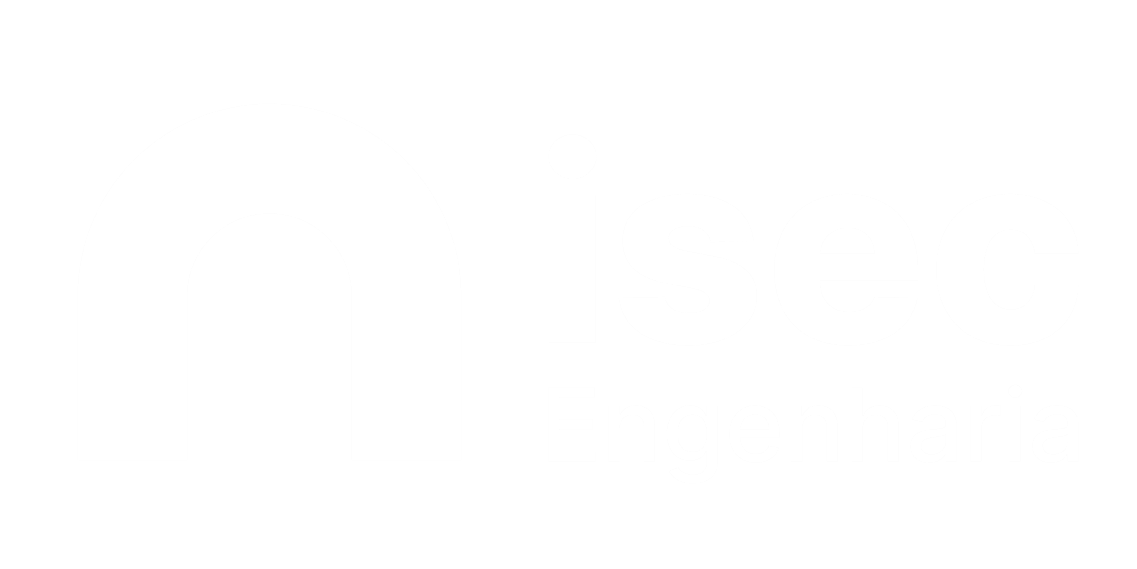
A picture containing text

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 de 2025

Índice

[Índice i](#_Toc208510758)

[Índice de figuras iii](#_Toc208510759)

[Lista de siglas e acrónimos v](#_Toc208510760)

[Agradecimentos 1](#_Toc208510761)

[Resumo 2](#_Toc208510762)

[1 Introdução 3](#_Toc208510763)

[1.1 Estrutura do relatório 3](#_Toc208510764)

[2 Estado Pré-existente do Projeto 5](#_Toc208510765)

[3 Planeamento e Conceção do Sistema 8](#_Toc208510766)

[3.1 Objetivos 8](#_Toc208510767)

[3.2 Arquitetura do sistema 12](#_Toc208510768)

[3.3 Tecnologias usadas 13](#_Toc208510769)

[4 Implementação da API 14](#_Toc208510770)

[4.1 Arquitetura da API 14](#_Toc208510771)

[4.1.1 Estrutura e Organização 14](#_Toc208510772)

[4.1.2 Modelo de Dados 16](#_Toc208510773)

[4.2 Implementação da API 18](#_Toc208510774)

[4.3 *Endpoints* e Funcionalidades 18](#_Toc208510775)

[4.4 Formato e Estrutura das Respostas 19](#_Toc208510776)

[4.5 Segurança e Autenticação 21](#_Toc208510777)

[4.6 Geração de Documentos 22](#_Toc208510778)

[4.7 Tarefas assíncronas 23](#_Toc208510779)

[5 Interface e Design visual 26](#_Toc208510780)

[5.1 Conceção visual do sistema 26](#_Toc208510781)

[5.1.1 Cores 26](#_Toc208510782)

[5.1.2 Tipografia 27](#_Toc208510783)

[5.1.3 Espaçamentos 27](#_Toc208510784)

[5.1.4 Componentes 27](#_Toc208510785)

[5.2 *Design* da Aplicação 28](#_Toc208510786)

[5.3 Arquitetura da Interface 28](#_Toc208510787)

[5.3.1 Organização do projeto 28](#_Toc208510788)

[5.4 Implementação da Interface 29](#_Toc208510789)

[5.4.1 Definição e organização dos estilos 29](#_Toc208510790)

[5.4.2 Responsividade 30](#_Toc208510791)

[5.4.3 Componentes reutilizáveis e flexibilidade 31](#_Toc208510792)

[5.4.4 Gestão de Sessões 33](#_Toc208510793)

[5.4.5 Navegação e gestão de rotas 34](#_Toc208510794)

[5.4.6 Estrutura modular das páginas 35](#_Toc208510795)

[5.4.7 Suporte para formatação de texto 36](#_Toc208510796)

[6 Interação com o Sistema 37](#_Toc208510797)

[6.1 Autenticação e Gestão de Sessões 37](#_Toc208510798)

[6.2 *Dashboard* do Docente 39](#_Toc208510799)

[6.3 Funcionalidades de Pesquisa e Filtragem 40](#_Toc208510800)

[6.4 Gestão de Docentes 41](#_Toc208510801)

[6.5 Gestão de Calendários 44](#_Toc208510802)

[6.5.1 Criação de um Calendário 45](#_Toc208510803)

[6.5.2 Visualização de um Calendário 45](#_Toc208510804)

[6.6 Submissão de uma Proposta 46](#_Toc208510805)

[6.7 Exportação da Proposta em PDF 50](#_Toc208510806)

[6.7.1 Personalização do *Template* 52](#_Toc208510807)

[7 Desenvolvimento Futuro 53](#_Toc208510808)

[7.1 Testes 55](#_Toc208510809)

[7.2 Melhorias e Otimizações 56](#_Toc208510810)

[8 Conclusão 57](#_Toc208510811)

[Referências bibliográficas 58](#_Toc208510812)

[Apêndices e Anexos I](#_Toc208510813)

[Apêndice A II](#_Toc208510814)

[Apêndice B III](#_Toc208510815)

[Apêndice C IV](#_Toc208510816)

[Anexo A V](#_Toc208510817)

Índice de figuras

[Figura 1 – Interface inicial do Sistema 7](#_Toc208510595)

[Figura 2 – Listagem de Funcionalidades 9](#_Toc208510596)

[Figura 3 – Arquitetura do Sistema 12](#_Toc208510597)

[Figura 4 – Arquitetura do Django 14](#_Toc208510598)

[Figura 5 – Diagrama Entidade-Relação 16](#_Toc208510599)

[Figura 6 – Classe *Accounts* 17](#_Toc208510600)

[Figura 7 – Classe Aluno 17](#_Toc208510601)

[Figura 8 – Estrutura padrão dos endpoints da API 18](#_Toc208510602)

[Figura 9 – Estrutura de um objeto em JSON 19](#_Toc208510603)

[Figura 10 – Resposta JSON com atributos de edição e eliminação 20](#_Toc208510604)

[Figura 11 – Códigos de resposta 21](#_Toc208510605)

[Figura 12 – Exemplo de *placeholders* no *template* e correspondência no código 22](#_Toc208510606)

[Figura 13 – Agendamento da tarefa principal 24](#_Toc208510607)

[Figura 14 – Tarefa assíncrona 25](#_Toc208510608)

[Figura 15 – Representação de uma cor com as suas nove variações 26](#_Toc208510609)

[Figura 16 – Estrutura do *UserContext* e *UserProvider* 33](#_Toc208510610)

[Figura 17 – Acesso à informação do utilizador 34](#_Toc208510611)

[Figura 18 – Estrutura das rotas no *App.js* 35](#_Toc208510612)

[Figura 19 – Estrutura *Markdown* 37](#_Toc208510613)

[Figura 20 – Página de *Login* 38](#_Toc208510614)

[Figura 21 – *Dashboard* do Administrador 38](#_Toc208510615)

[Figura 22 – Página inicial com as permissões padrão 39](#_Toc208510616)

[Figura 23 – Página inicial com todas as permissões 39](#_Toc208510617)

[Figura 24 – Lista completa sem filtros 40](#_Toc208510618)

[Figura 25 – Lista filtrada pelo “Tipo de Proposta” 40](#_Toc208510619)

[Figura 26 – Lista filtrada pelo “Curso” e pelo “Calendário” 41](#_Toc208510620)

[Figura 27 – Listagem de docentes 42](#_Toc208510621)

[Figura 28 – Visualização de um docente 42](#_Toc208510622)

[Figura 29 – Edição de um docente 43](#_Toc208510623)

[Figura 30 – Listagem com um docente removido 44](#_Toc208510624)

[Figura 31 – Listagem de Calendários 44](#_Toc208510625)

[Figura 32 – Formulário de criação de calendário 45](#_Toc208510626)

[Figura 33 – Visualização de um Calendário 46](#_Toc208510627)

[Figura 34 – Mensagem de erro do formulário 47](#_Toc208510628)

[Figura 35 – Formulário de submissão de uma Proposta 47](#_Toc208510629)

[Figura 36 – Seleção de curso, ramos e calendário no formulário de submissão 48](#_Toc208510630)

[Figura 37 – Criação de um Orientador no formulário de submissão 48](#_Toc208510631)

[Figura 38 – Página de Visualização de uma Proposta 49](#_Toc208510632)

[Figura 39 – Botão para geração do PDF 50](#_Toc208510633)

[Figura 40 – Documento PDF da proposta gerado 51](#_Toc208510634)

[Figura 41 – Estado final das funcionalidades 53](#_Toc208510635)

Lista de siglas e acrónimos

|  |  |
| --- | --- |
| API | *Application Programming Interface* |
| CSS | *Cascading Style Sheets* |
| HTML | *HyperText Markup Language* |
| HTTP | *Hypertext Transfer Protocol* |
| IDE | *Integrated Development Environment* |
| 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* |
| VSCode | *Visual Studio Code* |

Agradecimentos

Agradeço ao Professor César Páris, orientador do projeto, e ao Professor João Costa, responsável pela Comissão de Estágios da LEI e da LEI-PL, pela orientação, disponibilidade e apoio prestados ao longo do semestre, que foram essenciais para a concretização deste trabalho.

Resumo

Este relatório descreve o desenvolvimento de um Sistema de Informação para Gestão de Parcerias de Estágio e Projetos do ISEC (Instituto Superior de Engenharia de Coimbra), criado no âmbito da Licenciatura em Engenharia Informática. O projeto teve como objetivo desenvolver uma plataforma centralizada para gerir de forma eficiente as colaborações entre o Instituto e entidades externas no âmbito de Estágios e Projetos de investigação.

A solução implementada baseia-se numa arquitetura cliente-servidor, utilizando Django para o desenvolvimento da API (*Application Programming Interface*) e gestão de dados no backend, e React para a interface do utilizador no frontend. O sistema integra funcionalidades de autenticação JWT (*JavaScript Object Notation Web Token*), gestão de utilizadores com diferentes perfis (Administrador, Docente, Representante e Aluno), submissão e aprovação de propostas, geração automática de documentos PDF (*Portable Document Format*) e sistema avançado de filtragem de informação.

Durante o desenvolvimento foram aplicados princípios de arquitetura modular, que permitem a separação de responsabilidades e facilitam a manutenção e evolução do código. A API segue padrões RESTful, assegurando uma comunicação clara e consistente entre cliente e servidor, além de possibilitar a reutilização da mesma em diferentes interfaces.

Os resultados obtidos evidenciam que a solução desenvolvida pode vir a responder eficazmente às necessidades identificadas, proporcionando uma base sólida para a evolução futura do sistema e permitindo a sua escalabilidade para a gestão de colaborações institucionais em contextos mais amplos.

**Palavras-Chave:** Sistema de Informação, Gestão de Estágios, API RESTful, Django, React.

# Introdução

Este relatório descreve o trabalho desenvolvido no âmbito da Licenciatura em Engenharia Informática no ISEC. 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, sendo responsável pela Comissão de Estágios da LEI e da LEI-PL, atuou como cliente do sistema.

## Estrutura do relatório

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

* **Introdução:** contextualiza o projeto, apresentando o problema identificado, a motivação para o desenvolvimento do sistema e a estrutura do relatório.
* **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, 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.
* **Conclusão:** resume os resultados alcançados, avalia a adequação do sistema aos objetivos iniciais e reflete sobre os desafios enfrentados, lições aprendidas e contribuições do projeto.

# 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. Para além disso, a interface gráfica apresentava uma organização básica e pouco apelativa, sem grande preocupação com usabilidade. A navegação entre as páginas não era totalmente intuitiva, e faltavam elementos de *feedback* visual. 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.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 1 – Interface inicial do Sistema

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 2 – 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 3 – 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. 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 (*Integrated Development Environment*) distintos, cada um escolhido com base nas necessidades especificas do *frontend* e do *backend*.

As tecnologias e ferramentas utilizadas incluem:

* **VSCode (Visual Studio Code)** [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 concebida 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.

## Arquitetura da API

A API foi implementada utilizando o framework Django, que segue o padrão arquitetural MVT. Neste modelo, o *Model* define a estrutura dos dados e a forma como estes são armazenados na base de dados; o *View* contém a lógica de processamento e manipulação dos dados, recebendo pedidos do cliente e devolvendo respostas apropriadas; o Template é utilizado para gerar conteúdos dinâmicos, como documentos exportáveis ou páginas HTML, embora nesta aplicação a interface seja gerida separadamente pelo frontend em React.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 4 – Arquitetura do Django

### Estrutura e Organização

Um projeto Django é organizado de forma hierárquica, com uma aplicação principal que contém os ficheiros essenciais de configuração e gestão do projeto, e diversas aplicações adicionais que implementam funcionalidades específicas.

Em Django, uma aplicação representa um conjunto autónomo de funcionalidades que cumpre um determinado objetivo dentro do projeto. Cada aplicação possui os seus próprios modelos, *views*, *templates* e URLs, permitindo organizar o código de forma modular e reutilizável. Um projeto Django pode conter várias aplicações, que colaboram entre si para fornecer a totalidade da aplicação.

Na **aplicação principal**, os ficheiros típicos incluem:

* ***settings.py*** – define as configurações globais do projeto, como base de dados, aplicações instaladas, autenticação, variáveis de ambiente e definições de segurança.
* ***urls.py*** – estabelece os caminhos globais do projeto e a ligação a endpoints específicos de cada aplicação.
* **manage.py** – utilitário de linha de comando que permite executar tarefas administrativas, como migrações, criação de aplicações ou execução do servidor de desenvolvimento.

Dentro de cada aplicação Django, existem ficheiros que representam as camadas fundamentais do padrão MVT:

* ***models.py*** – define a estrutura dos dados e a forma como estes são armazenados na base de dados. Cada modelo representa uma tabela da base de dados, com atributos que correspondem a colunas e relações entre tabelas.
* ***views.py*** – contém as funções responsáveis por processar pedidos e gerar respostas. As *views* atuam como a lógica de negócio da aplicação, interpretando dados da base de dados, aplicando regras de negócio e devolvendo respostas.
* ***urls.py*** – define os caminhos específicos da aplicação, ligando cada URL a uma *view* correspondente.

Além disso, projetos Django podem incluir pastas como *templates* para geração de documentos ou páginas HTML.

No caso deste sistema, o projeto Django contém duas apps principais: gestor\_estagios, que define essencialmente as configurações e definições globais da aplicação, funcionando como a base do projeto; e api, que concentra toda a lógica funcional, incluindo modelos, *views*, URLs, templates e tarefas assíncronas, sendo responsável por fornecer as funcionalidades principais do sistema através de endpoints.

* ***api/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.
* ***api/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.
* ***api/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.

### 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 5 – 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 6 – 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 7 – 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*), ou seja, as operações sobre os dados podem ser feitas diretamente no código, sem a necessidade de escrever *queries* em SQL (*Structured Query Language*) manualmente.

A 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, 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.

## *Endpoints* e Funcionalidades

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 8 – 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.

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

## Formato e Estrutura das Respostas

Todas as respostas da API são devolvidas em formato JSON, um padrão leve e amplamente utilizado na troca de dados entre cliente e servidor. O JSON organiza a informação em pares de chave–valor e listas ordenadas, de forma simples e intuitiva [12]. Por ser independente da linguagem de programação e suportado pela maioria das tecnologias modernas, tornou-se o formato de eleição para aplicações web.

Além da sua versatilidade, o JSON combina duas vantagens importantes: é facilmente legível por humanos e, ao mesmo tempo, simples de interpretar por máquinas. Graças a estas características, o *frontend* pode utilizar diretamente os dados recebidos e apresentá-los na interface sem necessidade de transformações adicionais.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 9 – Estrutura de um objeto em JSON

As respostas JSON seguem uma estrutura clara e organizada, adaptada ao tipo de recurso solicitado. Para além da informação principal, podem incluir dados adicionais que orientam a forma como o utilizador interage com o recurso.

Um exemplo prático surge no caso das Propostas. Nem todos os utilizadores têm as mesmas permissões sobre um recurso: apenas o responsável pela conta da empresa ou o orientador da empresa associado a uma proposta pode editá-la ou eliminá-la.

Para tornar esta lógica visível no *frontend*, a resposta inclui atributos como *can\_edit* e *can\_delete*, que indicam se o utilizador autenticado possui essas permissões. Desta forma, a interface consegue adaptar-se dinamicamente, mostrando ou ocultando botões de edição e eliminação consoante o perfil do utilizador.

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Os conteúdos gerados por IA podem estar incorretos.

Figura 10 – Resposta JSON com atributos de edição e eliminação

Adicionalmente, cada resposta da é acompanhada de um código de HTTP[[4]](#footnote-4) que indica o resultado da operação. Estes códigos permitem ao cliente interpretar rapidamente se a ação foi bem-sucedida, se ocorreu algum erro ou se há necessidade de realizar ações corretivas.

Figura 11 – Códigos de resposta

|  |  |
| --- | --- |
| Código | Descrição |
| 200 | A pedido foi processada com sucesso e os dados resultantes devolvidos. |
| 201 | Um novo recurso foi criado com sucesso. |
| 204 | A operação foi concluída com sucesso, mas não há dados a devolver. |
| 400 | Dados fornecidos estão incorretos, incompletos ou mal formatados. |
| 401 | Token de autenticação ausente ou inválido. |
| 403 | O utilizador não tem permissão para aceder ao recurso. |
| 404 | O recurso solicitado não existe. |
| 500 | Erro inesperado ocorreu no servidor durante o processamento. |

## Segurança e Autenticação

A segurança da API é assegurada através da utilização de *tokens* JWT para autenticação. Sempre que um utilizador realiza o login com sucesso, a API gera um token JWT que é enviado de volta ao cliente. Este token contém informação codificada sobre o utilizador.

Todos os pedidos feitos à API devem incluir este *token* no cabeçalho de autorização. No momento de processamento de cada pedido, o *token* é decodificado, permitindo verificar a identidade do utilizador e determinar se possui permissão para executar a ação solicitada. Caso o *token* esteja ausente, inválido ou expirado, a API responde com o código de erro 401 (não autorizado).

O *token* tem uma duração definida de 1 dia, após a qual o utilizador deverá efetuar um novo login para obter um *token* válido. A chave de encriptação utilizada para gerar e validar os *tokens* é definida no ficheiro *.env*, garantindo que informação sensível permaneça protegida e separada do código-fonte.

## Geração de Documentos

A API disponibiliza a funcionalidade **F04 - Criação automática do documento da proposta**, 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 [13]; a **docx2pdf** realiza a conversão do ficheiro Word para PDF [14]; e a biblioteca **tempfile** gere a criação de ficheiros temporários, evitando armazenamentos desnecessários no servidor [15].

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 12 – 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.

Em termos técnicos, a exportação de propostas para PDF não apresenta limites de tamanho ou de número de páginas diretamente impostos pelo sistema, uma vez que o envio do ficheiro é realizado através de um *FileResponse* do Django, que lida de forma eficiente com documentos de qualquer dimensão.

No entanto, podem surgir restrições indiretas associadas a outros fatores:

* Dependência do *template* – O processo de geração do documento depende da correspondência exata entre os *placeholders* definidos no *template* e os campos configuradas no dicionário da API. Qualquer discrepância resulta em falhas no preenchimento do documento final.
* Conversão para PDF – A transformação do ficheiro Word em PDF é realizada com recurso à biblioteca docx2pdf, implicando a criação de ficheiros temporários no servidor. Eventuais falhas neste processo podem impedir a conclusão da exportação.
* Condições do servidor – Embora o sistema não imponha limites explícitos de tamanho, erros podem surgir em situações extremas, como falta de espaço em disco, memória insuficiente ou largura de banda reduzida, que afetem a criação e transmissão do ficheiro.

Todos os erros resultantes destes fatores são capturados pelo sistema, que devolve ao utilizador um HTTP 500 – Erro Interno do Servidor, garantindo que falhas inesperadas não interrompam de forma silenciosa o funcionamento da aplicação.

## Tarefas assíncronas

Para permitir a execução de tarefas em *background* de forma assíncrona, foi integrado o Celery na API. O Celery [16] 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 [17] 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 13 – 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 14 – 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 - Automação das atribuições** e **F08 - Preferência de orientadores**, 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 um 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 15 – 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. Estes componentes foram desenvolvidos de forma modular e reutilizável, podendo ser aplicados em diferentes páginas e contextos, adaptando-se aos diversos perfis de utilizador. Os componentes criados foram: Botões, Campos de formulários, Barra de Navegação, Cartões, Painéis, Etiquetas, Indicadores, Alertas e Tabelas.

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 funcionais distintos, cada um adaptado às necessidades específicas dos 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 a partilha de componentes comuns entre os diferentes fluxos. Esta 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 Apêndice B, onde estão descritas todas as decisões tomadas ao nível visual.

## Arquitetura 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 (*Cascading Style Sheets*) 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*.

## Implementação da Interface

Com a arquitetura da interface definida e a estrutura do projeto organizada, a implementação focou-se em transformar os componentes e páginas planeados em funcionalidades interativas e consistentes. O desenvolvimento do frontend aproveitou a modularidade do React para criar componentes reutilizáveis, garantir uma navegação intuitiva e centralizar a gestão de estado e sessão.

### 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 {}*[[8]](#footnote-8)*:

* **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[[9]](#footnote-9):** *--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 (*HyperText Markup Language*) 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*.

### Responsividade

A responsividade da interface foi assegurada através da utilização do **Bootstrap**, que fornece um conjunto de classes pré-definidas para organização do *layout* em diferentes resoluções de ecrã. O *frontend* faz uso das utilidades de *flexbox* disponibilizadas pela *framework*, permitindo controlar a orientação, o alinhamento e a distribuição dos elementos na página.

O ***flexbox*** é um modelo de layout do CSS que facilita a criação de estruturas dinâmicas e adaptáveis. Com esta estrutura, é possível organizar elementos na horizontal ou vertical, distribuir automaticamente os espaços livres entre eles e alinhar os conteúdos de forma consistente, mesmo em situações de redimensionamento da janela ou alteração do tamanho do ecrã.

Para garantir que o layout se adapta a diferentes dispositivos, foram utilizados os *breakpoints* do Bootstrap, que definem intervalos de largura do ecrã a partir dos quais o design se ajusta automaticamente:

* ***Extra small (xs):*** < 576px – geralmente usado para *smartphones*.
* ***Small (sm):*** ≥ 576px – *smartphones* maiores.
* ***Medium (md):*** ≥ 768px – tablets.
* ***Large (lg):*** ≥ 992px – portáteis.
* ***Extra large (xl):*** ≥ 1200px – *desktops* de maior dimensão.
* ***XXL (xxl):*** ≥ 1400px – monitores muito grandes.

Apesar desta solução fornecer uma base sólida de responsividade, a aplicação foi inicialmente concebida sobretudo para ecrãs de maior dimensão (computadores portáteis ou *desktops*). Assim, reconhece-se a necessidade futura de desenvolver um design dedicado a dispositivos móveis, de modo a otimizar a experiência de utilização em *smartphones* e tablets, melhorando a usabilidade em ecrãs mais pequenos.

### 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.

#### Botões

Os Botões são elementos fundamentais de interação na interface, permitindo ao utilizador executar ações como submeter formulários, navegar entre páginas ou confirmar operações. No sistema, foram definidos três tipos principais de botões — *PrimaryButton*, *SecondaryButton* e *TertiaryButton* — que seguem a mesma lógica de implementação, diferenciando-se apenas ao nível visual.

**Propriedades dos Botões:**

* ***content:*** define o conteúdo dentro do botão. Pode ser texto simples ou elementos mais complexos em HTML.
* ***small:*** altera o tamanho do botão, apresentando uma versão mais compacta.
* ***type:*** define o tipo do botão em HTML (*button*, *submit* ou *reset*), de acordo com a funcionalidade desejada [18].
* ***action:*** função a ser executada quando o botão é clicado. Garante a ligação entre a interface e a lógica do sistema.
* ***disabled:*** desativa o botão, impedindo cliques e indicando visualmente ao utilizador que a ação não está disponível.

#### Etiquetas (*Pills*)

As *Pills* são etiquetas coloridas que permitem identificar categorias ou atributos (ex.: “Estágio”, “Projeto”, “Presencial”). A cor pode ser definida automaticamente com base no tipo ou atribuída manualmente. Este componente suporta ainda a utilização de tooltips, permitindo fornecer informações adicionais quando o utilizador passa o cursor sobre a etiqueta.

**Propriedades das *Pills*:**

* ***type:*** define a categoria da etiqueta (e.g., Estágio, Projeto).
* ***text:*** texto exibido na etiqueta. Se não for indicado, é assumido o valor correspondente ao “*type*”.
* ***color:*** cor da etiqueta. Se não for válida, é usada a cor definida pelo “*type*”.
* ***collapse:*** se verdadeiro, mostra apenas a primeira letra do texto.
* ***tooltip:*** define o texto do *tooltip* (mostrado ao passar o rato sobre a etiqueta).
* ***tooltipPosition:*** posição do *tooltip* em relação à etiqueta.

#### Etiquetas (*State*)

O componente *State* é usado para representar graficamente o estado de uma Candidatura. Cada estado tem um ícone e texto correspondentes, permitindo uma rápida perceção visual da situação atual.

**Propriedades do *State*:**

* ***state:*** número que define o estado atual, mapeado para um ícone, cor e texto específicos (ex.: 1 = Pendente, 2 = Colocado).
* ***hideState:*** se verdadeiro, esconde a *label* “Estado:”.
* ***hideText:*** se verdadeiro, esconde o texto associado ao estado (mostrando apenas o ícone).
* ***tooltip:*** ativa um *tooltip* com o texto do estado.
* ***tooltipPosition:*** define a posição do *tooltip* em relação ao ícone.

#### Alertas

O componente *Alert* foi criado de forma genérica para abranger diferentes tipos de mensagens, adaptando automaticamente ícones, cores e dimensões conforme as propriedades recebidas.

**Propriedades do *Alert*:**

* ***text:*** define a mensagem exibida.
* ***type:*** indica o tipo de alerta (por exemplo: erro, sucesso, informação, etc.).
* ***small:*** apresenta o alerta numa versão mais compacta.

### Gestão de Sessões

A gestão de sessão no frontend foi implementada utilizando o *hook[[10]](#footnote-10)* ***useContext***, que permite partilhar os dados do utilizador por toda a aplicação de forma centralizada e reativa. O contexto é definido na pasta *contexts/* através do *UserProvider*, que encapsula toda a aplicação, fornecendo acesso ao estado do utilizador (*userInfo*) e à função para atualizar este estado (*setUserInfo*).

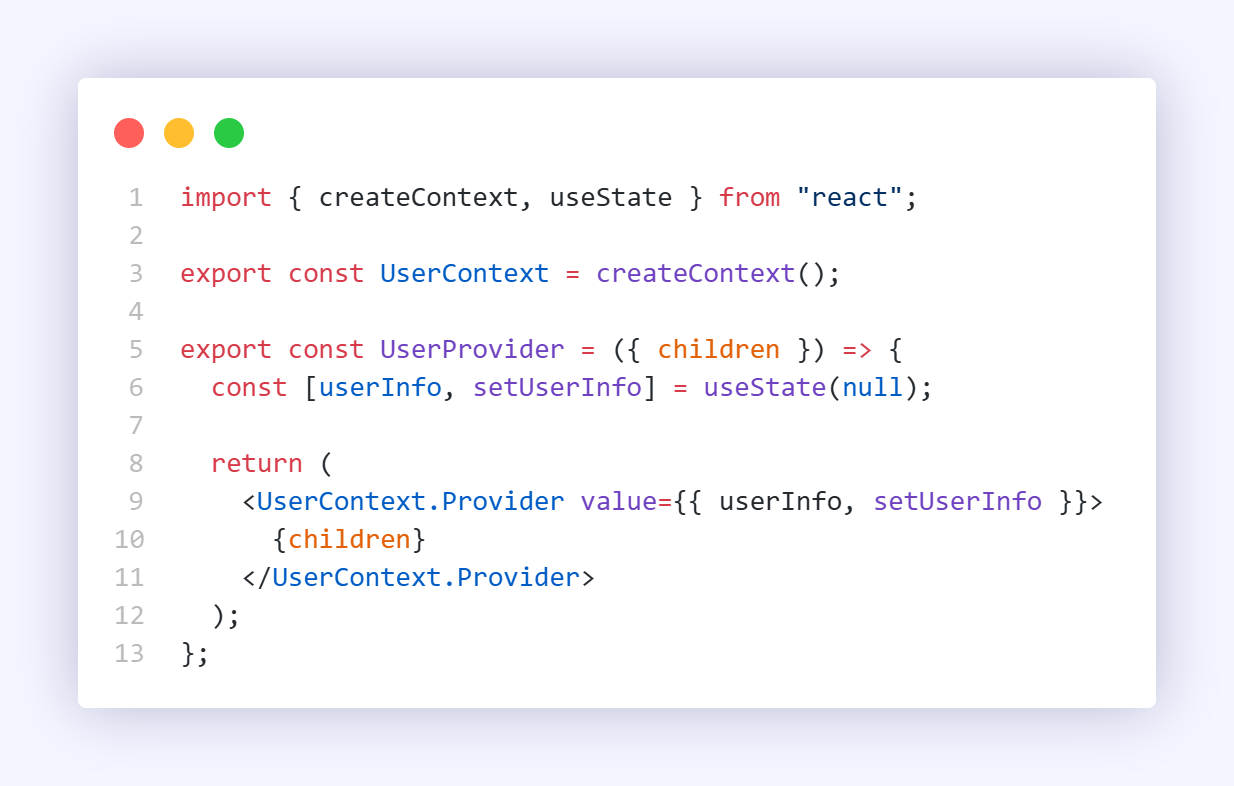


Figura 16 – Estrutura do *UserContext* e *UserProvider*

Dentro de cada componente que necessita de informação sobre o utilizador, *useContext* é utilizado para aceder aos dados armazenados no contexto, garantindo que a informação é sempre atualizada e consistente em toda a interface:

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 17 – Acesso à informação do utilizador

O processo de login preenche o contexto com os dados do utilizador após a autenticação bem-sucedida. Estes dados são também armazenados no ***localStorage***, que é uma funcionalidade do navegador que permite guardar informações de forma persistente no lado do cliente, mesmo após o utilizador fechar ou atualizar a página. Isto garante que os dados essenciais, como *tokens* de autenticação e informações de sessão, permanecem disponíveis entre sessões do utilizador, melhorando a experiência de utilização e evitando a necessidade de novo login a cada visita.

### 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.

Uma rota simples em React Router define a correspondência direta entre um caminho da aplicação e o componente que deve ser apresentado. Por exemplo, ao configurar uma rota com *path="/dashboard"* e *element={<Dashboard />}*, o sistema assegura que sempre que o utilizador acede ao endereço */dashboard*, o componente *Dashboard* é exibido. Este mecanismo básico permite mapear URLs para componentes de forma clara e organizada.

No caso da aplicação, algumas rotas utilizam uma estrutura hierárquica para agrupar páginas relacionadas. Como exemplificado na **Figura 18**, o caminho principal */company* centraliza todas as rotas relacionadas com o módulo de Empresas, incluindo */list*, */view* e */edit*. Esta organização em árvore melhora a legibilidade e manutenção do código e facilita a escalabilidade do sistema, permitindo adicionar novas páginas sem comprometer a clareza da navegação.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 18 – Estrutura das rotas no *App.js*

### 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* [[11]](#footnote-11) e *.css [[12]](#footnote-12)* 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.

### Suporte para formatação de texto

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 [19], 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[[13]](#footnote-13).

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 19 – Estrutura *Markdown*

# Interação com o Sistema

De entre os diversos fluxos implementados durante o desenvolvimento, apresentam-se de seguida 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 e Gestão de Sessões

O processo de autenticação de um utilizador segue o procedimento habitual, no qual são solicitadas as credenciais (email e palavra-passe). As credenciais 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* (Figura 21), 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 20 – Página de *Login*

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 21 – *Dashboard* do Administrador

## *Dashboard* 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 22 – 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 23 – Página inicial com todas as permissões

## Funcionalidades de Pesquisa e Filtragem

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, navegar pelas diversas listas seria complexo e demorado.

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. Uma das características mais importantes é a possibilidade de combinar múltiplos filtros, permitindo ao utilizador restringir os resultados com base em diferentes parâmetros simultaneamente, refinando significativamente a informação apresentada.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 24 – Lista completa sem filtros

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 25 – Lista filtrada pelo “Tipo de Proposta”

Uma imagem com texto, Tipo de letra, número, file

Os conteúdos gerados por IA podem estar incorretos.

Figura 26 – Lista filtrada pelo “Curso” e pelo “Calendário”

Como exemplificado na **Figura 24**, é possível visualizar a lista completa sem qualquer filtro aplicado. Ao utilizar o *dropdown* “Tipo” para filtrar pelo tipo de proposta, a lista é automaticamente atualizada para mostrar apenas as propostas correspondentes, neste caso “Projeto”, como ilustrado na **Figura 25**. A combinação de filtros, por exemplo “Curso” e “Calendário”, permite uma filtragem ainda mais precisa, restringindo os resultados apenas às propostas de Licenciatura em Engenharia Informática e aos calendários de 2025/2026, conforme mostrado na **Figura 26**.

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, software

Os conteúdos gerados por IA podem estar incorretos.

Figura 27 – 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 28 – 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

Os conteúdos gerados por IA podem estar incorretos.

Figura 29 – 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 30 – Listagem com um docente removido

Os docentes removidos aparecem na lista com um fundo escuro, como se pode ver na Figura 30.

## Gestão de Calendários

A gestão de calendários no sistema permite 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 31 – 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, recibo

Os conteúdos gerados por IA podem estar incorretos.

Figura 32 – 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 33 – 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.

Uma imagem com texto, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Figura 34 – Mensagem de erro do formulário

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 35 – 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 36 – 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 37 – Criação de um Orientador no formulário de submissão

Após a submissão de uma nova proposta, o sistema redireciona o utilizador para a página de visualização da mesma. Nesta página, é possível confirmar os dados introduzidos, verificar o estado atual da proposta e aceder a funcionalidades adicionais, como a edição (quando disponível) ou o acompanhamento do processo de validação pelo administrador.

Nesta página, é possível ver todos os dados da proposta e aceder a funcionalidades adicionais, como a edição (caso ainda não tenha sido divulgada aos alunos), descarregar o ficheiro da proposta em formato PDF e acompanhar a ocupação das vagas disponibilizadas.

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

Os conteúdos gerados por IA podem estar incorretos.

Figura 38 – Página de Visualização de uma Proposta

## Exportação da Proposta em PDF

O sistema disponibiliza a funcionalidade **F04 - Criação automática do documento da proposta**, 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, preenchendo o *template* e gerando um ficheiro Word temporário.
3. O Word é convertido num ficheiro PDF e devolvido ao cliente.
4. O cliente 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 39 – 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 40 – Documento PDF da proposta gerado

### Personalização do *Template*

O *template* em formato .docx pode ser editado como um documento Word normal, permitindo ajustar livremente o estilo, a estrutura e o texto descritivo. Para além do conteúdo fixo, o documento contém chaves de substituição, que seguem a sintaxe *{{nome\_do\_conteudo}}*. Estas chaves funcionam como marcadores que são preenchidos automaticamente com a informação proveniente da API no momento da geração do documento.

É importante salientar que o nome da chave no *template* deve coincidir exatamente com a chave definida no dicionário no código da API. Por exemplo, se no código a chave definida for “curso”, então o *template* deve conter o marcador *{{curso}}*.

O sistema suporta ainda condições lógicas dentro do *template*, permitindo que determinadas secções sejam impressas apenas se a informação existir. Isto evita, por exemplo, que sejam exibidos campos vazios ou informações irrelevantes numa proposta que não possua determinados atributos.

A explicação detalhada sobre o processo de formatação, bibliotecas utilizadas e a forma como o dicionário de substituição é definido pode ser encontrada no capitulo **4.6 Geração de Documentos**, onde esta funcionalidade é descrita em maior profundidade.

# 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 41 – 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 41 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 Apêndice C.

## Testes

O desenvolvimento das funcionalidades principais consumiu a totalidade do tempo disponível no âmbito académico, não sendo possível realizar a fase de testes com utilizadores finais. Foi, contudo, desenvolvido um plano estruturado para implementar sessões de teste com empresas e alunos, de forma a validar os principais fluxos da aplicação em condições reais de utilização.

Estes testes estão programados para ocorrer no primeiro semestre do ano letivo 2025/2026. A escolha deste período deve-se ao facto de existir um menor volume de alunos em processo de estágio, o que permitirá realizar a validação sem causar impacto direto nos processos académicos. Assim, caso o sistema revele instabilidade ou problemas graves, será sempre possível regressar ao processo tradicional sem riscos de atrasos ou prejuízos.

Os processos a testar incluíam , por ordem de implementação:

* Registo de Empresas e Representantes.
* Registo de Alunos.
* Submissão de Propostas.
* Divulgação de Propostas aos Alunos.

A estratégia de testes contemplaria múltiplas dimensões: testes de usabilidade para validar a experiência do utilizador, testes de carga para avaliar o comportamento do sistema sob *stress*, e testes de aceitação pelos utilizadores finais para confirmar que os requisitos funcionais foram adequadamente implementados.

O objetivo seria avaliar o desempenho do sistema com múltiplos utilizadores em simultâneo, possibilitando verificar se a aplicação mantém a performance esperada. Isso permitiria identificar necessidades de otimização das listagens, introdução de mecanismos de *cache* ou outras melhorias técnicas, garantindo maior escalabilidade e eficiência no futuro.

## Melhorias e Otimizações

Embora o sistema desenvolvido constitua uma base sólida, preparada para suportar todas as funcionalidades planeadas, algumas partes centrais ficaram por implementar para que a aplicação se torne totalmente funcional. A arquitetura e as estruturas definidas ao longo do projeto permitem, no entanto, que estas funcionalidades possam ser adicionadas de forma consistente e integrada em fases futuras de desenvolvimento.

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

Este projeto teve como objetivo o desenvolvimento de uma aplicação de um Sistema de Informação para Gestão de Parcerias de Estágio e Projetos do ISEC, procurando facilitar a comunicação, a organização e o acompanhamento das diferentes parcerias.

Os objetivos inicialmente definidos não foram alcançados na totalidade, mas o sistema resultante já permite centralizar os processos de gestão das propostas e assegura várias funcionalidades administrativas relevantes. Apesar de algumas funcionalidades mais centrais, como a submissão de candidaturas, permanecerem por implementar, foi criada uma base sólida e uma arquitetura preparada para suportar de forma consistente a expansão futura do sistema, garantindo que todas as necessidades previstas possam vir a ser integradas.

Durante o desenvolvimento, surgiram diversos desafios que exigiram especial atenção. A compreensão e implementação das regras de negócio associadas à gestão de estágios revelou-se particularmente complexa, devido à necessidade de mapear corretamente os diferentes intervenientes e respetivos papéis no processo. Paralelamente, a adoção da framework Django constituiu uma dificuldade acrescida, dado que se tratava de uma tecnologia ainda não explorada anteriormente.

O impacto esperado para o ISEC é significativo, uma vez que a aplicação permitirá reduzir o esforço administrativo, centralizar processos e oferecer uma experiência mais clara e eficiente para alunos, docentes e empresas. Para além disso, o sistema tem potencial para ser reutilizado e adaptado em contextos distintos, como, por exemplo, a gestão de projetos de investigação ou outros processos colaborativos.

Em reflexão, este projeto representou um desafio enriquecedor, permitindo trabalhar em todas as fases do desenvolvimento de um sistema — desde o *design* até ao produto final. Apesar das dificuldades enfrentadas, o resultado alcançado representa um trabalho consistente, que contribuiu significativamente não apenas para o desenvolvimento técnico, mas também para a capacidade de estruturar e implementar uma solução complexa de forma integrada.

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] | MDN contributors, “Working with JSON,” 18 agosto 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn\_web\_development/Core/Scripting/JSON. [Acedido em 11 setembro 2025]. |
| [13] | E. Lapouyade, “Welcome to python-docx-template’s documentation!,” 2015. [Online]. Available: https://docxtpl.readthedocs.io/en/latest. [Acedido em 03 setembro 2025]. |
| [14] | A. Johri, “docx2pdf · PyPI,” 11 dezembro 2021. [Online]. Available: https://pypi.org/project/docx2pdf. [Acedido em 03 setembro 2025]. |
| [15] | 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]. |
| [16] | 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]. |
| [17] | Redis, “Redis - The Real-time Data Platform,” [Online]. Available: https://redis.io. [Acedido em 02 setembro 2025]. |
| [18] | Refsnes Data, “HTML <button> type Attribute,” 1999. [Online]. Available: https://www.w3schools.com/tags/att\_button\_type.asp. [Acedido em 08 setembro 2025]. |
| [19] | M. Cone, “Markdown Guide,” 2025. [Online]. Available: www.markdownguide.org. [Acedido em 26 agosto 2025]. |
| [20] | L. Gupta, “HTTP Status Codes,” 9 agosto 2024. [Online]. Available: https://restfulapi.net/http-status-codes. [Acedido em 20 julho 2025]. |
| [21] | “Markdown Syntax Cheatsheet,” 2021. [Online]. Available: https://www.markdown-cheatsheet.com. [Acedido em 26 agosto 2025]. |
| [22] | Bloomreach, Inc, “Basic Syntax of Jinja,” 2021. [Online]. Available: https://documentation.bloomreach.com/engagement/docs/jinja-syntax. [Acedido em 03 setembro 2025]. |
| [23] | Meta Platforms, Inc., “Introducing Hooks – React,” 2025 . [Online]. Available: https://legacy.reactjs.org/docs/hooks-intro.html?utm\_source=chatgpt.com. [Acedido em 08 setembro 2025]. |

Apêndices e Anexos

[Apêndice A II](#_Toc208501517)

[Apêndice B III](#_Toc208501518)

[Apêndice C IV](#_Toc208501519)

[Anexo A V](#_Toc208501520)

Apêndice A

Apêndice B

Apêndice C

Anexo A

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 [20]. [↑](#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 [22]. [↑](#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. 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-8)
9. 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-9)
10. Um *Hook* em React é uma função especial que permite “ligar” funcionalidades do React a componentes funcionais [23]. [↑](#footnote-ref-10)
11. 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-11)
12. 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-12)
13. Para referência adicional da sintaxe de Markdown, pode ser consultado o recurso disponível em [21]. [↑](#footnote-ref-13)