



UNIVERSIDADE
LUSÓFONA

Site DEISI

Trabalho Final de Curso

Relatório Final

Eduardo Filipe Fernandes Miranda

Prof. Pedro Alves

Trabalho Final de Curso | LEI | 24/04/2022

Direitos de cópia

Site DEISI, Copyright de Eduardo Filipe Fernandes Miranda, ULHT.

A Escola de Comunicação, Arquitetura, Artes e Tecnologias da Informação (ECATI) e a Universidade Lusófona de Humanidades e Tecnologias (ULHT) têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Resumo

A *Internet* é, cada vez mais, um dos meios de comunicação com maior impacto a nível mundial. As instituições recorrem cada vez mais à *Internet* para comunicarem com o público e fornecerem informações e serviços.

O aparecimento da pandemia COVID-19 voltou a mostrar que esta é uma ferramenta essencial na vida da população, pois, através desta, foi possível que um quinto dos trabalhadores portugueses pudessem trabalhar remotamente e que milhares de alunos finalistas do secundário pudessem recolher toda a informação necessária para se candidatarem e ingressarem no ensino superior. Neste contexto, os *websites* institucionais das faculdades, como o *website* do DEISI, têm um papel crucial na partilha de informação relativa à oferta formativa com possíveis interessados em se candidatarem às instituições, pelo que necessitam constantemente de manter a sua informação atualizada e, periodicamente, de sofrer algumas alterações, oferecendo assim uma melhor experiência aos utilizadores.

Este trabalho tem como objetivo, após diversas análises, reformular o *website* do DEISI e respectiva arquitetura de navegação, de modo a poder oferecer aos utilizadores as informações mais recentes sobre o departamento e respetivos cursos e uma melhor experiência, através da criação e inclusão de novos elementos, como páginas dinâmicas e ligações a API's externas.

Palavras-chave: *Front-End, Back-End, Website, Content Management Services*

Abstract

The internet is increasingly one of the means of communication with the greatest impact worldwide. Companies are using the internet to communicate with the public and provide information and services more and more.

The emergence of the COVID-19 pandemic has once again shown that this is an essential tool in the lives of the population, as it made it possible for a fifth of Portuguese workers to work remotely and for thousands of high school finalists to collect all the information required to apply and enter college. In this context, the universities' institutional websites, such as the DEISI website, play a crucial role in sharing information regarding the training offer with potential interested parties in applying to institutions, so they constantly need to keep their information updated and, periodically, to undergo some changes, thus offering a better experience to users.

This work aims, after several analyses, to reformulate the DEISI website and its navigation architecture, in order to offer users the most recent information about the department and its courses and a better experience, through the creation and inclusion of new elements such as dynamic pages and links to external API's.

Keywords: Front-End, Back-End, Website, Content Management Services

Índice

Resumo	iii
Abstract	iv
Índice.....	v
Lista de Figuras	vi
1. Identificação do Problema	1
1.1. Contexto	1
1.2. Objetivos.....	1
1.3. Estrutura do Trabalho	2
2. Levantamento de Requisitos	4
3. Viabilidade e Pertinência.....	8
4. Solução Desenvolvida.....	9
4.1. Disciplinas e Áreas Científicas	12
4.2. Protótipo Inicial.....	12
4.3. Front-End	17
4.4. Back-End	22
4.5. Solução Final	27
5. Benchmarking	31
A. FCT-UNL	31
B. IST	32
C. FCUL.....	33
D. FEUP	34
E. ULHT	35
5.1. Análise Global:	36
6. Método e Planeamento	38
7. Resultados.....	40
8. Conclusão.....	43
Bibliografia	44
Glossário	45

Lista de Figuras

Fig. 1 - Homepage Website Atual DEISI	1
Fig. 2 - Representação do Funcionamento do DOM Virtual.....	9
Fig. 3 - Funcionamento da Framework Spring MVC.....	10
Fig. 4 – Arquitetura Da Aplicação	11
Fig. 5 – Diagrama de ER das Entidades do Website	11
Fig. 6 – Landing Page Mockup.....	13
Fig. 7 – Landing Page Mockup Dispositivos Móveis	13
Fig. 8 – Landing Page Protótipo Inicial	14
Fig. 9 – Landing Page Protótipo Inicial Com Menu Visível.....	14
Fig. 10 – Página Métricas Protótipo Inicial	15
Fig. 11 – Página Estudantes Protótipo Inicial	15
Fig. 12 – Página Empresas Protótipo Inicial.....	16
Fig. 13 – Página Docentes Protótipo Inicial	16
Fig. 14 – Estrutura da Aplicação Front-End	17
Fig. 15 - Ficheiro index.jsx	18
Fig. 16 – Ficheiro App.jsx.....	19
Fig. 17 – Ficheiro api_request.js.....	19
Fig. 18 – Ficheiro StudentsPage.jsx	20
Fig. 19 – Ficheiro Carousel.jsx	21
Fig. 20 – Ficheiro UserCard.jsx	21
Fig. 21 – Estrutura da Aplicação Back-End	22
Fig. 22 – Lista de Alunos Existentes na Base de Dados	24
Fig. 23 – Formulário de Submissão de Novos Alunos	24
Fig. 24 – Endpoint da Lista de Alunos	25
Fig. 25 – Algumas Funções da Classe StudentsController.kt	25
Fig. 26 – Função de Listagem de Alunos da Classe StudentsAPIController.kt	26
Fig. 27 – Função de Upload das Imagens na CDN da Classe StudentsController.kt	26
Fig. 28 – Landing Page da Solução Final	27
Fig. 29 – Landing Page da Solução Final c/ Menu	27
Fig. 30 – Página Métricas da Solução Final.....	28
Fig. 31 - Página de Alunos da Solução Final	28
Fig. 32 – Página de Empresas da Solução Final	29

Fig. 33 – Página de Docentes da Solução Final.....	29
Fig. 34 – Página de Cursos da Solução Final	30
Fig. 35 - Homepage Website Departamento Informática FCT-UNL.....	31
Fig. 36 - Mapa Website Departamento Informática FCT-UNL (3 Níveis)	32
Fig. 37 - Homepage Website Departamento Informática Técnico.....	32
Fig. 38 - Mapa Website Departamento Informática IST (3 Níveis)	33
Fig. 39 - Homepage Website Departamento Informática FCUL	33
Fig. 40 - Mapa Website Departamento Informática FCUL (3 Níveis).....	34
Fig. 41 - Homepage Website Departamento Informática FEUP	34
Fig. 42 - Mapa Website Departamento Informática FEUP (3 Níveis).....	35
Fig. 43 - Homepage Website Atual DEISI.....	35
Fig. 44 - Mapa Website Departamento Informática ULHT (3 Níveis)	36
Fig. 45 - Calendário de Execução.....	38

1. Identificação do Problema

1.1. Contexto

O website do DEISI^[INULHT21] é a principal representação on-line do Departamento de Engenharia Informática e Sistemas de Informação. Sendo um website que foi desenvolvido há alguns anos, apresenta um problema crucial resultante da sua idade: um *design* antiquado. O *design* do website, apesar de simples, é um *design* bastante antigo, pelo que este problema é crucial na experiência dos utilizadores. Um *design* atualizado que combine algumas das tendências atuais poderá melhorar substancialmente a experiência dos utilizadores.



Fig. 1 - Homepage Website Atual DEISI

Outro problema identificado é a falta de atualização da informação existente no website. Após uma pesquisa feita no mesmo, foi possível concluir que a última atualização à informação foi realizada em 2019. É crucial para os utilizadores que a informação sobre o departamento e respetivos cursos esteja sempre atualizada. Dentro deste contexto, é possível identificar outro problema com bastante impacto nos utilizadores, o tipo de informação que é transmitida. Algumas informações nele presentes são informações com maior relevância para alunos ou docentes da instituição, e não para utilizadores externos, podendo ser um factor de grande influência no interesse dos utilizadores.

1.2. Objetivos

Tendo em conta o problema apresentado e a proposta inicial de resolução do mesmo, o website do DEISI^[INULHT21] desenvolvido visa oferecer aos utilizadores uma melhor e mais clara experiência, com as informações mais recentes relativas ao departamento e respetivos cursos, permitindo, simultaneamente, que o conteúdo seja atualizado com simplicidade e

regularidade. Algumas funcionalidades definidas após a análise do problema não foram implementadas como responsividade do *website* ou a *SEO (Search Engine Optimization)*, no entanto, todas estas funcionalidades irão ser implementadas após o contexto de avaliação do TFC, bem como novas funcionalidades identificadas ao longo do seu desenvolvimento, que irão favorecer o website, reforçando a solução aos problemas identificados.

1.3. Estrutura do Trabalho

Este trabalho encontra-se estruturado em sete capítulos, sendo este capítulo o primeiro, “Introdução”, no qual é feita uma exposição do problema e são descritos os objetivos para a resolução do mesmo, bem como uma análise comparativa da solução desenvolvida face à proposta inicial.

No segundo capítulo, “Levantamento de Requisitos”, são indicadas as características da solução produzida, bem como uma descrição detalhada de cada uma. São também identificadas as características adicionadas após o levantamento inicial, bem como as que foram implementadas e as que irão ser implementadas na etapa seguinte.

No terceiro capítulo, “Viabilidade e Pertinência”, é demonstrada a viabilidade e relevância deste projeto, esclarecendo-se que se trata de um projeto com um impacto positivo e que este contribui para a resolução do problema identificado no presente capítulo. São também neste capítulo justificados os baixos custos de desenvolvimento deste projeto, custos já existentes do projeto anterior e as características que apresenta para não se esgotar enquanto projeto académico, podendo ser continuado após a conclusão deste TFC.

No quarto capítulo, “Solução Desenvolvida”, são introduzidas as principais tecnologias utilizadas na realização deste trabalho (*React.js*[REACT21] e *Spring MVC*[SPMVC21]) bem como as suas principais funcionalidades. São também introduzidas bibliotecas adicionais a utilizar durante o desenvolvimento do trabalho. De seguida, são apresentadas as disciplinas e áreas científicas do curso que são aplicadas na solução proposta, bem como um protótipo inicial da mesma. Por fim, é apresentada a solução desenvolvida com base nos requisitos identificados no capítulo anterior e no protótipo inicial, dividida em dois sub-capítulos, “Front-end” e “Back-end”, nos quais são explicadas as arquiteturas, o modelo de dados, as técnicas utilizadas e são apresentados ecrãs exemplo.

No quinto capítulo, “Benchmarking”, é feita uma apresentação da análise comparativa da solução proposta face a outros *websites* de outros departamentos de informática de diversas faculdades. Neste capítulo, é feita uma análise individual a cada *website*, incluindo o *website* atual do DEISI, seguido de uma análise global dos problemas identificados em todos.

No sexto capítulo, “Plano de Testes e Validação”, é apresentada uma lista de sugestões de testes que visam demonstrar o funcionamento da solução bem como a aplicabilidade dos requisitos definidos.

No sétimo capítulo, “Calendário”, é proposto, em formato *Gantt*, um plano de trabalho para o desenvolvimento deste TFC.

2. Levantamento de Requisitos

Como foi referido no capítulo anterior, o objetivo deste projeto é desenvolver um novo *website* para o Departamento de Engenharia Informática e Sistemas de Informação com uma interface e conteúdos novos e atualizados, de modo a proporcionar a utilizadores externos uma boa experiência com a plataforma e a captar a sua atenção e interesse no departamento e respectivos cursos. Desta forma, foi feito um levantamento de requisitos junto de alguns professores do DEISI, de modo a melhor cumprir os objetivos inicialmente definidos.

Para o novo website do DEISI, foram definidos os seguintes requisitos:

- Criação de 6 páginas principais (*landing page*, página com métricas do curso, página de alunos da faculdade, página de empresas, página de docentes e página de cursos).
- Criação de um menu simples e capaz de acomodar futuras adições.
- Aplicação do efeito de *scroll “full page”*.
- Inclusão de uma animação na *landing page*.
- Adição de estatísticas às páginas do *website*, alusivas ao contexto onde se encontram, à exceção da *landing page*.
- Adição de carrosséis às páginas de alunos, docentes e empresas, com perfis de alunos, docentes e logotipos de empresas, respetivamente.
- Adição de um botão a cada perfil de aluno e docente, permitindo a visualização de detalhes sobre o mesmo.
- Adição de um carrossel à página de cursos com informações sobre os mesmos.
- Visualização de dados provenientes do *back-end* via *web-services*
 - Dados dos cabeçalhos das páginas (título, estatística e textos)
 - Dados dos carrosséis (perfis de alunos e docentes, dados das empresas e cursos do departamento)
 - Dados do gráfico da página de métricas
- Automatização do processo de submissão das fotografias dos carrosséis (fotografias dos perfis de alunos e docentes e logotipos das empresas).
- Adição de uma *cache* ao servidor

Para o primeiro requisito definido, de forma a que os utilizadores tenham a melhor experiência possível com o *website* e toda a informação seja apresentada de uma maneira simples e clara, este foi dividido em 6 páginas principais:

1. a *landing page*, a página inicial do website, constituída por uma animação, o título do departamento e um botão que facilita a navegação.
2. a página de métricas sobre o departamento e respetivos cursos, constituída pelo título da página, por um pequeno parágrafo referente ao departamento, duas estatísticas

referentes ao número de formados pelos cursos do departamento e um gráfico também alusivo ao número de formados pelos cursos do departamento.

3. a página de alunos, constituída pelo título da mesma, duas estatísticas referentes aos alunos dos cursos do departamento, um pequeno parágrafo referente aos alunos e um carrossel, constituído por perfis de diversos alunos que frequentaram os cursos do departamento.
4. a página de empresas, constituída pelo título da mesma, duas estatísticas referentes à empregabilidade e salário mínimo médio dos cursos do departamento, um pequeno parágrafo referente às empresas onde os alunos dos diversos cursos costumam ingressar depois de terminarem o curso e um carrossel com os diferentes logotipos.
5. a página de docentes, à semelhança da página de alunos, constituída pelo título da mesma, duas estatísticas referentes aos docentes dos cursos do departamento, um pequeno parágrafo referente aos docentes e um carrossel, constituído por perfis de diversos docentes que lecionam nos cursos do departamento.
6. a página de cursos, à semelhança da página de alunos, docentes e empresas, constituída pelo título da mesma, o número de cursos existentes no departamento, um pequeno parágrafo referente aos cursos e um carrossel, constituído por cartões informativos sobre os diversos cursos, cartões estes que contém as seguintes informações:
 - a. título do curso
 - b. grau do curso
 - c. pequeno texto informativo sobre o curso
 - d. botão que direciona o utilizador para a página oficial da ULHT sobre o curso

Para o segundo requisito, de forma a simplificar a navegação do *website*, foi implementado um menu, capaz de ser utilizado em cada página e que, ao ser utilizado, não oculte o conteúdo da mesma. Este menu contém os *links* para todas as outras páginas do *website* e é capaz de acomodar futuras opções.

Para o terceiro requisito, de modo a que, novamente, a navegação do *website* seja simplificada, foi implementado o efeito de *scroll “full scroll”*. Este efeito permite aos utilizadores navegarem entre páginas com um efeito “*full-screen*”, isto é, de modo a que as páginas sejam demonstradas por completo, sem que o seu conteúdo seja dividido ao fazer *scroll*.

Para o quarto requisito, de modo a cativar os utilizadores quando entram no novo *website* do departamento, foi implementada uma animação na *landing page*.

Para o quinto requisito, de forma a transmitir informações importantes aos utilizadores externos de uma forma simples e clara, têm que ser adicionadas a cada página estatísticas alusivas à página onde se encontra. Isto é, na prática:

- na página de métricas relativas ao departamento, as estatísticas adicionadas são alusivas ao número de formados pelos cursos do departamento e ao número de inscritos no ano passado.
- na página de alunos, as estatísticas a adicionar podem ser alusivas, por exemplo, à satisfação dos alunos com os cursos do departamento.
- na página de empresas, as estatísticas adicionadas são alusivas à empregabilidade e ao salário médio inicial.
- na página de docentes, as estatísticas adicionadas são alusivas ao grau de educação dos docentes e do seu contacto com o mundo empresarial.
- na página de cursos, não foi adicionada nenhuma estatística, mas sim o número de cursos existentes no departamento.

Para o sexto requisito, de forma a dar a conhecer a utilizadores externos diversos alunos que já frequentaram os cursos do departamento e os docentes dos mesmos, foi implementado um carrossel nas respetivas páginas, com diversos perfis de alunos e docentes e seus detalhes. No caso dos alunos, os detalhes são relativos ao curso que frequentaram, à empresa onde trabalham e qual o cargo que desempenham. No caso dos docentes, os detalhes deverão ser relativos ao percurso académico, às cadeiras que lecionam e a experiência profissional passada. No caso das empresas, para dar a conhecer as mesmas a utilizadores externos, a sua página deve conter um carrossel com os logotipos das empresas referidas, bem como uma pequena descrição da mesma.

Para o sétimo requisito, de modo a que os detalhes dos alunos e dos docentes sejam facilmente acedidos e ocultos conforme a necessidade dos utilizadores, foi adicionado, a cada perfil do carrossel, um botão que permita visualizar e ocultar esses detalhes.

Para o oitavo requisito, de modo a dar a conhecer os diversos cursos existentes no departamento, foi implementado um carrossel constituído por cartões informativos sobre os diversos cursos, como foi referido no ponto 6º do primeiro requisito. Este carrossel apresenta um comportamento oposto ao dos carrosséis criados anteriormente, de modo a que a interação com o utilizador seja superior, isto é, contrariamente aos carrosséis anteriormente implementados, a informação deste não altera automaticamente, para o utilizador poder consultar a informação completa necessita de interagir com os seus controlos.

Para o nono requisito, de modo a que toda a informação do *website* possa ser controlada através do *back-end*, podendo assim ser atualizada regularmente e com simplicidade, foi desenvolvida uma *RESTful API* (formato *JSON*) com múltiplos *endpoints* que fornecem informação a diferentes componentes do *website*, tais como todos os carrosséis. Irão também ser desenvolvidos outros *endpoints* para os cabeçalhos das páginas e para o gráfico na página das métricas, como referido nos pontos 2 e 3 do mesmo.

Para o décimo requisito, de forma a simplificar o processo de criação ou atualização de elementos para os carrosséis constituídos por imagens, irá ser implementado, no formulário de edição/criação de novos elementos, um sistema de upload de imagens que automaticamente faça upload destas para a CDN onde se encontram guardadas, sem ter que haver um contacto direto com a mesma.

Para o décimo primeiro requisito, de forma a garantir que site se encontra sempre disponível, mesmo em situações de ataque ao servidor onde se encontra, ou mesmo uma resposta mais rápida aos utilizadores, deve ser implementada uma *cache* no *back-end*.

Nesta etapa final do projeto, quase todos os requisitos identificados foram implementados, com exceção da responsividade, alguma informação existente no *website*, como as estatísticas dos alunos e os parágrafos alusivos às páginas onde se encontram e a *cache* que, apesar de não ter sido implementada no *back-end*, este está preparado para a sua implementação. Foi feita também uma alteração ao 3º requisito, o qual não irá ser implementado, pois torna o *website* pouco compatível com dispositivos móveis.

Persiste ainda um erro identificado na última fase de desenvolvimento, erro este que surgiu com a adição de novas funcionalidades, nomeadamente com a criação da aplicação *React.js*[REACT21] e com a adição dos botões para os detalhes dos perfis: o carrossel não retorna ao início quando é apresentada a totalidade dos elementos. Houve a necessidade de desativar o *infinite scroll* nos carrosséis com os perfis de alunos, docentes e empresas, pois com esta funcionalidade ativa não era possível ver as informações de alguns perfis. Para a resolução deste problema é necessário seguir uma das seguintes soluções:

- criar uma função que faça com que o carrossel retorne ao início quando é mostrado o último utilizador
- configurar as definições dos carrosséis com outras propriedades adequadas

Ainda não foi possível resolver este erro pois a utilização de diferentes propriedades para a configuração dos carrosséis pode ter algum impacto na aparência do website, pelo que este erro tem que ser corrigido paralelamente com o desenvolvimento da responsividade.

3. Viabilidade e Pertinência

Este projeto enquadra-se nos pressupostos de viabilidade e pertinência, pois a solução final deste *website*, desenvolvida a pedido da direção do curso de Engenharia Informática, continuará a representar o Departamento de Engenharia Informática e Sistemas de Informação, trazendo vantagens tanto para o departamento como para a Universidade Lusófona de Humanidades e Tecnologias. Não será um projeto que se esgote enquanto projeto académico, podendo ser continuado após a conclusão deste TFC, pois irá representar o departamento, tal como a versão atual do *website* representa, até ser suscetível a alterações.

À exceção dos custos de *hosting* do *website* e respectivas bases de dados, custos existentes desde o lançamento inicial e suportados pela faculdade, não irão haver custos extra dado que todas as ferramentas e serviços utilizados durante o desenvolvimento deste trabalho não têm quaisquer custos.

A solução desenvolvida proporcionará ao Departamento de Engenharia Informática e Sistemas de Informação um novo *website*, com uma interface e conteúdos novos e atualizados, bem como uma maior facilidade em manter esses conteúdos atualizados, de modo a proporcionar a utilizadores externos uma boa experiência com a plataforma, captando assim a sua atenção e interesse no departamento e respectivos cursos, ou até mesmo na Universidade Lusófona.

4. Solução Desenvolvida

Podemos dividir o desenvolvimento deste *website* em duas grandes componentes: o *front-end*, a interface gráfica do site e o *back-end*, a lógica por detrás de todo o site. No *front-end*, o grande foco será a utilização de *React.js*[REACT21], uma biblioteca leve de *JavaScript* que permite criar interfaces baseadas em componentes isolados e reutilizáveis, tornando o *website* eficiente e rápido. Também irão ser utilizadas outras bibliotecas de *JavaScript* para representação de dados e para a criação de carrosséis nas diversas do *website*, como o *Chart.js* [CHRTJS22] e o *Slick.js*[SLICK22], duas bibliotecas que permitem a criação de gráficos e a implementação dos carrosséis nas diversas páginas do *website*, respetivamente. No *back-end*, o grande foco será a utilização de *Spring MVC*[SPMVC21], uma *framework* de *Java* que simplifica a gestão de pedidos e respostas *HTTP* e estabelece a ligação à base de dados, neste caso *MySQL*. Irão também ser utilizadas a linguagem de *markup HTML5* na criação e modificação de elementos e a linguagem de estilos *CSS* para estilizar o *website*.

A biblioteca *React.js*[REACT21] é uma biblioteca de *JavaScript open-source*, desenvolvida pela *Meta* (anteriormente conhecida como *Facebook*) com o objetivo de criar interfaces eficientes para os utilizadores sem grandes complicações, através do conceito de componentes. Os componentes são elementos independentes e reutilizáveis, com um estado individual que pode ser atualizados individualmente quando os seus dados mudam. Para utilizar estes componentes e atualizá-los individualmente, o *React.js*[REACT21] usa um *DOM (Document Object Model)* virtual (*VDOM*), uma representação virtual da interface guardada em memória que é sincronizada com o *DOM* “real”.

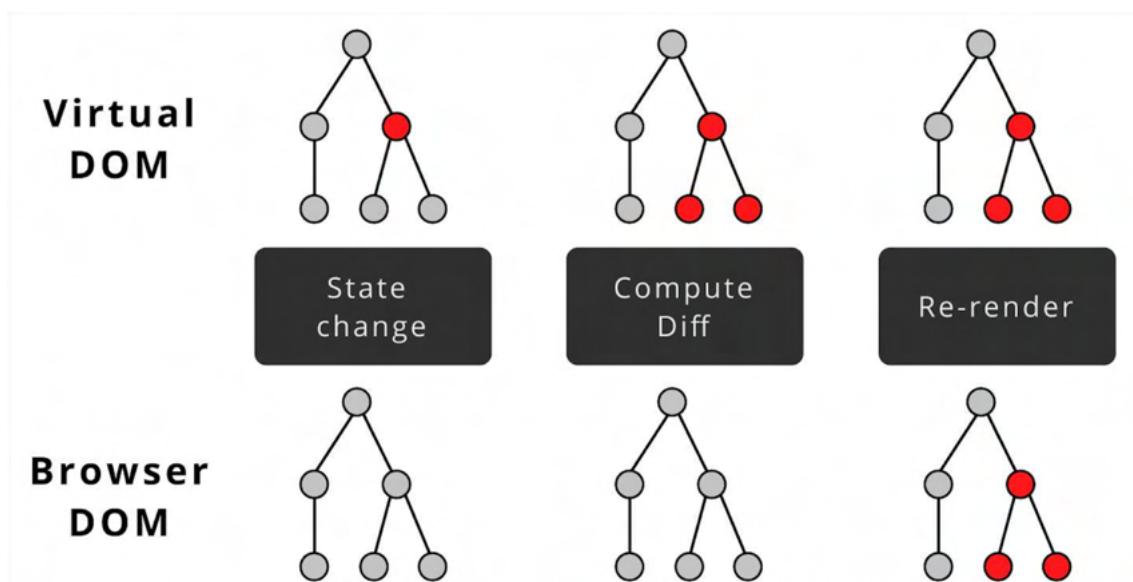


Fig. 2 - Representação do Funcionamento do DOM Virtual

O *React.js*[REACT21] permite ainda a utilização de uma extensão de sintaxe do *JavaScript*, denominada *JSX (JavaScript XML)*. O *JSX* permite-nos escrever *HTML* diretamente com o *JavaScript*, criando assim “elementos” *React.js*[REACT21], em vez de separar a linguagem de

markup e a lógica do *JavaScript* em ficheiros diferentes. Apesar de não ser necessário, a utilização desta sintaxe dá uma visão mais clara da interface quando trabalhada diretamente dentro do código de *JavaScript*. Permite também ao *React.js*[REACT21] mostrar mais mensagens de aviso e erros úteis (melhor *debugging*).

A framework *Spring MVC*[SPMVC21] funciona em torno de um elemento chamado *DispatcherServlet* (encaminhador de serviços), que encaminha pedidos *HTTP* para caminhos específicos, através de *handlers*. O *handler* pré-definido é baseado nas anotações `@Controller` e `@RequestMapping`, oferecendo uma ampla variedade de métodos de manipulação flexíveis.

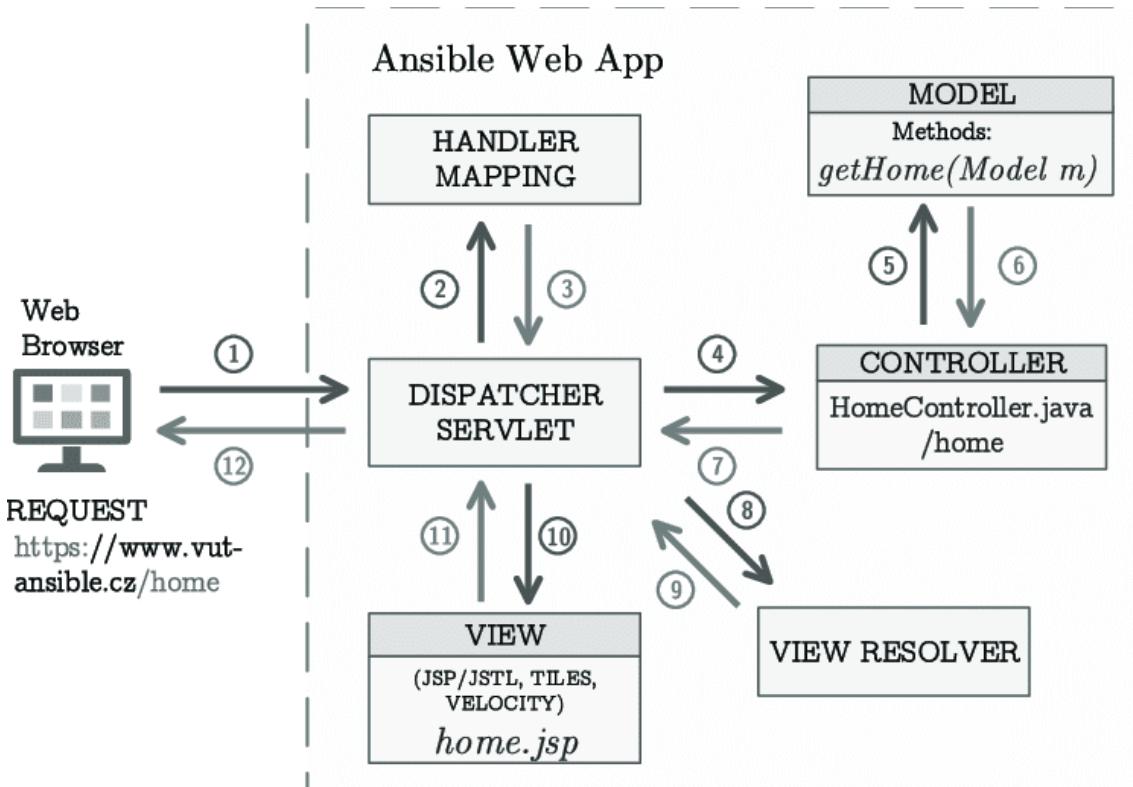


Fig. 3 - Funcionamento da Framework Spring MVC

No *Spring MVC*[SPMVC21], podemos utilizar qualquer objeto como um comando ou objeto de suporte de formulário, não sendo necessário implementar uma estrutura específica ou uma interface de classe base. A vinculação de dados do *Spring MVC*[SPMVC21] é altamente flexível: por exemplo, trata incompatibilidades de tipo como erros de validação, podendo ser avaliados pela aplicação e não como erros do sistema. Portanto, não é necessário duplicar as propriedades dos seus objetos de negócio como *strings* simples e não digitadas em seus objetos de formulário simplesmente para lidar com envios inválidos ou para converter as *strings* de maneira adequada. Em vez disso, geralmente é preferível vincular diretamente a objetos de negócios.

A resolução de visualização do *Spring MVC*[SPMVC21] é extremamente flexível. Um controlador é normalmente responsável por preparar um mapa de modelo com dados e selecionar um nome de visualização, o qual também pode gravar diretamente no fluxo de resposta e concluir o

pedido. A resolução do nome da visualização é altamente configurável através da extensão do arquivo ou da negociação do tipo de conteúdo do cabeçalho Aceitar, por meio de nomes de bean, um arquivo de propriedades ou até mesmo uma implementação de *ViewResolver* personalizada. O modelo é uma interface de mapa, que permite a abstração completa da tecnologia de visualização. É possível integrar diretamente com tecnologias de renderização baseadas em templates, como *JSP*, *Velocity* e *Freemarker*, ou gerar diretamente *XML*, *JSON*, *Atom* e muitos outros tipos de conteúdo. O modelo *Map* é simplesmente transformado num formato apropriado, como atributos de solicitação *JSP*, um modelo de template *Velocity*.

Abaixo estão representados os diagramas de arquitetura da aplicação, que explica como é feita a interação entre a aplicação, a base de dados e a CDN utilizada, e o diagrama de ER, que explica como são consituidas as entidades presentes neste website.

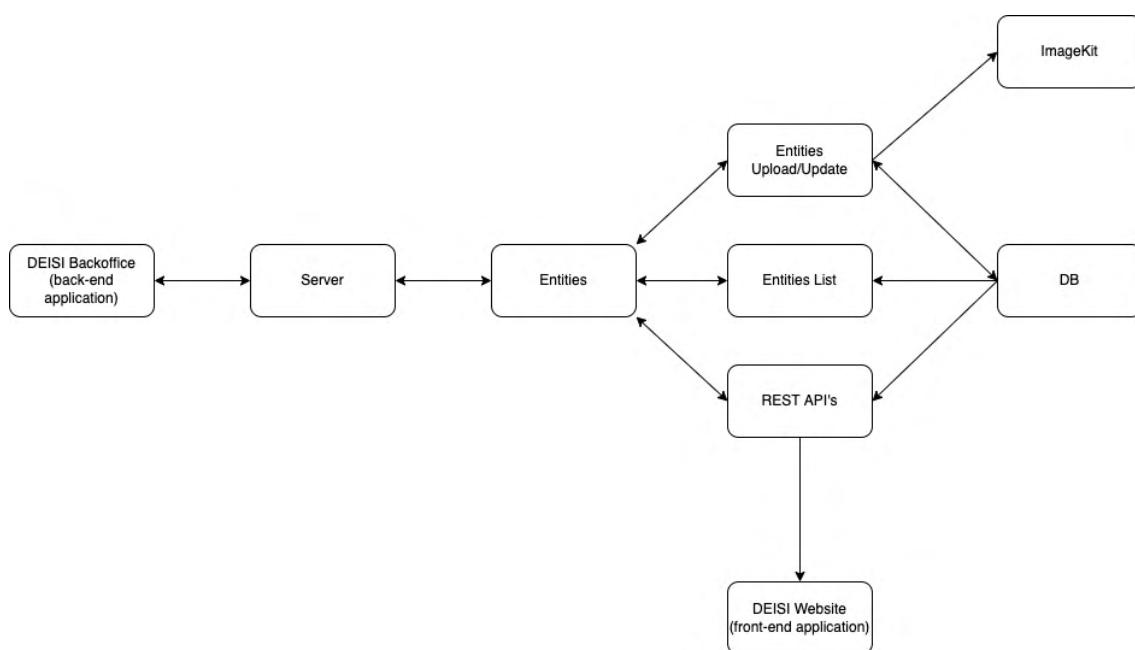


Fig. 4 – Arquitetura Da Aplicação

Company		Course		Page		Student		Teacher	
PK	<u>id int NOT NULL</u>	PK	<u>id int NOT NULL</u>	PK	<u>id int NOT NULL</u>	PK	<u>id int NOT NULL</u>	PK	<u>id int NOT NULL</u>
	name string		title char		pageName char		name char		name char
	imgSrc string		degree char		title char		gradYear int		age int
	description string		url char		stat char		imgSrc char		imgSrc char
			courseDescripiton char(1000)		statText char		description char(300)		desription char(300)
					pageText char(510)				

Fig. 5 – Diagrama de ER das Entidades do Website

4.1. Disciplinas e Áreas Científicas

As disciplinas através das quais adquiri conhecimentos que irei aplicar no desenvolvimento deste trabalho são:

- Bases de Dados - conhecimentos relacionados com bases de dados, desde a familiarização com a linguagem *SQL* à construção de bases de dados e à criação de *queries*.
- Programação *Web* - conhecimentos relacionados com o desenvolvimento *front-end* e *back-end*.
- Engenharia Requisitos e Testes - conhecimentos relacionados com o levantamento de requisitos, planeamento e gestão do projeto.
- Interação Humano-Máquina - conhecimentos relacionados com *design* da interface do utilizador e questões de acessibilidade.
- Algoritmia e Estrutura Dados - conhecimentos relacionados com a manipulação e análise dos dados.
- Linguagens de Programação II - conhecimentos relacionados com programação orientada a objetos.

4.2. Protótipo Inicial

Na 2^a fase de desenvolvimento deste TFC e intercalado com o levantamento de requisitos feito inicialmente, junto de alguns professores do DEISI, foi desenvolvido um *mockup* e, de seguida, um protótipo inicial, o qual procurou implementar a maioria dos requisitos identificados no terceiro capítulo, “Levantamento de Requisitos”.

O mockup desenvolvido foi utilizado como base para a *landing page* e para fazer um pequeno estudo de uma possível *landing page* para dispositivos móveis, na qual apenas foi estudado o método de navegação:



Fig. 6 – Landing Page Mockup



Fig. 7 – Landing Page Mockup Dispositivos Móveis

Com base no *mockup* anteriormente desenvolvido e após o levantamento de alguns requisitos, foi desenvolvido o protótipo inicial:

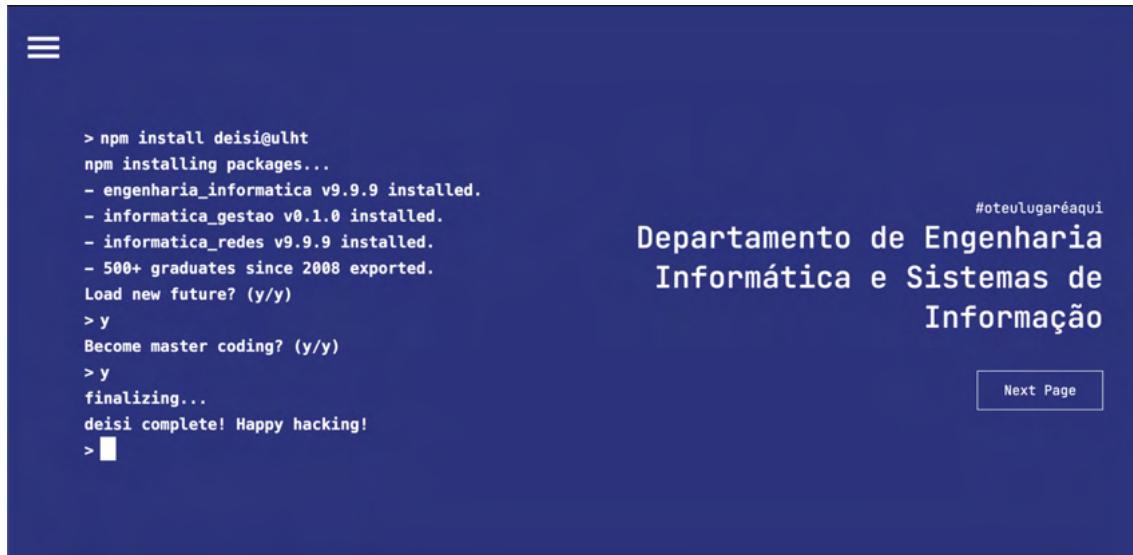


Fig. 8 – Landing Page Protótipo Inicial

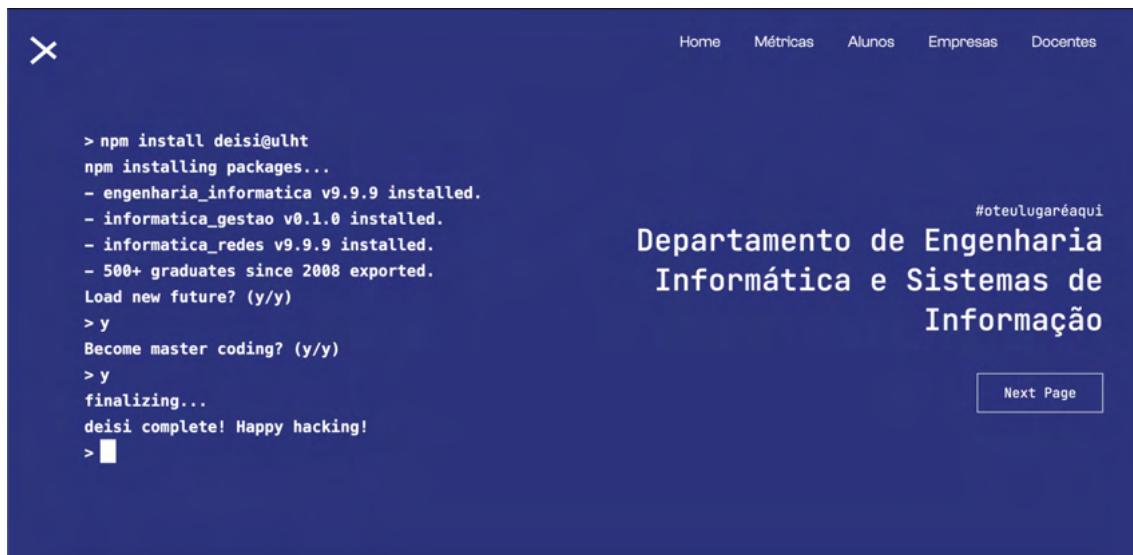


Fig. 9 – Landing Page Protótipo Inicial Com Menu Visível

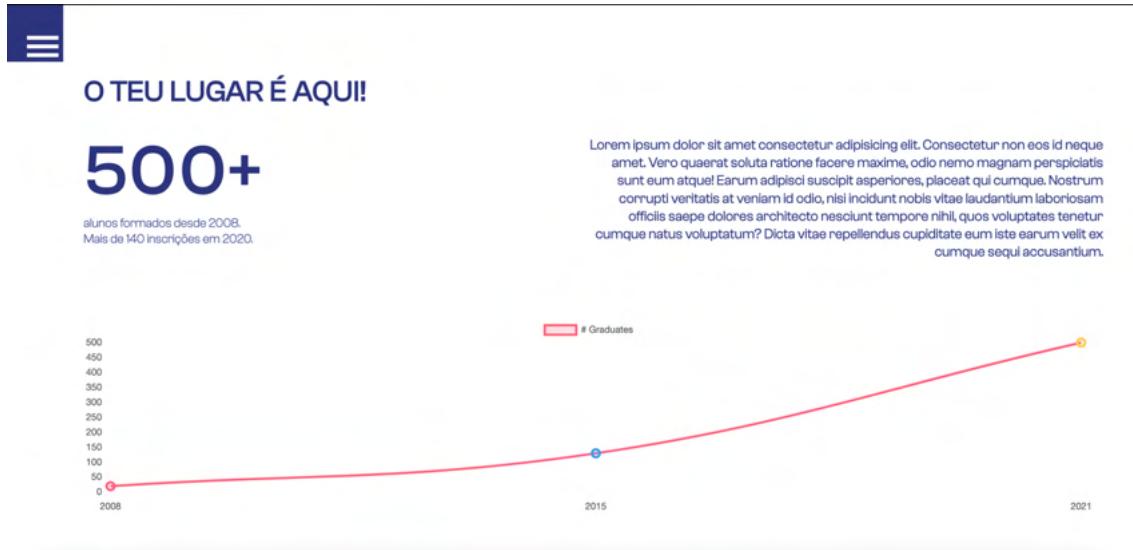


Fig. 10 – Página Métricas Protótipo Inicial

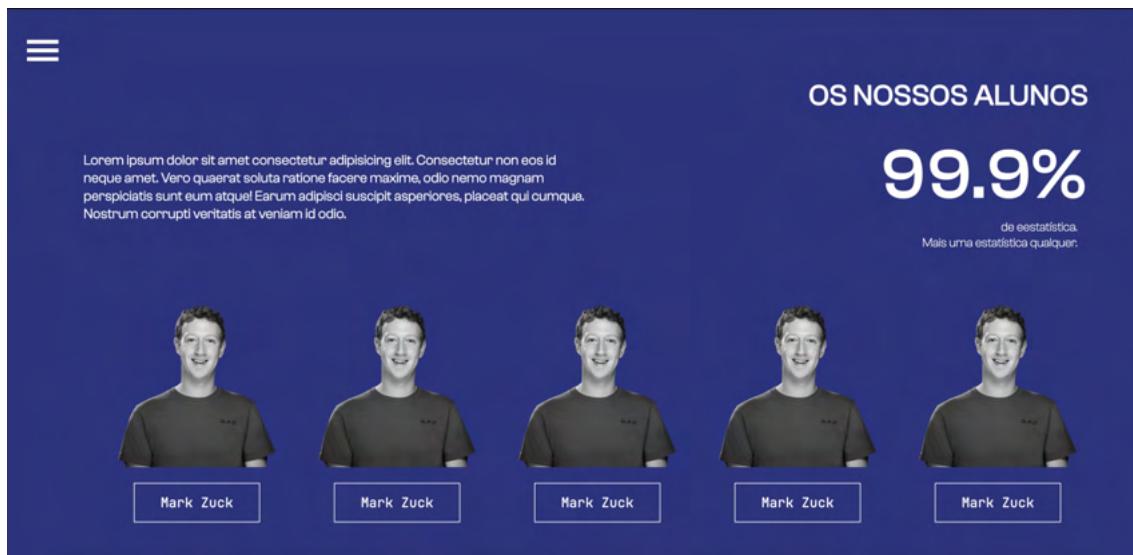


Fig. 11 – Página Estudantes Protótipo Inicial



Fig. 12 – Página Empresas Protótipo Inicial

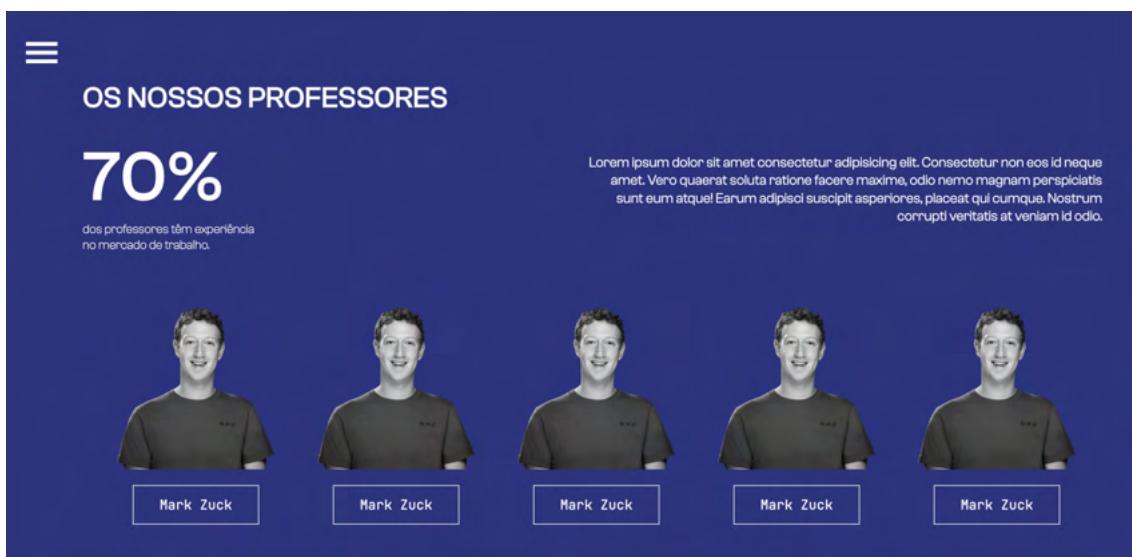


Fig. 13 – Página Docentes Protótipo Inicial

Na etapa em que este protótipo foi desenvolvido, não foi possível implementar algumas das funcionalidades, como a responsividade do *website* ou a visualização de dados provenientes do *back-end* via *web-services*, no entanto, foi implementada a maioria dos requisitos identificados, com exceção das estatísticas na páginas dos alunos e a possibilidade de visualizar e ocultar os detalhes dos perfis de alunos e docentes, quando os respectivos botões são clicados. Também é de notar que não é possível, através das imagens acima, visualizar o funcionamento de certos elementos e funcionalidades, como os diversos carrosséis, a animação da *landing page*, o efeito “*full scroll*”, entre outros.

4.3. Front-End

Como foi referido no início deste capítulo, para o desenvolvimento *front-end* foi utilizada a biblioteca *React.js*[REACT21]. Neste sentido, foi criada uma aplicação *React.js*[REACT21] manualmente, permitindo assim uma maior flexibilidade na modificação de configurações e reduzindo as dependências apenas às necessárias, tornando a aplicação mais leve.

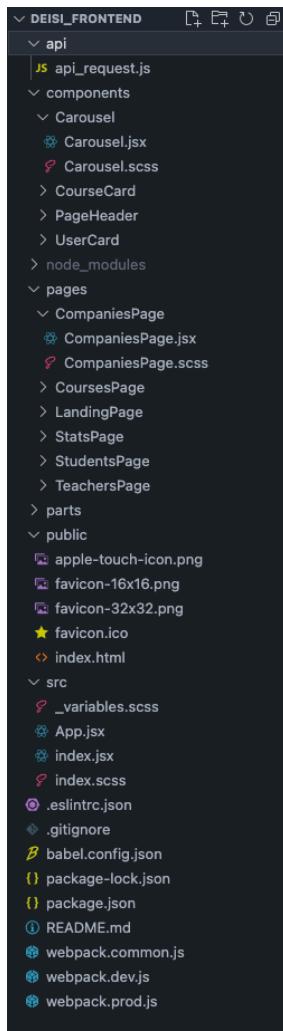
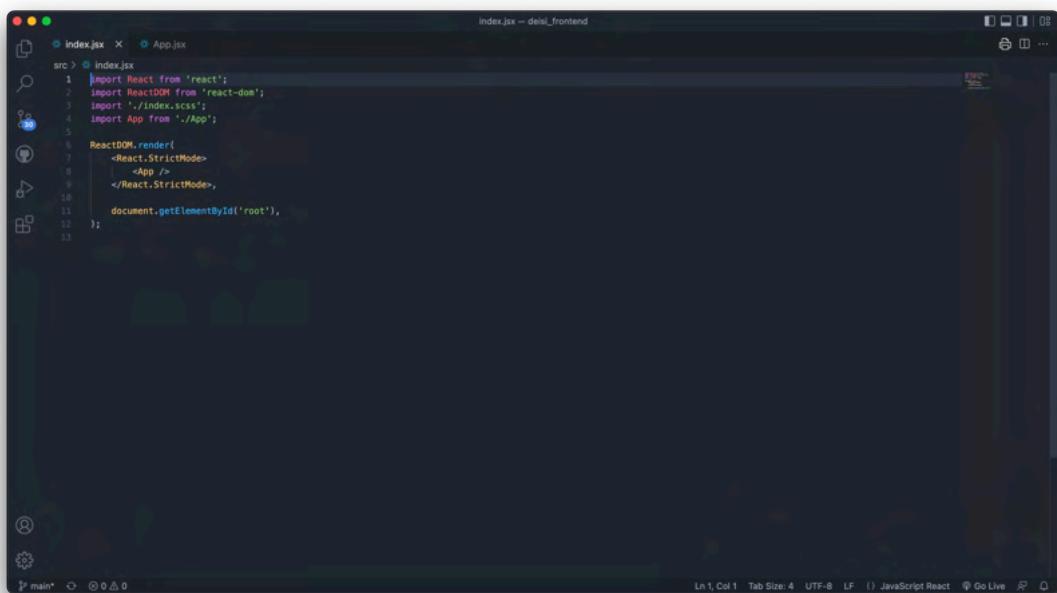


Fig. 14 – Estrutura da Aplicação Front-End

A aplicação *React.js*[REACT21] é criada sobre *Webpack*[WBPCK22], um *bundler* de *JavaScript*, isto é, uma ferramenta responsável por empacotar todo o código num único ficheiro *JavaScript*, diminuindo assim o tamanho do projeto, o que trás algumas vantagens como o carregamento mais rápido do projeto e uma melhoria no tempo de resposta aos utilizadores. O *Webpack*[WBPCK22] também pode empacotar outro tipo de *assets* de uma aplicação *front-end*, desde que tenha os *loaders* necessários para o fazer. É também através do *Webpack*[WBPCK22] que são definidos os ambientes de desenvolvimento e de produção. Uma das vantagens da utilização de *React.js*[REACT21] referida anteriormente, no início deste capítulo, é a utilização de uma extensão de sintaxe do *JavaScript*, denominada *JSX*. No entanto, o *browser* não consegue interpretar diretamente a sintaxe utilizada e por isso necessita de um transpilador. Desta forma, é utilizado o transpilador *Babel*[BABEL22] para realizar essa tarefa. É também ainda

utilizado o *linter ESLint*[ESLINT22], isto é, uma ferramenta de análise de código que identifica bugs, erros de programação ou erros de estilo, através do qual é seguido o guia de estilos *Airbnb*, de forma a que o código seja executado sem erros e que seja mais legível para futuros desenvolvimentos.

O ponto de entrada desta aplicação é o ficheiro “*index.jsx*” na pasta “*/src*”, que por sua vez chama o ficheiro “*App.js*”, ficheiro este onde é feito o layout do *website* e são chamadas todas as páginas que o constituem assim como as partes que o constituem, como a barra de navegação. É também através do ficheiro “*index.jsx*” que é chamado o único ficheiro *HTML* utilizado em toda a aplicação, onde irão ser renderizadas todas as páginas e elementos criados. Todos os elementos desta aplicação têm dois ficheiros: um ficheiro com extensão *.jsx*, onde é definida a sua estrutura bem como as suas funcionalidades e um ficheiro *.scss*, que define o seu estilo, ou parte dele. A aplicação, na mesma diretoria que o ficheiro “*index.jsx*” também tem um ficheiro chamado “*index.scss*”, que define os estilos utilizados por toda a aplicação.



The screenshot shows a code editor window with a dark theme. The title bar says "index.jsx - deis1_frontend". The left sidebar shows a file tree with "index.jsx" and "App.jsx" under a "src" folder. The main editor area contains the following code:

```
index.jsx
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.scss';
4 import App from './App';
5
6 ReactDOM.render(
7   <React.StrictMode>
8     <App />
9   </React.StrictMode>
10
11   document.getElementById('root'),
12 );
```

The status bar at the bottom shows "Ln 1, Col 1 Tab Size: 4 UTF-8 LF () JavaScript React Go Live".

Fig. 15 - Ficheiro index.jsx

```

import React from 'react';
import Navbar from './parts/Navbar/Navbar';
import LandingPage from './pages/LandingPage/LandingPage';
import StatsPage from './pages/StatsPage/StatsPage';
import StudentsPage from './pages/StudentsPage/StudentsPage';
import CompaniesPage from './pages/CompaniesPage/CompaniesPage';
import TeachersPage from './pages/TeachersPage/TeachersPage';
import CoursesPage from './pages/CoursesPage/CoursesPage';

function App() {
  return (
    <div>
      <header>
        <Navbar />
      </header>
      <main>
        <LandingPage />
        <StatsPage />
        <StudentsPage />
        <CompaniesPage />
        <TeachersPage />
        <CoursesPage />
      </main>
    </div>
  );
}

export default App;

```

Fig. 16 – Ficheiro App.jsx

Foram definidos alguns componentes, isto é, elementos independentes e reutilizáveis, pois existem diversos elementos semelhantes neste *website*, simplificando assim o código do mesmo. Encontram-se na pasta “/components” e estes são todos os cabeçalhos das páginas, os cartões usados em todos os carrosséis do *website* e o carrossel utilizado para demonstrar os perfis de alunos, docentes e empresas. Para realizar os pedidos à API é utilizado o ficheiro “api_request.js”.

```

import { useState, useEffect } from 'react';

export default function APIRequest(urlAPI) {
  const [isLoaded, setIsLoaded] = useState(false);
  const [isError, setIsError] = useState(undefined);
  const [apiData, setAPIData] = useState(undefined);

  useEffect(() => {
    fetch(urlAPI)
      .then(res => res.json())
      .then(data => {
        setIsLoaded(true);
        setAPIData(data);
        setIsError(false);
      })
      .catch(err => {
        setIsError(true);
        setIsError(err.status);
      });
  }, []);

  return [isLoaded, isError, apiData];
}

```

Fig. 17 – Ficheiro api_request.js

Tendo como exemplo do funcionamento do *website* e respetivas páginas, a página de alunos, é possível verificar que, em primeiro lugar, é feito um pedido à API, ao *endpoint* correspondente. De seguida, são definidas os parâmetros que irão ser passados ao cabeçalho desta página, em formato *JSON* (o componente está preparado já para receber os dados via API, cumprindo o primeiro ponto do 9º requisito definido no capítulo anterior). De seguida, é definido o *layout* da página, onde é criado um cabeçalho através do componente e dos parâmetros previamente definidos, bem como um carrossel através do componente também previamente criado, ao qual são passados os dados resultantes do pedido à API.

```
StudentPage.js 1 x
DEBIL_PROJECT ... StudentPage.js 1 x StudentPage.js > StudentsPage
pages > StudentPage.js > StudentPage.js > StudentsPage
  import React from 'react';
  import apiRequest from '../../../../../api/api_request';
  import PageHeader from '../../../../../components/PageHeader/PageHeader';
  import Carousel from '../../../../../components/Carousel/Carousel';
  import './StudentsPage.css';

  export default function StudentsPage() {
    const [isLoaded, isError, apidata] = apiRequest('/students/api/list');

    const pageSettings = {
      title: '01 NOSSOS ALUNOS',
      stat: '99.9%',
      statText: 'de estatística. Melhor uma estatística realista!',
      paragraph: [
        'Amet, vero quem dolor sit amet consectetur adipisci elit. Consectetur non ea id neque',
        'amet. Vera quem soluta facere maxime, odio nemo magnam perspicillatis sunt eum',
        'atque earum adipisci suscipit asperiores, placeat qui cumque. Nestrum corrupti',
        'veritatem at veniam id etia, nisi incidenti quis vita laudemus officilia',
        'cumque, quod est. Quod enim tempore, nihil quisque, qui est, qui est, qui est, qui est, qui est,',
        'veluptatum? Dicta vitam repellentes copiditate sum iste earum velit in sumo sequi',
      ],
      textColor: '#ffffff',
      orientation: 'row-reverse',
      statAlign: 'right',
      paragraphAlign: 'left',
    };

    return (
      <div id="students-page" className="students-page section page">
        <PageHeader {...pageSettings} />
        {isLoaded === false || isError === undefined || apidata === null ? (
          <div className="students-carousel">
            <Carousel carrousel={apidata} killores={5} buttonStyle="regular" textColor="#ffffff" />
          </div>
        ) : (
          <div className="error-carousel">
            <p>Erro! <code>api_request</code></p>
          </div>
        )}
      </div>
    );
  }
}

L9, Col 70 Tab Size 4 UTF-8 LF () JavaScript React ⌂ Go Live ⌂
```

Fig. 18 – Ficheiro StudentsPage.jsx

A criação do carrossel e respectivos cartões funciona de maneira semelhante à criação dos cabeçalhos. Ao carrossel, são passados os dados da API, aos quais irá ser feito um *map* e, para cada elemento, irá ser criado um cartão através do componente também previamente criado. É esse elemento que depois irá mostrar os dados relativos a cada perfil que lhe foram passados.

```

Carousel.jsx - deisifrontend
components > Carousel > Carousel.jsx > Carousel > constructor > cardData.propTypes callback
  11  export default function Carousel(props) {
  12    const {
  13      cardData, nrSlides, buttonStyle, textColor,
  14    } = props;
  15
  16    const sliderSettings = {
  17      arrows: false,
  18      dots: true,
  19      slidesToShow: nrSlides,
  20      slidesToScroll: 1,
  21      infinite: true,
  22      autoplay: true,
  23      autoplaySpeed: 2000,
  24      pauseOnHover: true,
  25      responsive: [
  26        {
  27          breakpoint: 1824,
  28          settings: {
  29            slidesToShow: 3,
  30          },
  31        },
  32        {
  33          breakpoint: 768,
  34          settings: {
  35            slidesToShow: 2,
  36          },
  37        },
  38        {
  39          breakpoint: 480,
  40          settings: {
  41            slidesToShow: 1,
  42          },
  43        },
  44      ],
  45    };
  46
  47    return (
  48      <div className="content">
  49        <Slider {...sliderSettings}>
  50          {cardData.map((card, index) => (
  51            <UserCard index={index} card={card} buttonStyle={buttonStyle} textColor={textColor} />
  52          ))
  53        </Slider>
  54      </div>
  55    );
  56  }
  57
  58  Carousel.propTypes = {
  59    cardData: PropTypes.array.isRequired,
  60    buttonStyle: PropTypes.object.isRequired,
  61    textColor: PropTypes.array.isRequired,
  62  };
  63
  64  UserCard.propTypes = {
  65    card: PropTypes.object.isRequired,
  66    buttonStyle: PropTypes.array.isRequired,
  67    textColor: PropTypes.array.isRequired,
  68  };
  69}
  
```

Fig. 19 – Ficheiro Carousel.jsx

```

UserCard.jsx - deisifrontend
components > UserCard > UserCard.jsx > UserCard > constructor > card.propTypes
  11  import React from 'react';
  12  import PropTypes from 'prop-types';
  13
  14  import './UserCard.css';
  15
  16  export default function UserCard(props) {
  17    const { card, buttonStyle, textColor } = props;
  18
  19    function toggleDescription() {
  20      const cardDescription = document.getElementById(card.name);
  21      const userDescription = document.getElementById(card.description);
  22
  23      if (imageStyle.display === 'flex') {
  24        imageStyle.display = 'none';
  25        userDescription.style.display = 'inline';
  26      } else {
  27        imageStyle.display = 'flex';
  28        userDescription.style.display = 'none';
  29      }
  30
  31      return (
  32        <div key={card.name} className="card">
  33          <div id={card.name} className="info no-display" style={{ color: textColor }}>{card.description}</div>
  34          <div id={card.name} className="image-container">
  35            <img alt={card.imageAlt || card.name} />
  36            <button className="user-button terminal-font button-{buttonStyle}" onClick={() => toggleDescription()}>{card.name}</button>
  37          </div>
  38        </div>
  39      );
  40    }
  41
  42    UserCard.propTypes = {
  43      card: PropTypes.object.isRequired,
  44      buttonStyle: PropTypes.array.isRequired,
  45      textColor: PropTypes.array.isRequired,
  46    };
  47  }
  48
  49  UserCard.defaultProps = {
  50    card: {},
  51    buttonStyle: [],
  52    textColor: []
  53  };
  54}
  
```

Fig. 20 – Ficheiro UserCard.jsx

Os problemas que mais dificultaram o desenvolvimento desta parte do trabalho foi um erro de CORS no ambiente de desenvolvimento, que impediam de fazer o pedido à API e visualizar os dados. Para resolver este problema, foi necessário utilizar uma proxy no ambiente de desenvolvimento, de modo a poder visualizar os dados provenientes do back-end.

4.4. Back-End

Como foi referido no início deste capítulo, para o desenvolvimento *back-end* foi utilizada a linguagem de programação *Kotlin*, através da qual foi implementada a *framework Spring MVC*[SPMVC21].

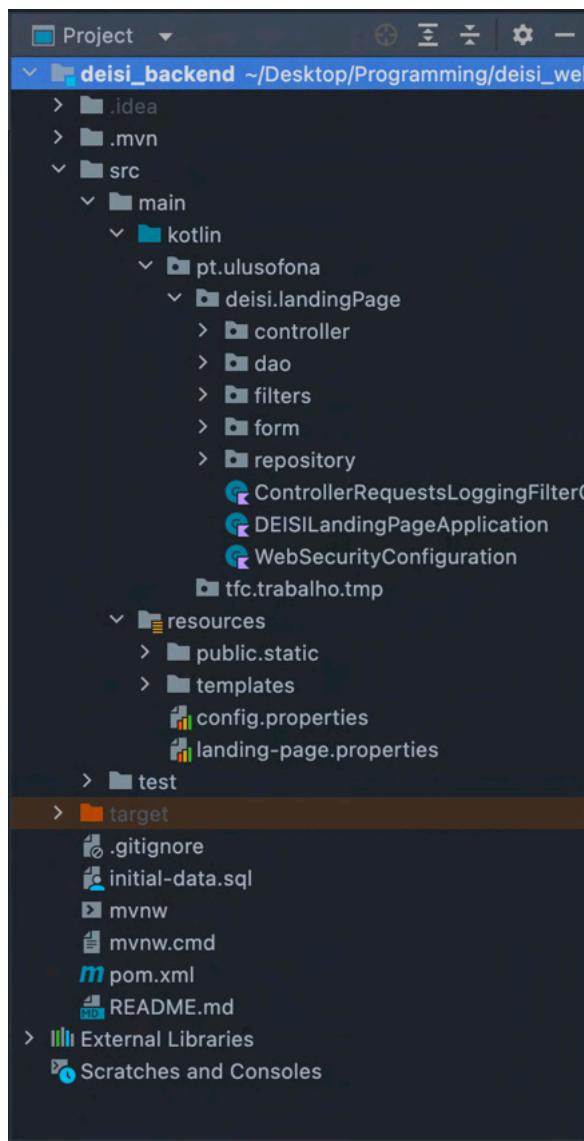


Fig. 21 – Estrutura da Aplicação Back-End

Esta aplicação é constituída por 7 pastas principais: *controllers*, *dao*, *filters*, *form*, *repository*, *templates* e *tmp*:

- a pasta de *controllers* contém os controladores de todas as entidades presentes, bem controladores dos respetivos *endpoints*, isto é, os controladores responsáveis por traduzir a informação adicionada através dos formulários para formato *JSON*. Os controladores são as classes onde se encontram todas as funções responsáveis pela leitura, criação, modificação ou eliminação das entidades a que

correspondem, fazendo a ponte entre a base de dados e a interface desta aplicação. Desta forma, são estas classes as responsáveis pelas respostas aos pedidos efetuados pelo utilizador. Foram criadas, para cada entidade, duas classes de controladores, uma responsável por todas as operações, outra responsável pelo *endpoint*

- na pasta *dao* encontram-se todas as entidades criadas na base de dados. São estas classes que definem os parâmetros de cada entidade, os quais irão ser utilizados nos formulários bem como na base de dados. Existem 4 classes *dao*, uma para cada entidade existente no projeto
- na pasta *filters* encontram-se os filtros para os ficheiros que a aplicação irá ou não servir nos pedidos dos utilizadores
- na pasta *form* encontram-se as classes responsáveis pelos formulários existentes nesta aplicação, isto é, é nas classes dos formulários que são definidas as características de cada campo a preencher, respeitando as entidades criadas na pasta *dao*. Existem 4 classes *form*, uma para cada formulário existente no projeto
- na pasta *repository* encontram-se as estruturas de dados onde são guardadas as entidades existentes na base de dados. É através destas estruturas que se torna possível visualizar todas as entidades existentes na base de dados, bem como é possível adicionar novas entidades, as quais irão ser guardadas na mesma. Existem 4 classes *repository* neste projeto, uma para cada entidade
- na pasta *templates* encontram-se todas as páginas HTML necessárias para o funcionamento deste projeto. Existem 3 páginas HTML para cada entidade, uma para a criação de novas entidades, uma para a eliminação de entidades e uma para a lista de entidades existentes. Na próxima etapa de desenvolvimento o número será reduzido a 2, pois a existência de uma página HTML apenas para a eliminação de entidades não traz quaisquer vantagens para o projeto
- à pasta *tmp* são adicionadas temporariamente as imagens submetidas através dos formulários na criação das entidades, para a submissão da CDN. Após a submissão, as imagens são eliminadas

Como foi referido no parágrafo anterior, toda a informação é adicionada à base de dados através de formulários, a qual é possível consultar através de uma lista de entidades existentes, correspondente à entidade que pretendemos consultar. A lista de dados é convertida, através dos *API controllers*, para formato *JSON*, informação esta que depois irá ser usada em diversos componentes na aplicação *front-end* através de pedidos feitos a estas API's através da função demonstrada na Fig. 15 do ponto anterior.

The screenshot shows a web application interface for managing student data. At the top, there's a navigation bar with links for 'Trabalho', 'Páginas', 'Alunos', 'Professores', 'Cursos', and 'Empresas'. Below the navigation is a search bar labeled 'Apenas alunos com idade:' followed by a text input field and a 'filtrar' button. The main content area is titled 'Listagem de alunos' and displays a table of student records. The columns in the table are 'Nome', 'Idade', 'Imagen' (with a thumbnail preview), 'Descrição' (containing a short testimonial), and 'Operações' (with edit and delete icons). The data in the table is as follows:

Nome	Idade	Imagen	Descrição	Operações
Bruno Bacelar	22	https://lk.imagekit.io/smm2b6sqe/students_photos/bruno-bacelar_F7tDk81KP.jpeg?ik-sdk-version=javascript-1.4.3&updatedAt=1650798806492	A Lusófona tem um ensino muito bom e professores que se preocupam que os alunos aprendam e que tenham uma boa carreira. Uma experiência que vou levar comigo para sempre no meu percurso profissional.	
Hugo Serras	21	https://lk.imagekit.io/smm2b6sqe/students_photos/hugo-serras_FAORaqff.jpg?ik-sdk-version=javascript-1.4.3&updatedAt=1650798806875	A minha experiência na Lusófona foi algo soberbo, em todos os sentidos. É o que se costuma dizer de "life changing". Não é fácil a adaptação de uma pessoa a ingressar numa licenciatura aos 35 anos, mas foi o melhor passo que poderia ter dado!	
Caldina Pires	24	https://lk.imagekit.io/smm2b6sqe/students_photos/caldina-pires_ux9zGpxt.jpg?ik-sdk-version=javascript-1.4.3&updatedAt=1650798806599	Não subestimo os números num certificado. Mas, mais importante que a "beleza" dos números é a história por detrás deles. Isso sim determinará o tipo de profissional que nascerá de ti. Eu escrevi as minhas melhores histórias na Lusófona.	
Tiago Pinto	25	https://lk.imagekit.io/smm2b6sqe/students_photos/tiago_pinto_DN0JHLia.jpg?ik-sdk-version=javascript-1.4.3&updatedAt=1650798806626	Entrei sem saber uma única linha de código e saí com a sensação que podia fazer a diferença no mundo empresarial.	
Cld Robalo	20	https://lk.imagekit.io/smm2b6sqe/students_photos/cld-robalo_Fv5AULtbv.jpeg?ik-sdk-version=javascript-1.4.3&updatedAt=1650798807004	A ULHT me permitiu dar os meus primeiros passos para realizar os meus sonhos académicos e profissionais.	

Fig. 22 – Lista de Alunos Existentes na Base de Dados

The screenshot shows a web application interface for creating a new student record. At the top, there's a navigation bar with links for 'Backoffice Deisi Website', 'Páginas', 'Alunos', 'Professores', 'Cursos', and 'Empresas'. A user profile icon 'user1' is also visible. Below the navigation is a form titled 'Criar aluno'. The form fields are: 'Nome' (Name), 'Ano Graduação' (Year of graduation), 'Link Imagem' (Image link) with a 'Choose file' button, and 'Descrição' (Description). There's also a placeholder 'Descrição do aluno' (Student description). At the bottom of the form is a blue 'Gravar' (Save) button.

Fig. 23 – Formulário de Submissão de Novos Alunos

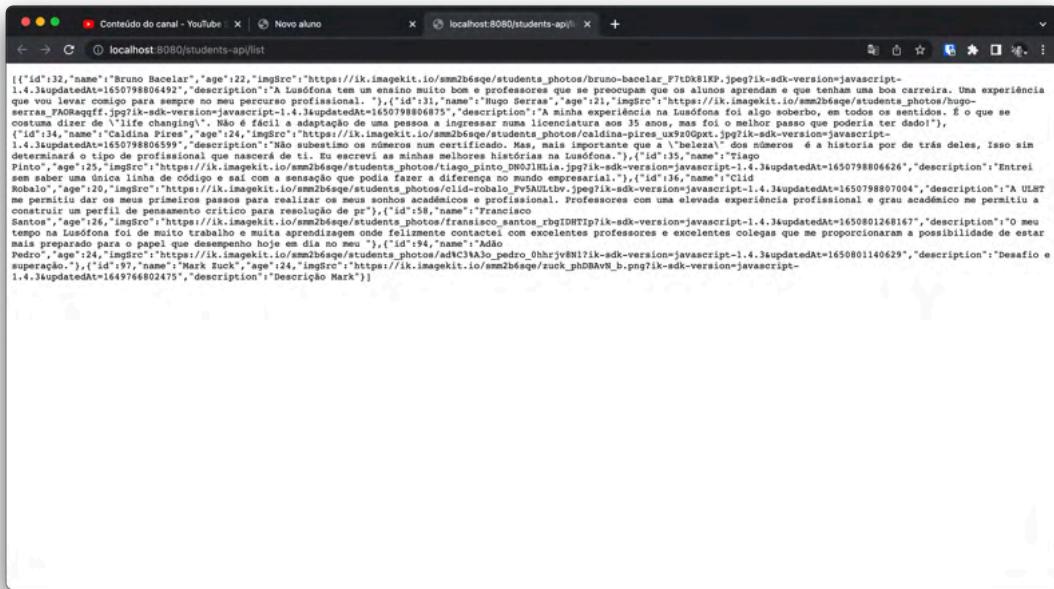


Fig. 24 – Endpoint da Lista de Alunos

The screenshot shows the IntelliJ IDEA interface with the code editor open to the `StudentController.kt` file. The code defines a controller for managing students. It includes methods for showing a new student form, editing an existing student, and deleting a student. Each method uses `@GetMapping` annotations and returns a string representing a view name. The code also uses `ModelMap` to pass data to the view, including the student's ID, name, age, image source, and description. The `studentRepository` is used to find students by their ID.

```
deisi_backend - StudentController.kt

delsi_backend > src > main > kotlin > pt > ulusofona > tfo > trabalho > controller > StudentController.kt
Project > StudentAPIController.kt > StudentController.kt

30
31
32
33     @GetMapping(value = ["/new"])
34     fun showStudentForm(model: ModelMap): String {
35         model["studentForm"] = StudentForm()
36         return "new-student-form"
37     }
38
39     @GetMapping(value = ["/edit/{id}"])
40     fun showStudentForm(@PathVariable("id") id: Long, model: ModelMap): String {
41         val studentOptional = studentRepository.findById(id)
42         if (studentOptional.isPresent) {
43             val student = studentOptional.get()
44             model["studentForm"] = StudentForm(studentId = student.id.toString(), name = student.name, age = student.age, imgSrc = student.imgSrc, description = student.description)
45         }
46
47         return "new-student-form"
48     }
49
50     @GetMapping(value = ["/delete/{id}"])
51     fun showStudentDelete(@PathVariable("id") id: Long, model: ModelMap): String {
52         val studentOptional = studentRepository.findById(id)
53         if (studentOptional.isPresent) {
54             val student = studentOptional.get()
55             model["studentForm"] = StudentForm(studentId = student.id.toString(), name = student.name, age = student.age, imgSrc = student.imgSrc, description = student.description)
56         }
57
58         return "delete-student-form"
59     }
60 }

All files are up-to-date (7 minutes ago) 15:12 LF UTT-B 4 spaces Event Log
```

Fig. 25 – Algumas Funções da Classe StudentsController.kt

```

@RestController
@RequestMapping("/students-api")
class StudentAPIController(val studentRepository: StudentRepository) {

    @GetMapping(value = ["/list"])
    fun listStudents(@RequestParam("age") age: Int?): List<Student> {
        val students = if (age == null) {
            studentRepository.findAll()
        } else {
            studentRepository.findByAge(age)
        }
        return students
    }
}

```

Fig. 26 – Função de Listagem de Alunos da Classe StudentsAPIController.kt

```

borisdvpr
fun uploadImageCDN(studentForm: StudentForm, file: MultipartFile) {
    val fileName: String = file.originalFilename ?: "temp"
    val tempDir = Files.createTempDirectory( prefix: "landing-page-uploads").toFile()

    val newImage = File(tempDir, fileName)
    file.transferTo(newImage)

    val bytes = Files.readAllBytes(Paths.get(newImage.path))
    val fileCreateRequest = FileCreateRequest(bytes, fileName)
    fileCreateRequest.isUseUniqueFileName = false
    val result = ImageKit.getInstance().upload(fileCreateRequest)

    studentForm.imgSrc = result.url
    newImage.delete()
}

```

Fig. 27 – Função de Upload das Imagens na CDN da Classe StudentsController.kt

4.5. Solução Final

Com base nos trabalhos desenvolvidos no protótipo inicial, no *back-end* e no *front-end*, foi possível chegar à solução final abaixo demonstrada, visualmente semelhante, apenas com pequenas diferenças em algumas funcionalidades a implementar na última fase de desenvolvimento:

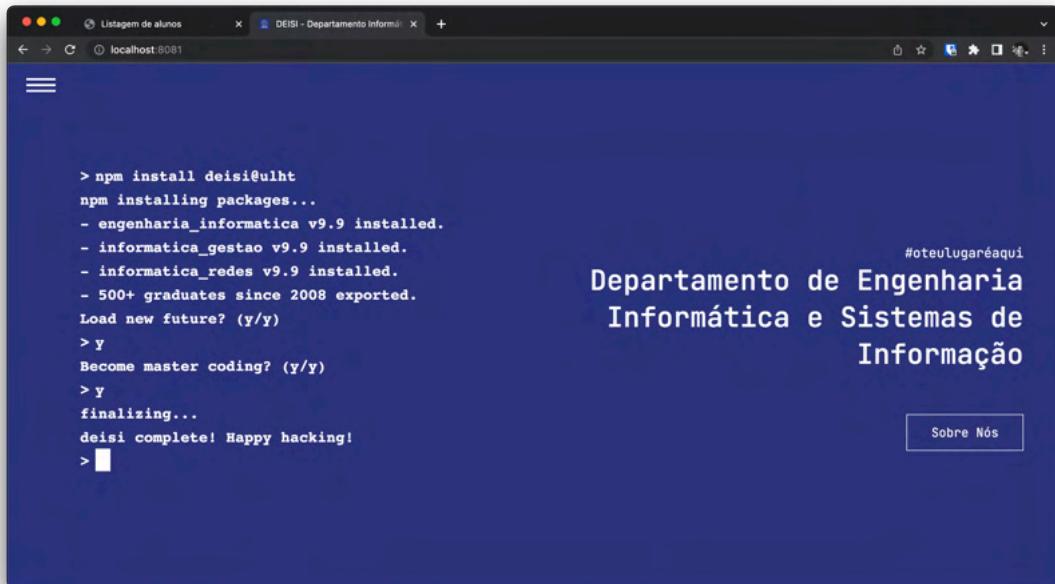


Fig. 28 – Landing Page da Solução Final

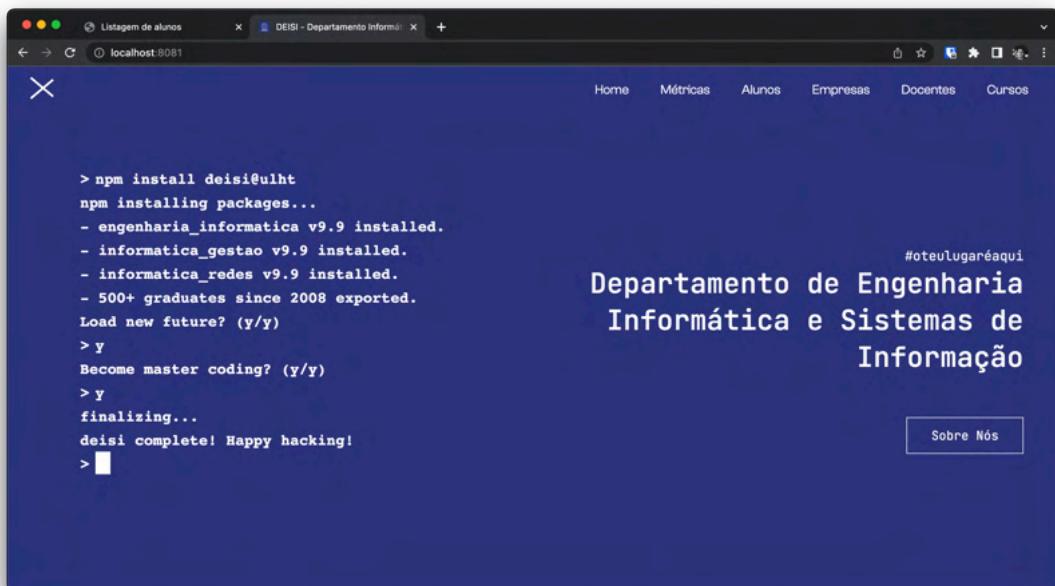


Fig. 29 – Landing Page da Solução Final c/ Menu

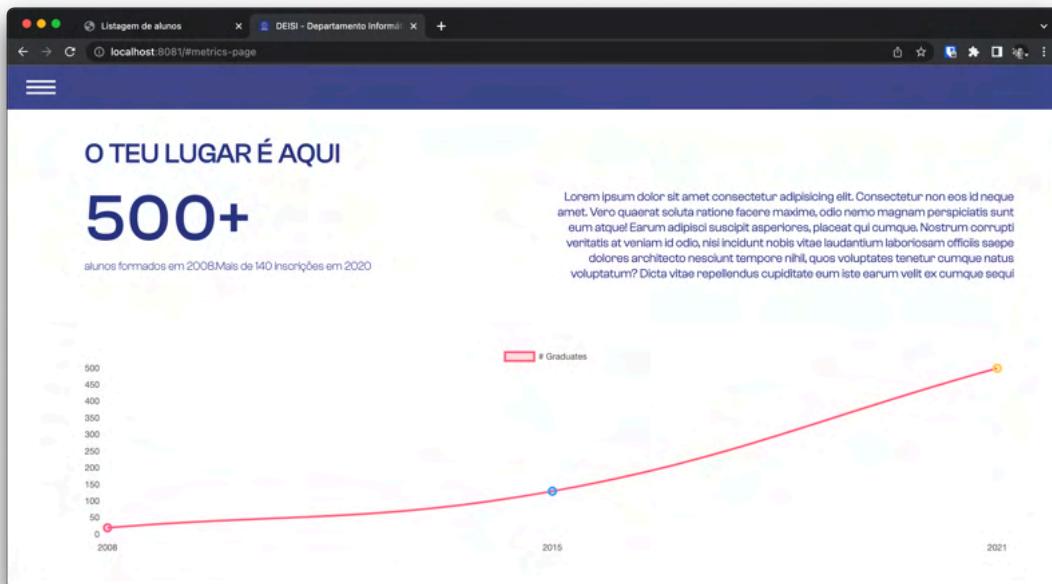


Fig. 30 – Página Métricas da Solução Final

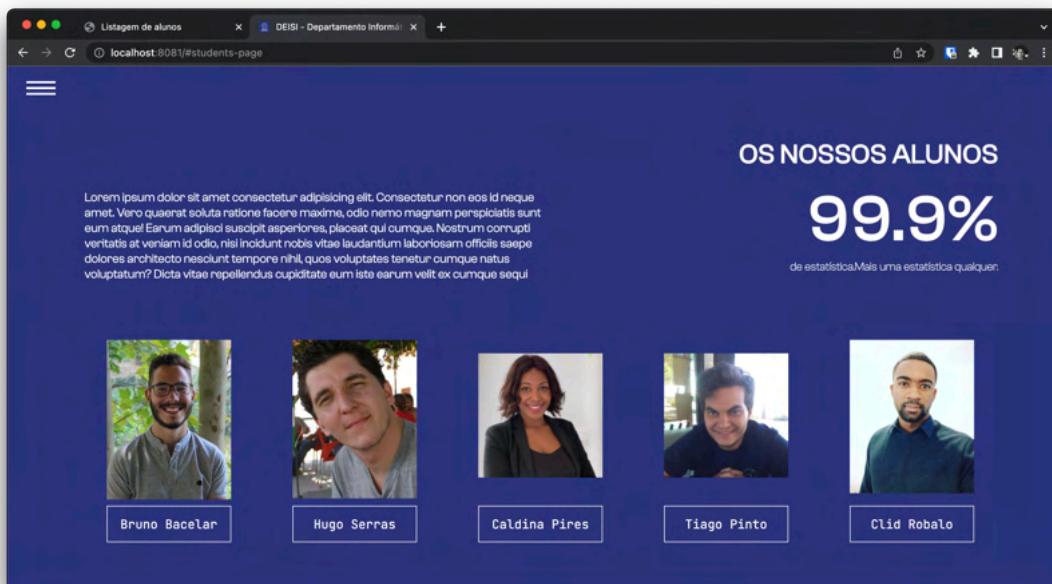


Fig. 31 - Página de Alunos da Solução Final

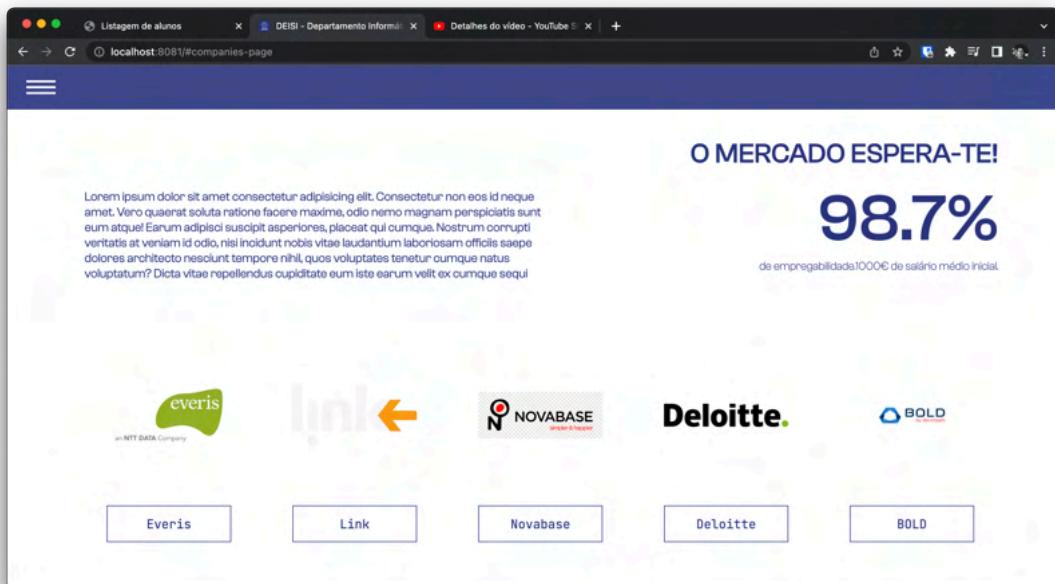


Fig. 32 – Página de Empresas da Solução Final

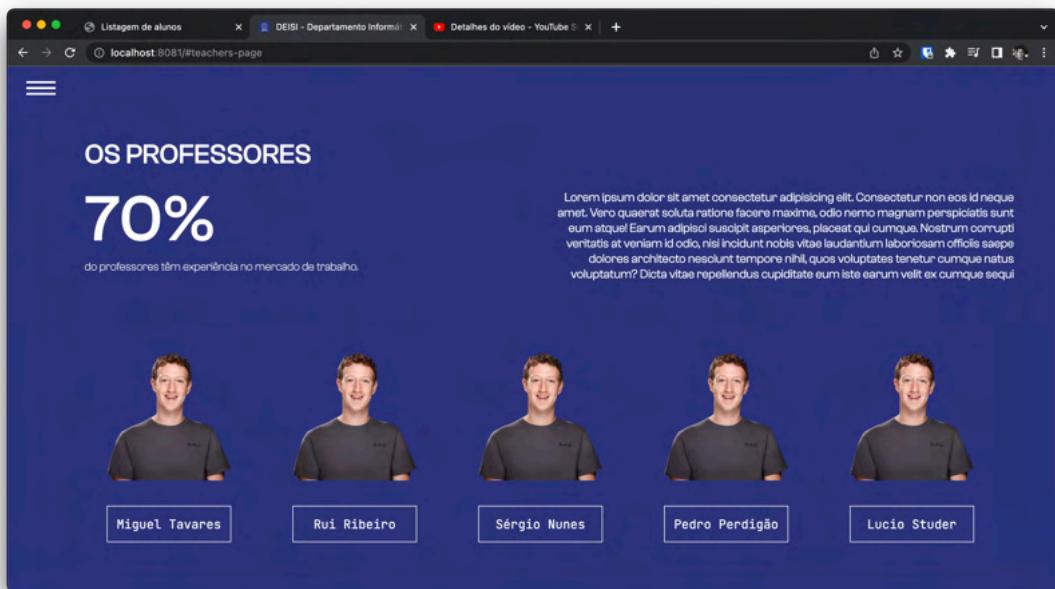


Fig. 33 – Página de Docentes da Solução Final

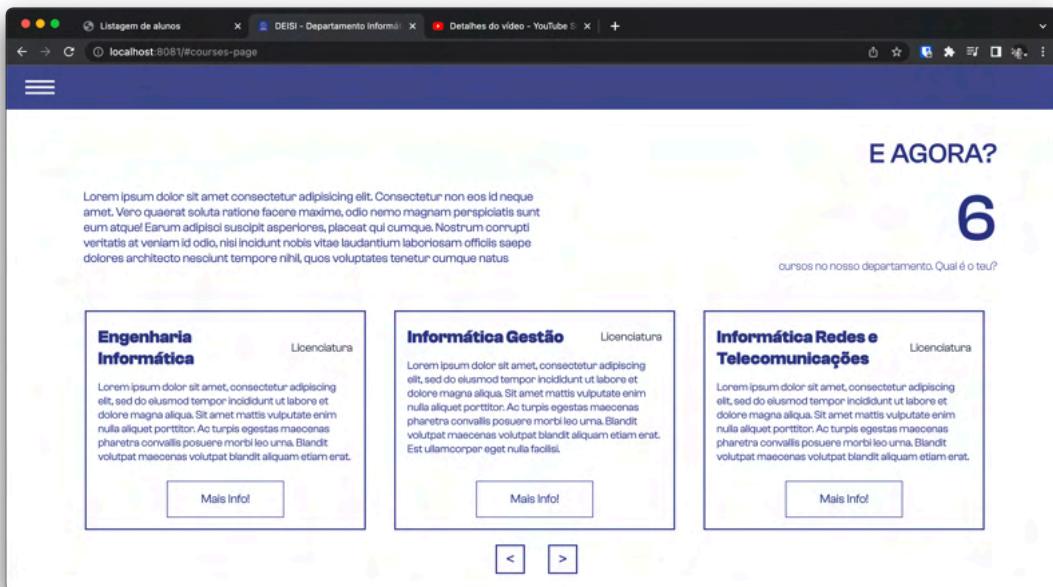


Fig. 34 – Página de Cursos da Solução Final

Também é de notar que, tal como no protótipo inicial, não é possível, através das imagens acima, visualizar o funcionamento de certos elementos e funcionalidades.

O maior desafio deste projeto foi a criação de um back-end totalmente independente de serviços externos por ele utilizados, nomeadamente da CDN onde todas as imagens presentes no website se encontram, o *Imagekit.io*, bem como a criação de componentes *front-end* com o mesmo comportamento, capazes de lidar com diferentes dados.

5. Benchmarking

Para a realização desta análise, foram consultados os *websites* dos seguintes departamentos:

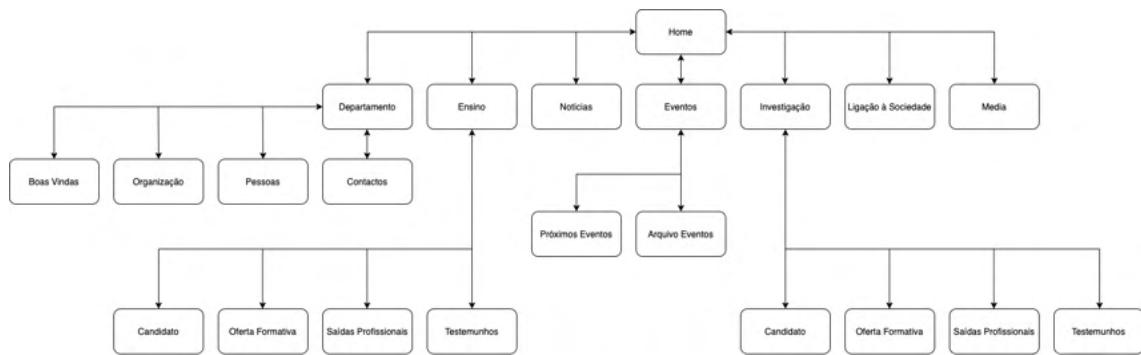
- FCT-UNL - <https://www.di.fct.unl.pt/>
- IST - <https://dei.tecnico.ulisboa.pt/>
- FCUL - <https://ciencias.ulisboa.pt/pt/%C3%B3rg%C3%A3os-5>
- FEUP - https://sigarra.up.pt/feup/pt/uni_geral.unidade_view?pv_unidade=151
- ULHT - <https://deisi.ulusofona.pt/>

Para cada *website*, foi feito um levantamento do mapa do site, assim como uma análise crítica dos seus pontos positivos e negativos.

A. FCT-UNL



Fig. 35 - Homepage Website Departamento Informática FCT-UNL

**Fig. 36 - Mapa Website Departamento Informática FCT-UNL (3 Níveis)**

O website do departamento de informática do FCT é um website que apresenta uma navegação simples e alguns elementos interativos, os quais partilham experiências e trabalhos realizados por alunos. No entanto, é um website com um *design* muito semelhante ao website da faculdade onde se insere e apresenta algumas inconsistências em certas funcionalidades.

B. IST

**Fig. 37 - Homepage Website Departamento Informática Técnico**

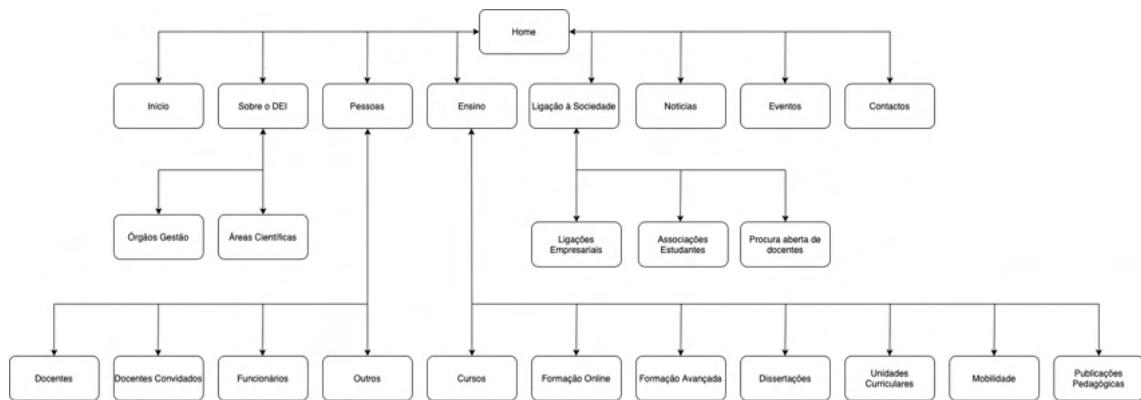


Fig. 38 - Mapa Website Departamento Informática IST (3 Níveis)

O website do departamento de informática do IST é um website que também apresenta uma navegação simples e contém informação atualizada recentemente. No entanto é um website que também apresenta um design muito semelhante ao website da faculdade onde se insere e apresenta alguma informação com maior relevância para docentes e alunos.

C. FCUL



Fig. 39 - Homepage Website Departamento Informática FCUL

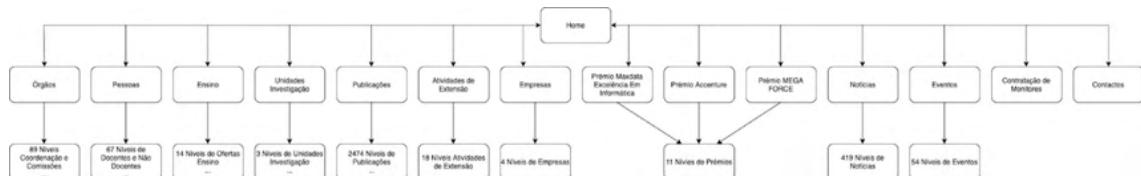


Fig. 40 - Mapa Website Departamento Informática FCUL (3 Níveis)

O website do departamento de informática da FCUL é um website que apresenta informação atualizada recentemente e apresenta bastante informação sobre os cursos que representa. No entanto, é um website com uma navegação extremamente complexa, apresentando algumas inconsistências na mesma e apresenta muita informação com maior relevância para docentes e alunos.

D. FEUP

A captura de tela da homepage do Departamento de Engenharia Informática da FEUP. A barra superior mostra o nome da faculdade e links para "BIBLIOTECA ONLINE", "FORMAÇÃO ONLINE", "FORMATO INSTITUIZIONAL", "Sobre", "Notícias", "Eventos", "Publicações", "Projetos", "Orientações de Projetos / Dissertações / Teses" e "Contacto". O menu principal à esquerda inclui "Boas vindas", "Órgãos de Gestão", "Departamentos", "Serviços", "Estudantes", "Pessoal", "Cursos", "I&D e Inovação", "Cooperação", "Candidatos", "Alumni", "Empresas", "Notícias", "Pesquisa", "Autenticação", "Utilizador", "Senha", "Iniciar sessão" e "Esqueceu-se da senha?". O formulário de login para "Utilizador" e "Senha" está visível. À direita, há uma seção sobre a "Missão" do DEI, que menciona missões internacionais e de investigação, e uma lista de subáreas de pesquisa. Abaixo disso, uma lista de ciclos de estudo e programas doutoriais (PRODEI, PDMOD, MAPPI). A barra lateral direita contém uma seção intitulada "OPÇÕES" com links para "Pessoal", "Investigadores", "Projetos", "Orientações de Projetos / Dissertações / Teses".

Fig. 41 - Homepage Website Departamento Informática FEUP

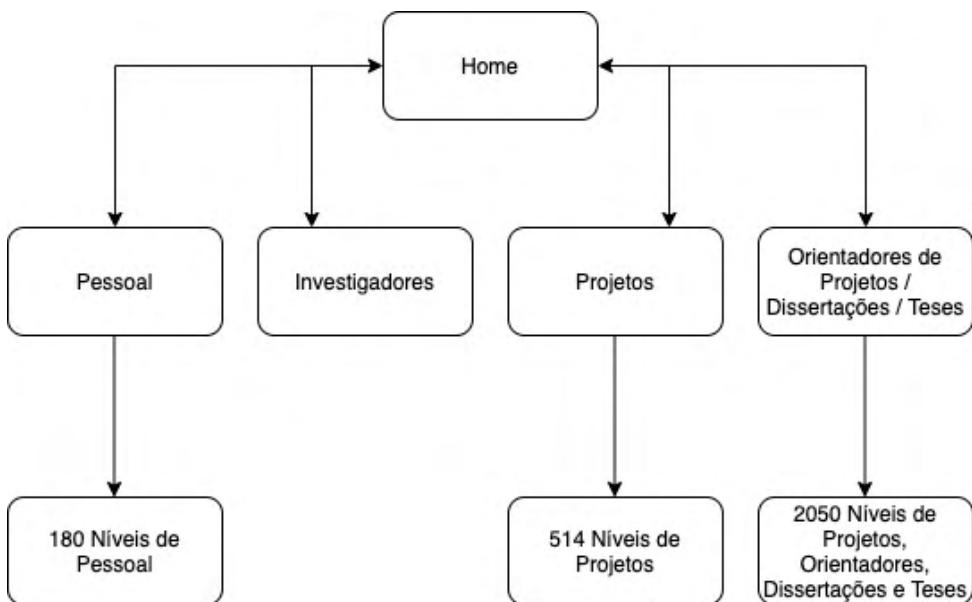


Fig. 42 - Mapa Website Departamento Informática FEUP (3 Níveis)

O website do departamento de informática da FEUP é um website extremamente desatualizado, com um *design* muito antiquado. Apresenta pouca informação relativamente aos cursos que representa e a maior parte da informação que contém é pouco relevante para utilizadores externos.

E. ULHT

A captura de tela da homepage do website do DEISI mostra o seguinte layout:

- Cabeçalho:** Logo da ECAATI (Escola de Comunicação, Arquitetura, Artes e Tecnologias de Informação) e o nome "Engenharia Informática e Sistemas de Informação".
- Navegação:** Barra superior com links para INÍCIO, DEPARTAMENTO, CURSOS, DEFESAS, INVESTIGAÇÃO, NOTÍCIAS, PARCEIROS, CONTACTOS e uma barra de pesquisa.
- Conteúdo principal:** Imagem de estudantes em sala de informática. Abaixo da imagem, uma descrição do DEISI: "O DEISI – Departamento de Engenharia Informática e Sistemas de Informação, faz parte da ECAATI – Escola de Comunicação, Arquitetura, Artes e Tecnologias de Informação de Universidade Lusófona e fornece um currículo de formação em Engenharia Informática e Sistemas de Informação, no fim do qual os candidatos obtêm o Diploma de Mestrado (M.Sc.) em Engenharia Informática e Sistemas".
- Notícias Recentes:** Seção com uma notícia: "7ª conferência – Consenso em Sistemas Descentralizados" (5 Dezembro, 2018 15:33). O resumo da notícia menciona "Abstract: Consensus is a fundamental problem in distributed computing and..." e um link "Ler mais →".
- Logotipo liSS:** Logotipo da "information system school" (liSS) com o texto "escola de sistemas de informação".

Fig. 43 - Homepage Website Atual DEISI

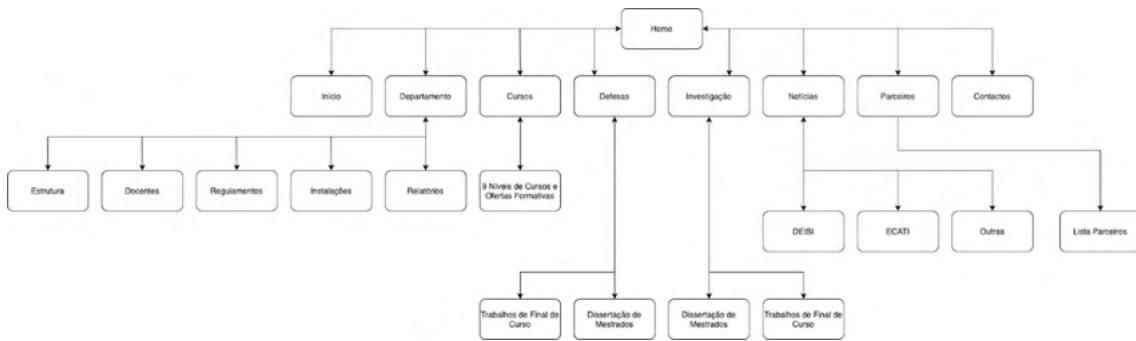


Fig. 44 - Mapa Website Departamento Informática ULHT (3 Níveis)

O website do departamento de informática da ULHT[INULHT21] é um website que apresenta uma navegação simples e informações sobre os cursos que representa. No entanto, é um website que apresenta informação desatualizada e um design antiquado.

5.1. Análise Global:

Todos os websites referidos apresentam algumas características em comum, que podem ser alvo de melhorias:

- Todos estes websites têm um *design* igual/semelhante ao website das faculdades onde os departamentos se inserem. Esta característica pode trazer uma vantagem, a familiaridade com o website caso o utilizador já tenha interagido com o website da faculdade. No entanto, uma interface diferente da interface utilizada em todas as páginas da instituição poderia despertar mais atenção aos utilizadores, podendo aumentar “a vontade” dos mesmos interagirem com o website.
- Alguns websites não apresentam elementos com qualquer tipo de interatividade com o utilizador, isto é, que fizessem com que o utilizador interagisse mais com o website, como por exemplo elementos de vídeo ou até mesmo pequenos jogos com perguntas simples que pudessem despertar mais atenção nos utilizadores. A presença de alguns elementos interativos seria outro fator que poderia aumentar o interesse dos utilizadores.
- Alguns dos websites apresentam um design mais antiquado. A adoção de um design mais moderno poderia melhorar a experiência dos utilizadores.
- Todos os websites apresentam um *dashboard* de informações maioritariamente relevantes para os alunos/docentes da faculdade. A presença de algumas informações relevantes para um público exterior à faculdade seria um fator que poderia aumentar o número de visitas aos websites.
- Alguns websites não apresentam informações sobre os cursos do departamento. A inclusão de uma apresentação dos cursos um pouco diferente da apresentação feita nos websites das faculdades poderia fornecer ao utilizador uma melhor visão dos cursos.

- Em alguns *websites* não existe qualquer tipo de elemento que partilhe as experiências/trabalhos realizados pelos alunos. A presença de certos elementos, como vídeos com o testemunho de alguns alunos, ou a partilha de alguns projetos realizados em diferentes etapas dos cursos poderia fornecer uma pequena ideia do que iria ser trabalhado ao longo dos anos a possíveis interessados nos diversos cursos do departamento.
- Alguns *websites* apresentam uma navegação demasiado complexa, com alguns níveis desnecessários. A simplificação da navegação dos *websites* e a abstração de alguns níveis melhora a experiência do utilizador.
- Alguns *websites* não apresentam certos elementos de acessibilidade que podem ter impacto na interação com utilizadores específicos, como a possibilidade de traduzir o *website* para inglês.
- Alguns *websites* apresentam informação desatualizada. A atualização constante das informações nestes *websites* é crucial para garantir que as informações certas são sempre transmitidas aos utilizadores.
- Alguns *websites* apresentam algumas inconsistências em algumas funcionalidades. Tendo como exemplo o *website* do departamento de informática do FCT, ao clicar na opção de traduzir o *website* para inglês, alguns elementos do *website* alteram-se, como os conteúdos do menu ou alguns elementos de estilo. É necessário garantir a consistência do *website* e o funcionamento correto de todas as funcionalidades, para garantir uma boa e igual experiência para todos os utilizadores.

Em relação ao *website* do DEISI, foram encontrados os seguintes problemas, à semelhança dos *websites* anteriormente analisados:

- Apesar do *design* ser diferente do *website* da faculdade, este está um pouco desatualizado.
- Não existe qualquer tipo de interação com o utilizador.
- O *website* contém um conjunto de informações maioritariamente relevantes para os alunos/docentes da faculdade.
- As interações em todos estes *websites* são pouco animadas.
- Não existe qualquer tipo de elemento que partilhe as experiências/trabalhos realizados pelos alunos.
- A informação existente no *website* está desatualizada.

O reposicionamento desta solução, face ao *benchmarking* feito e face ao cumprimento da maioria dos requisitos definidos ao longo do desenvolvimento irá apresentar um *website* destacável dos *websites* analisados, pois irá proporcionar uma experiência aos utilizadores cujo foco principal foi a resolução dos problemas identificados nos restantes.

6. Método e Planeamento

Na figura abaixo encontra-se a proposta de calendário em formato *Gantt*.

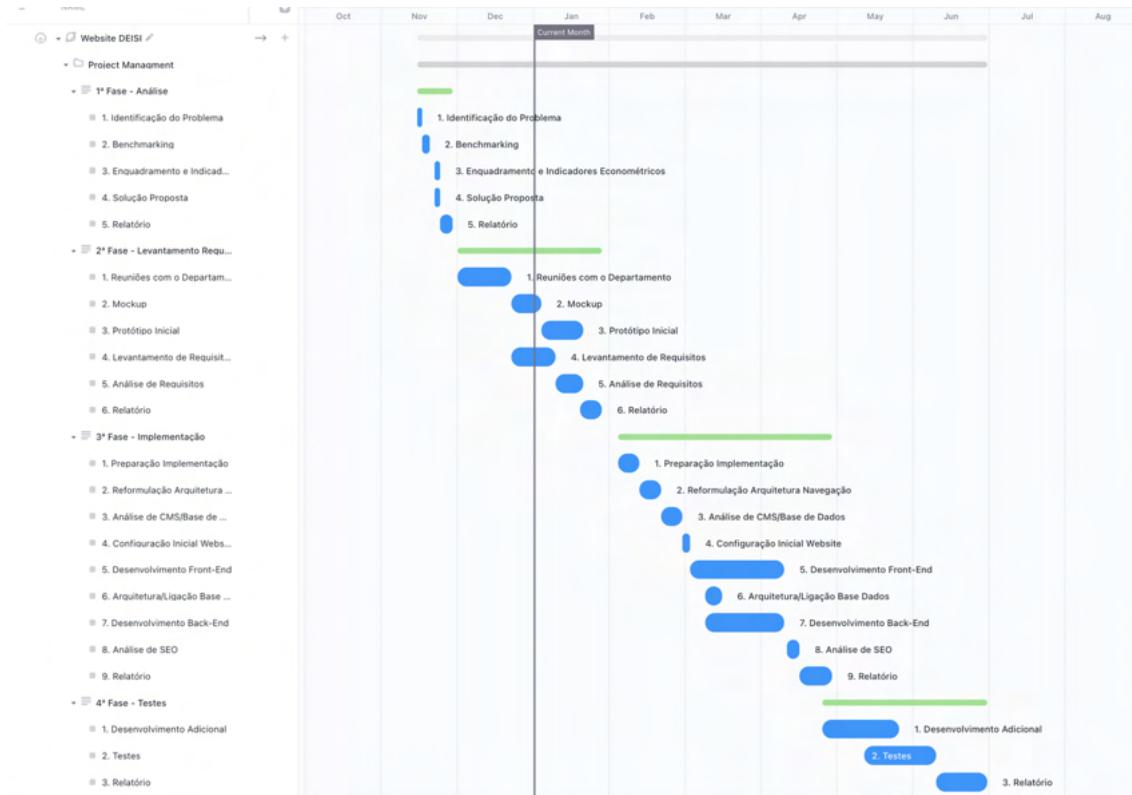


Fig. 45 - Calendário de Execução

O desenvolvimento deste trabalho foi dividido em 4 etapas, correspondentes às quatro entregas agendadas do TFC.

Na primeira etapa, correspondente à fase inicial do trabalho, foi feita a identificação do problema, o *benchmarking*, o enquadramento e indicadores económicos, a solução proposta e foi desenvolvido o primeiro relatório intercalar.

Na segunda etapa, correspondente à segunda fase do trabalho, foi feito, junto do professor coordenador de professores do Departamento de Engenharia Informática e Sistemas de Informação, um levantamento de requisitos e, posteriormente, uma análise dos mesmos. Foi também desenvolvido um *mock-up* seguido de um protótipo inicial do website. Por fim, foi feito o relatório intermédio, resultante da junção do primeiro relatório intercalar com o levantamento e análise de requisitos realizada.

Na terceira etapa, correspondente à terceira fase do trabalho, foi desenvolvida uma grande parte do *website*, desde a planificação da nova versão ao desenvolvimento *front-end* e *back-end*, assim como a análise de *CMS*. Foi também desenvolvido o segundo relatório intercalar, resultante da junção do relatório intermédio com o protótipo desenvolvido e o resultado das análises a ele realizadas.

Na quarta e última etapa, correspondente à fase final do trabalho, foi feito algum desenvolvimento complementar à fase anterior, resultado da identificação de novos requisitos, bem como a resolução de alguns problemas identificados. Foi também desenvolvido o relatório final, resultante da junção do segundo relatório intercalar com o resultado dos testes realizados ao website.

O calendário proposto, presente na Fig. 43, foi cumprido na sua maioria, com exceção de algumas etapas que não foram realizadas, como a análise de SEO ou a realização de testes, os quais foram feitos apenas ao longo do desenvolvimento da solução pelo próprio, dos quais alguns foram validados pelo professor orientador.

7. Resultados

Como referido no Cap. 2, nesta etapa final do projeto, quase todos os requisitos identificados foram implementados, com exceção da responsividade, alguma informação existente no website, como as estatísticas dos alunos e os parágrafos alusivos às páginas onde se encontram e a cache que, apesar de não ter sido implementada no back-end, este está preparado para a sua implementação. Também foi referido que ainda não foi possível resolver os erros existentes pois a utilização de diferentes propriedades para a configuração dos carrosséis pode ter algum impacto na aparência do website, pelo que este erro tem que ser corrigido paralelamente com o desenvolvimento da responsividade.

Relativamente aos requisitos implementados, seguem-se abaixo os resultados:

- 1º requisito - o primeiro requisito foi cumprido na sua totalidade, como é possível verificar da Fig. 26 à Fig. 32 do sub capítulo 4.5, Solução Final
- 2º requisito - o segundo requisito foi cumprido na sua totalidade, como é possível verificar na Fig. 27 do sub capítulo 4.5, Solução Final
- 3º requisito - o terceiro requisito, como referido no Cap. 2, Levantamento de Requisitos, não irá ser implementado, pois torna o *website* pouco compatível com dispositivos móveis
- 4º requisito - o quarto requisito foi cumprido na sua totalidade, estando este presente na *landing page* do *website*, no entanto, não é possível, através de imagens, visualizar o seu funcionamento
- 5º requisito - o quinto requisito foi cumprido na sua totalidade, como é possível verificar da Fig. 27 à Fig. 32 do sub capítulo 4.5, Solução Final, com exceção das estatísticas na página de alunos
- 6º requisito - o sexto requisito foi cumprido na sua totalidade, como é possível verificar da Fig. 29 à Fig. 31 do sub capítulo 4.5, Solução Final
- 7º requisito - o sétimo requisito foi cumprido na sua totalidade, como é possível verificar da Fig. 29 à Fig. 31 do sub capítulo 4.5, Solução Fina
- 8º requisito - o oitavo requisito foi cumprido na sua totalidade, como é possível verificar na Fig. 32 do sub capítulo 4.5, Solução Fina
- 9º requisito - o nono requisito foi cumprido na sua totalidade, pois foram criadas entidades para todos os elementos constituintes das páginas, com um comportamento semelhante ou igual ao demonstrado nas Fig. 20, 21 e 22, no sub capítulo 4.4, *Back-End*
- 10º requisito - o décimo requisito foi cumprido na sua totalidade, como é possível verificar na Fig. 21 à Fig. 31 do sub capítulo 4.4, *Back-End*
- 11º requisito - o décimo primeiro requisito, como referido no Cap. 2, Levantamento de Requisitos, ainda não foi implementado, no entanto, o *back-end* encontra-se preparado para a sua implementação

No segundo relatório intercalar, foram ainda definidos os seguintes testes:

Teste	Passou
Adição dos diversos cabeçalhos das páginas que constituem o website e verificar que os dados são iguais aos dados previamente definidos	<input checked="" type="checkbox"/>
Adição de um novo aluno à lista de alunos e verificar que este foi adicionado ao carrossel de alunos	<input checked="" type="checkbox"/>
Adição de uma nova empresa à lista de empresas e verificar que esta foi adicionada ao carrossel de empresas	<input checked="" type="checkbox"/>
Adição de um novo docente à lista de docentes e verificar que este foi adicionado ao carrossel de docentes	<input checked="" type="checkbox"/>
Adição de um novo curso à lista de cursos e verificar que este foi adicionado ao carrossel de cursos	<input checked="" type="checkbox"/>
Remoção de um aluno da lista de alunos e verificar que este foi removido do carrossel de alunos	<input checked="" type="checkbox"/>
Remoção de uma empresa da lista de empresas e verificar que este foi removido do carrossel de empresas	<input checked="" type="checkbox"/>
Remoção de um docente da lista de docentes e verificar que este foi removido do carrossel de docentes	<input checked="" type="checkbox"/>
Remoção de um curso da lista de cursos e verificar que este foi removido do carrossel de cursos	<input checked="" type="checkbox"/>
Edição de um aluno da lista de alunos e verificar as alterações realizadas no carrossel de alunos	<input checked="" type="checkbox"/>
Edição de uma empresa da lista de empresas e verificar as alterações realizadas no carrossel de empresas	<input checked="" type="checkbox"/>
Edição de um docente da lista de docentes e verificar as alterações realizadas no carrossel de docentes	<input checked="" type="checkbox"/>
Edição de um curso da lista de cursos e verificar as alterações realizadas no carrossel de cursos	<input checked="" type="checkbox"/>
Visualização da descrição de um aluno e verificar que a descrição corresponde à descrição definida para o mesmo	<input checked="" type="checkbox"/>
Visualização da descrição de uma empresa e verificar que a descrição corresponde à descrição definida para a mesma	<input checked="" type="checkbox"/>
Visualização da descrição de um docente e verificar que a descrição corresponde à descrição definida para o mesmo	<input checked="" type="checkbox"/>
Selecionar a opção de “Mais Informações” de um dos cursos e verificar que o utilizador é redirecionado à página oficial do curso, no website da Universidade Lusófona de Humanidades e Tecnologias	<input checked="" type="checkbox"/>

Devido ao desenvolvimento de novas funcionalidades, bem como um pequeno atraso em todo o desenvolvimento desta solução, não foi possível realizar testes a utilizadores. No entanto, todos os testes descritos acima foram realizados pelo próprio, sendo que estas foram validadas pelo professor orientador.

Também é de notar que, apesar de nem todos os requisitos terem sido implementados, esta solução já se encontra em produção, mantendo os erros previamente identificados. É possível aceder ao website através do link <https://deisi.ulusofona.pt/landing-page/>.

8. Conclusão

Como referido no sub capítulo 4.5, Solução Final, maior desafio deste projeto foi a criação de um *back-end* totalmente independente de serviços externos por ele utilizados, nomeadamente da CDN onde todas as imagens presentes no *website* se encontram, o *Imagekit.io*, bem como a criação de componentes *front-end* com o mesmo comportamento, capazes de lidar com diferentes dados. No entanto, após a superação de todas as dificuldades e a implementação desta solução final e com o cumprimento da maioria das metas definidas, penso que o resultado obtido foi bastante positivo, atendendo às expectativas do cliente.

Devido à implementação de novas funcionalidades, bem como um pequeno atraso em todo o desenvolvimento desta solução, não foi possível implementar todos os requisitos definidos inicialmente e ao longo deste trabalho, pelo que, tal como combinado pelo professor orientador, estes irão ser implementados fora do contexto da avaliação deste TFC, de maneira a garantir o principal objetivo deste trabalho: o desenvolvimento de um *website* que visa oferecer aos utilizadores uma melhor e mais clara experiência, com as informações mais recentes relativas ao departamento e respetivos cursos, permitindo, simultaneamente, que o conteúdo seja atualizado com simplicidade e regularidade.

Como também foi referido no Cap. 2, Viabilidade e Pertinência, não é um projeto que se esgote enquanto projeto académico, podendo ser continuado após a conclusão deste TFC, pois irá representar o departamento, tal como a versão atual do *website* representa, até ser suscetível a alterações. Poderá também ser um projeto que, não suscetível a alterações, possa integrar, à sua volta, uma plataforma reformulada, semelhante ao *website* atual do departamento, oferecendo não só informações e vantagens a utilizadores externos e possíveis novos alunos, como também a alunos do departamento.

Bibliografia

- [BABEL22] Documentação Babel.js, <https://babeljs.io/>, acedido em Abr. 2022
- [CHRTJS22] Documentação Chart.js, <https://www.chartjs.org/docs/latest/>, acedido em Jan. 2022
- [DEISI22] Regulamento Trabalho de Final de Curso, https://moodle.ensinolusofona.pt/pluginfile.php/161324/mod_resource/content/1/5BTFC-21.22%5DRegulamento_v.1.pdf, Out. 2021
- [ESLINT22] Documentação ESLint, <https://eslint.org/>, acedido em Abr. 2022
- [FCT21] Departamento Engenharia Informática FCT-UNL, <https://www.di.fct.unl.pt/>, acedido em Nov. 2021.
- [FCUL21] Departamento Engenharia Informática FCUL, <https://ciencias.ulisboa.pt/pt/%C3%B3rg%C3%A3os-5>, acedido em Nov. 2021.
- [FEUP21] Departamento Engenharia Informática FEUP, https://sigarra.up.pt/feup/pt/uni_geral.unidade_view?pv_unidade=151, acedido em Nov. 2021.
- [INULHT21] Departamento Engenharia Informática ULHT, <https://deisi.ulusofona.pt/>, acedido em Nov. 2021.
- [IST21] Departamento Engenharia Informática IST, <https://dei.tecnico.ulisboa.pt/>, acedido em Nov. 2021.
- [PIDEISI22] Protótipo Inicial Site DEISI, https://eduardo-22002197.github.io/deisi_prototype/public/index.html
- [REACT21] Documentação ReactJS, <https://reactjs.org/docs/getting-started.html>, acedido em Nov. 2021.
- [SLICK22] Documentação Slick.js, <https://kenwheeler.github.io/slick/>, acedido em Jan. 2022
- [SPMVC21] Documentação Spring MVC, <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>, acedido em Nov. 2021.
- [ULHT21] Universidade Lusófona de Humanidades e Tecnologias, <https://deisi.ulusofona.pt/>, acedido em Nov. 2021.
- [WBPCK22] Documentação Webpack, <https://webpack.js.org/>, acedido em Abr. 2022

Glossário

API	Application Program Interface
CDN	Content Delivery Network
CMS	Content Management Services
CSS	Cascade Style Sheets
DEISI	Departamento de Engenharia Informática e Sistemas de Informação
ER	Entidade Relação
DOM	Document Object Model
FCT-UNL	Faculdade Ciências e Tecnologias da Universidade Nova de Lisboa
FCUL	Faculdade de Ciências da Universidade de Lisboa
FEUP	Faculdade de Engenharia Universidade do Porto
HTML5	Hypertext Markup Language, versão 5
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IST	Instituto Superior Técnico
JSX	JavaScript XML
LEI	Licenciatura em Engenharia Informática
LIG	Licenciatura em Informática de Gestão
MVC	Model View Content
REST	Representational State Transfer
RESTful	Representational State Transfer
SEO	Search Engine Optimisation
SQL	Structured Query Language
TFC	Trabalho Final de Curso
ULHT	Universidade Lusófona de Humanidades e Tecnologias
VDOM	Virtual Document Object Model
XML	Extensible Markup Language