionalidade | Frontend | Backend |
| Geral | | | |
| F01 | Exportação de atribuições | — | Planeado |
| F02 | Exportação de propostas | — | Parcial |
| F03 | Automação das atribuições | — | Planeado |
| F04 | Criação automática do documento da proposta | — | Completo |
| F05 | Redesign da interface | Completo | — |
| F06 | Sistema de comunicação centralizado | Por Fazer | Por Fazer |
| F07 | Preenchimento automático de Protocolos | — | Por Fazer |
| F08 | Preferência de orientadores | Por Fazer | Por Fazer |
| F09 | Filtragem de Listas | Parcial | — |
| Autenticações e Registos | | | |
| F10 | Autenticação | Completo | Completo |
| F11 | Registo de Docentes | Completo | Completo |
| F12 | Registo de Alunos | Completo | Completo |
| F13 | Registo de Empresas | Completo | Completo |
| F14 | Registo de Representantes | Completo | Completo |
| Gestão do Sistema | | | |
| F15 | Alterar Password do Admin | Completo | Completo |
| F16 | Alterar Password por defeito dos Alunos | Completo | Completo |
| F17 | Alterar Password por defeito dos Docentes | Completo | Completo |
| Gestão de Áreas Científicas | | | |
| F18 | Criar | Completo | Completo |
| F19 | Editar | Completo | Completo |
| F20 | Remover | Completo | Completo |
| Gestão de Cursos | | | |
| F21 | Criar | Completo | Completo |
| F22 | Editar | Completo | Completo |
| F23 | Remover | Completo | Completo |
| Gestão de Calendários | | | |
| F24 | Criar | Completo | Completo |
| F25 | Editar | Completo | Completo |
| F26 | Remover | Completo | Completo |
| Gestão de Docentes | | | |
| F27 | Criar | Completo | Completo |
| F28 | Editar | Completo | Completo |
| F29 | Desativar | Completo | Completo |
| Gestão de Alunos | | | |
| F30 | Criar | Completo | Completo |
| F31 | Editar | Completo | Completo |
| F32 | Desativar | Completo | Completo |
| Gestão de Empresas | | | |
| F33 | Editar | Completo | Completo |
| F34 | Desativar | Completo | Completo |
| Gestão do Representantes | | | |
| F35 | Editar | Completo | Completo |
| F36 | Desativar | Completo | Completo |
| Gestão de Propostas | | | |
| F37 | Editar | Completo | Completo |
| F38 | Remover | Completo | Completo |
| Gestão de Candidaturas | | | |
| F39 | Editar | Completo | Por Fazer |
| Submissão de Propostas | | | |
| F40 | Submissão por parte de uma Empresa | Completo | Completo |
| F41 | Submissão por parte de um Docente | Completo | Completo |
| Fluxo do Representante | | | |
| F42 | Orientação | Completo | Por Fazer |
| F43 | Gestão de Propostas | Completo | Completo |
| F44 | Convidar Representantes | Completo | Completo |
| F45 | Edição do Perfil | Completo | Completo |
| Fluxo do Docente | | | |
| F46 | Permissões | Completo | Completo |
| F47 | Orientação | Por Fazer | Por Fazer |
| F48 | Gestão de Propostas | Completo | Completo |
| F49 | Edição do Perfil | Completo | Completo |
| Fluxo do Aluno | | | |
| F50 | Listar Propostas disponíveis | Completo | Completo |
| F51 | Submeter Candidatura | Por Fazer | Por Fazer |
| F52 | Estado da Candidatura | Planeado | Planeado |
| F53 | Edição do Perfil | Completo | Completo |

Os estados apresentados na Figura 25 têm o seguinte significado:

* Completo – Funcionalidade totalmente implementado.
* Parcial – Funcionalidade parcialmente implementado.
* Planeado – Funcionalidade foi planeada, mas não foi realizada.
* Por Fazer – Funcionalidade ainda não iniciada.
* “—" – Indica que a funcionalidade não está presente na respetiva secção, ou seja, não é responsabilidade desse componente.

Do total de 53 funcionalidades previstas, 39 foram implementadas na íntegra, enquanto 14 ficaram parcialmente desenvolvidas ou apenas planeadas. A lista completa das funcionalidades, com o estado final de cada uma, encontra-se no Anexo C.

Em reflexão, de todo o trabalho realizador existem alguns pontos que poderiam ser melhorados e ou adicionados à aplicação, como por exemplo:

* Melhorar a filtragem de listas com mais opções, como filtrar propostas e alunos pelo calendário a que pertencem, entre outros.
* Melhorar a interface para dispositivos moveis que, apesar de estar implementada a estruturação pode ser melhorada para permitir uma melhor experiência de utilização em dispositivos com ecrãs menores.
* Aplicar um sistema de importação de dados completo, como a importação de Alunos, Docentes, Propostas, Empresas e os seus Representantes, entre outros dados relevantes.
* Criar um sistema de auditoria para guardar as ações tomadas pelo Administrador e Docentes em operações de gestão (criar, editar, remover/desativar), para efeitos de auditoria, rastreabilidade e responsabilização das alterações realizadas no sistema.

# Conclusão

Apreciação do trabalho feito

Referências bibliográficas

|  |  |
| --- | --- |
| [1] | Django Software Foundation, “The web framework for perfectionists with deadlines.,” [Online]. Available: https://www.djangoproject.com. [Acedido em 27 junho 2025]. |
| [2] | Meta Platforms, Inc, “React,” [Online]. Available: https://react.dev. [Acedido em 27 junho 2025]. |
| [3] | K. Brush, “What is the MoSCoW method?,” 29 março 2023. [Online]. Available: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method. [Acedido em 25 agosto 2025]. |
| [4] | Bootstrap team, “Bootstrap · The most popular HTML, CSS, and JS library in the world.,” [Online]. Available: https://getbootstrap.com. [Acedido em 26 junho 2025]. |
| [5] | Microsoft, “Visual Studio Code,” Microsoft, [Online]. Available: https://code.visualstudio.com. [Acedido em 29 maio 2025]. |
| [6] | A. Uspenski, “Why VS Code remains a developer favorite, year after year,” 25 abril 2023. [Online]. Available: https://shiftmag.dev/vs-code-171/?utm\_source=chatgpt.com. [Acedido em 25 agosto 2025]. |
| [7] | JetBrains, “PyCharm,” JetBrains, [Online]. Available: https://www.jetbrains.com/pycharm. [Acedido em 29 maio 2025]. |
| [8] | A. Ryabtsev, “10 Best IDEs and Code Editors for Python/Django Development,” 28 outubro 2024. [Online]. Available: https://djangostars.com/blog/python-ide/?utm\_source=chatgpt.com. [Acedido em 25 agosto 2025]. |
| [9] | GitHub, Inc., “About GitHub,” [Online]. Available: https://github.com/about. [Acedido em 25 agosto 2025]. |
| [10] | Figma, Inc., “Figma,” Figma, Inc., [Online]. Available: https://www.figma.com. [Acedido em 29 maio 2025]. |
| [11] | L. Gupta, “REST API Tutorial: What is REST?,” 1 abril 2025. [Online]. Available: https://restfulapi.net. [Acedido em 20 julho 2025]. |
| [12] | E. Lapouyade, “Welcome to python-docx-template’s documentation!,” 2015. [Online]. Available: https://docxtpl.readthedocs.io/en/latest. [Acedido em 03 setembro 2025]. |
| [13] | A. Johri, “docx2pdf · PyPI,” 11 dezembro 2021. [Online]. Available: https://pypi.org/project/docx2pdf. [Acedido em 03 setembro 2025]. |
| [14] | Python Software Foundation, “tempfile — Generate temporary files and directories,” 03 setembro 2025. [Online]. Available: https://docs.python.org/3/library/tempfile.html. [Acedido em 03 setembro 2025]. |
| [15] | A. S. &. contributors, “Celery - Distributed Task Queue,” 2009. [Online]. Available: https://docs.celeryq.dev/en/v5.5.3/index.html. [Acedido em 02 setembro 2025]. |
| [16] | Redis, “Redis - The Real-time Data Platform,” [Online]. Available: https://redis.io. [Acedido em 02 setembro 2025]. |
| [17] | M. Cone, “Markdown Guide,” 2025. [Online]. Available: www.markdownguide.org. [Acedido em 26 agosto 2025]. |
| [18] | L. Gupta, “HTTP Status Codes,” 9 agosto 2024. [Online]. Available: https://restfulapi.net/http-status-codes. [Acedido em 20 julho 2025]. |
| [19] | “Markdown Syntax Cheatsheet,” 2021. [Online]. Available: https://www.markdown-cheatsheet.com. [Acedido em 26 agosto 2025]. |
| [20] | Bloomreach, Inc, “Basic Syntax of Jinja,” 2021. [Online]. Available: https://documentation.bloomreach.com/engagement/docs/jinja-syntax. [Acedido em 03 setembro 2025]. |

Anexos

Text

Description automatically generated with medium confidence

1. No contexto de um IDE, *debug* refere-se ao processo de execução controlada de um programa para identificar, analisar e corrigir erros ou comportamentos inesperados. Os IDEs oferecem ferramentas como pausar a execução, inspecionar variáveis e percorrer o código passo a passo facilitando a deteção e correção de erros durante o desenvolvimento. [↑](#footnote-ref-1)
2. Um Representante refere-se a qualquer entidade associada a uma Empresa, seja um funcionário, um orientador ou mesmo um departamento (como, por exemplo, os Recursos Humanos) [↑](#footnote-ref-2)
3. Os valores entre **{}** indicam parâmetros que devem ser substituídos conforme o contexto.

   Por exemplo, api/student/{id} significa que {id} deve ser substituído pelo ID do aluno específico. [↑](#footnote-ref-3)
4. A lista completa de códigos de resposta HTTP e os seus respetivos significados pode ser consultada na referência [18]. [↑](#footnote-ref-4)
5. Um ficheiro .docx é um documento criado no Microsoft Word. [↑](#footnote-ref-5)
6. Para referência adicional da sintaxe de Jinja2, pode ser consultado o recurso disponível em [20] [↑](#footnote-ref-6)
7. Um *broker* é um componente intermédio que gere a transmissão de mensagens entre produtores (quem envia tarefas) e consumidores (quem executa as tarefas). No Celery, o broker assegura que as tarefas são colocadas em fila e entregues aos *workers*, que são os processos responsáveis por executar essas tarefas de forma assíncrona e independente do fluxo principal da aplicação. [↑](#footnote-ref-7)
8. Ficheiros com a extensão *.jsx* são usados em React para definir componentes que combinam JavaScript com HTML. Estes ficheiros permitem criar a estrutura e lógica interativa de uma página ou componente. [↑](#footnote-ref-8)
9. Ficheiros com a extensão *.css* contêm folhas de estilo, que definem a apresentação visual de elementos HTML, incluindo cores, layouts, fontes e outros estilos visuais da interface. [↑](#footnote-ref-9)
10. Os valores entre chavetas **{}** representam parâmetros variáveis que são substituídos conforme o contexto de utilização (por exemplo, nomes de cores, tamanhos ou intensidades). [↑](#footnote-ref-10)
11. Os elementos HTML como h1, h2, entre outros, utilizam por defeito os tamanhos de texto definidos nas variáveis. A aplicação destes tamanhos fora desses contextos ocorre apenas em casos pontuais. [↑](#footnote-ref-11)
12. Para referência adicional da sintaxe de Markdown, pode ser consultado o recurso disponível em [19]. [↑](#footnote-ref-12)