Upgrade da plataforma CMS EasyWeb

Mestrado em Engenharia Informática - Computação Móvel

Sandro Maia Baptista, n.º 2220653

Leiria, setembro de 2024

Upgrade da plataforma CMS EasyWeb

Mestrado em Engenharia Informática - Computação Móvel

Sandro Maia Baptista, n.º 2220653

Trabalho de Projeto realizado sob a orientação do Professor Doutor Marco António de Oliveira Monteiro.

Leiria, setembro de 2024

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Engenharia Informática - Computação Móvel, no ano letivo 2023/2024, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Dedicatória

Com a realização deste projeto, quero dedicá-lo aos colaboradores da empresa LOBA – escritório Leiria que estiveram envolvidos neste projeto.

Quero destacar as pessoas Eliana Silva (gestora de projeto), Luís Oliveira (programador e criador da plataforma EasyWeb) e José Lopes (gestor de projeto) que seguiram o projeto, fornecendo-me orientação e ouvindo as minhas propostas para implementação de melhorias.

Agradecimentos

Com a realização deste projeto, agradeço a todos os colaboradores registados no escritório Leiria da empresa LOBA que me apoiaram no desenvolvimento das melhorias para o EasyWeb. Os seus contributos fizeram-me compreender a história do EasyWeb, discutir algumas ideias, de forma correta e evolutiva, e socializar.

Quero agradecer ao colega José Lopes, administrador em inCentea Marketing e Inovação e gestor de projeto em LOBA, por me fornecer este projeto para a conclusão do mestrado e me ter confiado para a elaboração de algumas melhorias para a plataforma.

Agradeço ao Professor Doutor Marco António de Oliveira Monteiro por me orientar a escrita deste documento e em esclarecer alguns pormenores técnicos durante o ano letivo.

Também pretendo agradecer aos meus pais por me terem confiado e apoiado durante a formação académica, desde a licenciatura até ao mestrado.

Resumo

Ao longo da utilização de plataformas de gestão de conteúdo para páginas Web, existem processos que se podem tornar repetitivos ou que demoram para serem realizados. Além disso, nem sempre possui funcionalidades internas que facilitem a gestão, envolvendo ferramentas externas.

Este projeto tem o objetivo de criar soluções para problemas encontrados recentemente numa plataforma CMS. As soluções passam por desenvolver componentes relacionados com o carregamento de ficheiros para servidores, gestão de tabelas, organização de ficheiros em servidores FTP e construção de páginas Web.

O componente de carregamento de ficheiros corresponde a um formulário que permite inserir múltiplos ficheiros, oferecendo funcionalidade de *drag-and-drop*. O módulo de organização de ficheiros do servidor FTP é um portal que mostra todos os ficheiros existentes e permite realizar ações com os mesmos. O componente de gestão de tabelas contém a funcionalidade de ordenar as linhas da tabela através de arrasto. O componente de construção de páginas Web corresponde a um editor de páginas embutido.

Durante o desenvolvimento dos diferentes componentes, foram realizados testes na plataforma, já num ambiente preparado para tal, para realizar adaptações e para obter resultados por parte dos utilizadores finais.

Com a implementação concluída, os utilizadores conseguem carregar múltiplos ficheiros em simultâneo mais rapidamente, sem realizar muitos cliques. Também têm a possibilidade de aceder ao servidor FTP sem utilizar ferramentas externas. Além disso, podem ordenar os registos numa tabela através de arrasto e conseguem criar e pré-visualizar páginas num editor.

Palavras-chave: CMS, Componentes, *Drag-and-drop*, Gestão de Ficheiros, Páginas Web, *Upload*

Abstract

When using content management platforms for web pages, there are processes that can become repetitive or time-consuming to carry out. In addition, it doesn't always have internal functionalities that make management easier, involving external tools.

The aim of this project is to create solutions to problems encountered recently in a CMS platform. The solutions include developing components related to uploading files to servers, managing tables, organising files on FTP servers and building web pages.

The file upload component is a form that allows you to insert multiple files, offering drag-and-drop functionality. The FTP server file organisation module is a portal that shows all existing files and allows you to perform actions with them. The table management component contains drag-and-drop functionality for sorting table rows. The web page construction component is an inbuilt page editor.

During development, tests were carried out on the platform, already in a prepared environment, to make adaptations and to obtain results from end users.

With the implementation finished, users can upload multiple files at the same time more quickly, without many clicks. They can also access the FTP server without using external tools. In addition, they can sort records in a table by dragging and can create and preview pages in an editor.

Keywords: CMS, Components, Drag-and-drop, Files Management, Upload, Web Pages

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

API	Application Program Interface
CDA	Content Delivery Application
CMA	Content Management Application
CMS	Content Management System
CSS	Cascading Style Sheets
CV	Curriculum vitæ
DDFU	Drag-Drop-Files-Upload
DDTRS	Drag-Drop-Table-Rows-Sort
ERP	Enterprise Resource Planning
ESTG	Escola Superior de Tecnologia e Gestão
FTP	File Transfer Protocol
HEX	Hexadecimal colors
HSL	Hue, Saturation, and Lightness colors
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IA	Inteligência Artificial
ID	Identificador
IoT	Internet of Things
JSON	JavaScript Object Notation
KB	KiloBytes
MB	MegaBytes
MIT	Massachusetts Institute of Technology
PDF	Portable Document Format
PHP	Hypertext Preprocessor
REST	Representational State Transfer
RGB	Red, Green and Blue colors
SEO	Search Engine Optimization
SVG	Scalable Vector Graphic
TI	Tecnologias de Informação
UI	User Interface

URL	Uniform Resource Locator
UX	User Experience
WYSIWYG	“What You See Is What You Get”

1. Introdução

As plataformas de gestão de páginas Web são importantes para a evolução contínua dos *websites*. Para que as páginas fiquem com conteúdo sempre atualizado, as ferramentas não podem falhar e devem estar preparadas para novos desafios da Internet. No entanto, nem sempre as funcionalidades estão preparadas para novas tecnologias e necessitam de melhorias, para facilitar os utilizadores na gestão de conteúdos e para manter o sistema funcional e sem problemas.

Durante a utilização, os utilizadores podem encontrar problemas nas aplicações, como falhas ao responder ou comportamentos inesperados e erráticos. Esses problemas são registados em listas para análise e resolução pelos programadores.

Um exemplo de falhas em funcionalidades que não são atualizadas regularmente pode ser observado no meu local de trabalho, num produto utilizado em negócios na área Web: o EasyWeb, um Content Management System (CMS) desenvolvido e comercializado pela empresa LOBA. Diante disso, surgiu uma proposta de projeto para a plataforma, com o objetivo de implementar melhorias em funcionalidades cruciais. Os problemas identificados foram levantados em reuniões de *kickoff*, juntamente com algumas ideias preliminares para a sua resolução. Assim, o principal objetivo deste projeto é executar as tarefas definidas numa lista de soluções previamente elaborada. Como a proposta surgiu no momento do registo do tema para a tese de mestrado, a empresa aceitou integrá-la no meu percurso académico e foi realizada uma submissão no início do ano letivo.

Na lista de tarefas para melhorias, destaca-se a implementação de:

- um painel para construção de páginas Web através de arrasto de componentes construídos;
- um explorador de ficheiros para servidores FTP criados;
- carregamento de imagens, utilizando *drag & drop*;
- ordenação de tabelas com o uso de arrasto das linhas.

Ao longo da implementação das melhorias, vários processos tiveram de ser realizados, desde a realização de reuniões, a apresentação dos requisitos, a procura de tecnologias semelhantes, como também a implementação dos componentes na plataforma.

Este desafio pretende ser um contributo para a todas as pessoas envolvida no EasyWeb, visto que as melhorias realizadas podem ser úteis tanto para os programadores como para os clientes e aplicadas em projetos futuros da empresa. Como programador de *back-end* na empresa, tenho esperança de que as melhorias terão sucesso e que novas versões do EasyWeb surgirão, sempre a acompanhar com as tecnologias mais recentes.

Este relatório está dividido em nove capítulos, sendo que o primeiro é esta introdução e o segundo (Estado de arte do CMS) apresenta o estado da arte, em que são apresentados conceitos e tecnologias que estão relacionados com o projeto e em que se observa produtos semelhantes ao EasyWeb. O terceiro capítulo (A empresa e EasyWeb) foca-se na apresentação da empresa e do produto CMS a melhorar neste projeto. O quarto capítulo (Análise de requisitos) está construído para mostrar análise de requisitos. O quinto capítulo (Processo de Desenvolvimento) apresenta a metodologia utilizada para a realização do projeto. O sexto capítulo (Arquitetura e Tecnologias) contém a arquitetura do projeto e as tecnologias existentes. O sétimo capítulo (Implementação) apresenta o desenvolvimento do projeto. O oitavo capítulo (Testes e Resultados) apresenta os testes realizados e a avaliação dos resultados. Por fim, o nono capítulo (Conclusão) apresenta as conclusões e o trabalho futuro.

2. Estado de arte do CMS

Antes de entrar na apresentação dos detalhes deste projeto, o conceito de CMS deve ser exposto sob a forma de contextualização, para se compreender melhor as principais funcionalidades que contém.

2.1. Introdução

A sigla 'CMS' consegue mostrar a tecnologia de *software* envolvida: a gestão de conteúdos aplicada num sistema informático. CMS realça, como objetivo principal, a utilização mais simples e dinâmica na gestão de informações numa plataforma. Direciona-se para os utilizadores que não possuem capacidades em lidar com pormenores técnicos das plataformas comuns, nomeadamente relacionadas com a Web [1].

As aplicações baseadas em CMS são feitas para oferecer uma ampla gama de funcionalidades, reduzindo o número de ajustes manuais necessários. A maioria dessas plataformas inclui ferramentas de edição de conteúdo básico, mas também podem suportar a integração de ferramentas externas por meio de *plugins*. Isso mostra a importância da versatilidade do CMS, pois ele deve ser adaptado continuamente para atender a vários contextos e necessidades.

2.1.1. Exemplos de aplicação de CMS

Os seguintes exemplos conseguem expressar casos práticos de CMS:

- Em vez de depender de equipas especializadas ou indivíduos com habilidades técnicas, a construção de um serviço de gestão de dados para adicionar ou editar artigos de *blog* é realizada por utilizadores comuns. Estes utilizam ferramentas intuitivas, dispensando a necessidade de competências informáticas avançadas;
- Na administração do conteúdo promovido nas redes sociais, os influenciadores não precisam dedicar tempo à inserção manual de publicações e fotografias através de código programado. Em vez disso, eles simplesmente clicam em áreas específicas, e, em questão de segundos, os seguidores têm acesso ao conteúdo mais recente, tornando o processo ágil e desprovido de complexidades técnicas.

Com base nos exemplos apresentados, os sistemas de gestão de dados são amplamente adequados e utilizados nos seguintes casos [2]:

- Edição de páginas Web;
- Lojas *online* e comércio eletrónico;
- Controlo de conteúdos digitais.

Edição de páginas Web

Conseguir controlar um sítio Web, ao nível do conteúdo abordado, com alguns cliques, isto é, que não necessite de uma equipa de programação para publicar manualmente, é considerado uma característica que marca um negócio, principalmente na área da digitalização [2]. Este aspeto é pensado desde a origem do conceito de CMS, mantendo-se até aos dias atuais.

Portanto, várias empresas de desenvolvimento de *software* foram construindo plataformas, com uma interface mais amiga dos utilizadores, que cumprissem os objetivos principais do CMS. Com ferramentas que tornam o processo de inserção mais simples e agradável, como a utilização de *drag-and-drop* ou de formulários para preenchimento de informação, os utilizadores não necessitam de realizar inúmeras formações sobre a utilização das plataformas [2].

No processo de construção de páginas Web, as plataformas fornecem alguns *templates*, com possibilidade em diminuir o tempo investido. Os *templates* podem ser constituídos por várias secções, não havendo a obrigação em recriar partes de páginas. Além disso, estão totalmente adaptados para diferentes formatos do ecrã [2], aplicando, de certo modo, a responsividade.

Além de conter funcionalidades associadas ao conteúdo, CMS também oferece automatismos que favorecem as otimizações [2] e, por conseguinte, a qualidade de construção dos *websites*. Estes sistemas gerem, automaticamente, situações associadas a *Search Engine Optimization* (SEO), na velocidade de carregamento, entre outros.

Lojas online *e comércio eletrónico*

O sistema *e-commerce* torna-se mais simples de ser bem gerido ao aplicar um CMS como estrutura de base. Algumas plataformas CMS foram desenvolvidas justamente para estar totalmente focadas no controlo de processos de compras *online* ou na gestão de inventários [2]. Com a utilização de integrações com entidades de pagamento, os atos de compra ficam mais práticos e menos complexos para os clientes, estando sempre informados do

ponto de situação das encomendas. O envio de notificações por *e-mail*, uma das técnicas usadas para alertar os clientes, é tratado automaticamente pelo próprio CMS.

Controlo de conteúdos digitais

Uma das vantagens do uso de CMS é a programação de tarefas. Por exemplo, a definição de momentos exatos para publicações de informação permite ao utilizador da plataforma de gestão ter mais controlo do seu tempo. Desta forma, não necessita de estar naquela específica hora a ativar publicações. Combinando as funcionalidades de criação de estruturas através de construção de secções para páginas Web, da gestão das informações inseridas e da forma como podem ficar visíveis ao público, um criador de conteúdo apenas precisa de adicionar o que pretende, porque todas as outras tarefas acabam por estar automatizadas.

2.1.2. História

Para compreender o percurso que os sistemas de gestão estão a seguir, é interessante conhecer a sua história, porque existiram mudanças na maneira como inicialmente foram imaginados e implementados.

CMS testemunhou um aumento significativo de popularidade ao longo das últimas duas décadas, possibilitando uma gestão mais controlada da informação em diversas plataformas. A sua utilidade não se restringe apenas a *websites*, mas também se estende a aplicações móveis, aplicações para computador, e outras soluções relacionadas ao conceito de CMS.

Começando na década de 2000, um dos pioneiros, embora inicialmente não rotulado como CMS, foi o Blogger [3] [4], que gerenciava o conteúdo de artigos de *blog* na plataforma, permitindo aos utilizadores apresentar informação com estruturas personalizadas e a capacidade de selecionar imagens complementares ao texto. No ano de 2003, o WordPress [5] surgiu, oferecendo funcionalidades avançadas de personalização de *websites*, incluindo a instalação de *plugins* específicos para resolver situações particulares. A sua popularidade cresceu rapidamente, competindo com outras plataformas, como Drupal [6] e Joomla [7].

Os CMS, tradicionalmente, eram desenvolvidos em PHP, facilitando a integração com o ambiente Web. No entanto, com surgimento de novas linguagens, como JavaScript, Python ou Ruby, PHP deixou de ser a única a estar envolvida. O desafio residia na comunicação entre um CMS, geralmente desenvolvido em PHP, e uma aplicação em JavaScript, por exemplo. A solução consistiu na criação de API que serviriam como intermediários, possibilitando

a comunicação eficaz entre as plataformas. Os CMS *headless*, que apostavam numa plataforma *back-end* independente do *front-end* e que ofereciam gestão de dados e consultas através de API, surgiram como resposta, utilizando arquitetura REST ou, em alguns casos, GraphQL pela sua flexibilidade.

Recentemente, a inteligência artificial e o *machine learning* têm impactado estes sistemas. A automatização de processos tornou-se uma prioridade, proporcionando uma maior rapidez e precisão no controlo de dados e da própria plataforma. As publicações automáticas em horários específicos, a geração de conteúdo e o uso de *chatbots* são exemplos de processos que podem ser aplicados. Contudo, a incorporação da inteligência artificial requer sempre uma abordagem cuidadosa, principalmente no acesso total da informação alocada.

Atualmente, observa-se um acréscimo do mundo interligado ao longo dos anos: o *online* cada vez está mais presente nas vidas pessoais e nos negócios. Em situações empresariais, se existir a intenção de conquistar mais consumidores e, por conseguinte, vender mais os seus produtos, é necessário cativar com funcionalidades que se diferenciam da concorrência e que causam, de certo modo, grande destaque. Uma maior flexibilidade e robustez para diversos ambientes são algumas das principais propostas que os utilizadores finais preferem, porque permite utilizar o mesmo sistema em qualquer cenário possível.

2.2.Arquitetura e Estrutura

A flexibilidade e a escalabilidade desafiam os CMS. Nos tempos atuais, uma plataforma pode ser operada em diversos dispositivos, e, para que aconteça sem obstáculos durante a utilização, é necessário que a plataforma funcione em qualquer dispositivo, desde que tenha os requisitos exigidos.

A evolução tecnológica tem influenciado a forma como os CMS são montados. Inicialmente, eram usados principalmente para gerir o conteúdo digital em *websites* específicos. As aplicações nativas nos computadores possuíam as suas próprias bases de dados internas. Mais tarde, com a aplicação de tecnologias que envolvem a comunicação com a Web, a sincronização e a existência de sistemas que integram plataformas distintas foram inseridas na estrutura central de um CMS.

De facto, com a adição de API e outras ferramentas tecnológicas, várias arquiteturas foram desenvolvidas, com o intuito de resolver, parcialmente ou totalmente, as limitações existentes quando CMS era relativamente recente.

Uma arquitetura CMS focaliza-se, principalmente, na comunicação entre os sistemas *front-end* e *back-end* [8]. O *front-end* tem a missão de apresentar os dados e estruturas que são controlados no *back-end*. Em muitos casos, uma aplicação de *back-end* pode servir apenas uma plataforma de *front-end*, cujo exemplos podem ser páginas Web institucionais. Na verdade, um painel de controlo de conteúdo deve permitir disponibilizar várias aplicações cliente em simultâneo, para aproveitar o mesmo conteúdo e para economizar tempo.

Atualmente, estão definidas quatro modelos de arquitetura de CMS [8]. Cada detalhe está descrito nos próximos subcapítulos. São os seguintes:

- Tradicional;
- *Decoupled*;
- *Headless*;
- Híbrida.

2.2.1. Arquitetura CMS tradicional

A arquitetura tradicional foi a primeira a ser utilizada nos CMS. Consiste na agregação dos sistemas *front-end* e *back-end*, podendo a arquitetura também ser conhecida por monolítica ou *coupled*. Desta forma, o *back-end* não fornece dados para outras aplicações cliente, o que mostra a existência de um CMS totalmente adaptado somente para o seu *front-end* [8].

Os projetos que não exigem complexidade no desenvolvimento e manutenção funcionam corretamente se utilizar esta arquitetura. A sua construção depende das funcionalidades pedidas pelos clientes, a interface torna-se mais simples e o sistema inclui o *front-end* e *back-end*, facilitando a sua instalação. Efetivamente, os produtos CMS tradicionais tendem a ser mais baratos [8].

Devido à forte ligação entre o *back-end* e o *front-end*, o conteúdo gerido pode ter dificuldades em ser utilizado noutras plataformas. Além disso, existe uma maior probabilidade de o CMS ter de ser constantemente atualizado, devido a pedidos dos clientes para novas

funcionalidades, visto que o programa foi desenvolvido com funcionalidades totalmente específicas. As modificações podem causar riscos de segurança e de funcionamento do próprio CMS [8].

2.2.2. Arquitetura CMS *decoupled*

A arquitetura está estruturada num formato completamente diferente da arquitetura monolítica. A separação entre o *front-end* e o *back-end* é mais notória e a comunicação entre ambos é realizada através de uma API. O *back-end* apenas tem o propósito de guardar os dados, enquanto o *front-end* os interpreta [8]. Isto permite a coexistência de diferentes *front-ends* associados no mesmo *back-end*, desde que consigam comunicar com a API. Com isto, os dois sistemas são independentes entre si, o que permite uma maior facilidade na manutenção e na escalabilidade.

Para a implementação desta arquitetura, o *back-end* tem a responsabilidade de, para além de gerir os dados armazenados nas suas bases de dados, disponibilizar uma integração consistente em API, que garanta a entrega dos dados para todas as interfaces que apresentam os dados. O *front-end*, por sua vez, deve preparar o sistema para a ligação com a API e interpretar os dados recebidos, para que possam ser apresentados ao utilizador.

A aplicação desta arquitetura traz benefícios para a equipa de desenvolvimento. Relativamente a API, a equipa pode escolher a arquitetura de API que melhor se adapta ao projeto, visionando a otimização e experiência do utilizador. Qualquer modificação no *back-end* não traz consequências para o *front-end*, desde que a API não seja alterada. A segurança também se encontra aplicada nesta arquitetura, em que a API tem uma grande responsabilidade no controlo de acessos ao *back-end* [8]. Outro ponto interessante é o controlo de tráfego de dados, que é realizado através da API. Este modelo permite a aplicação de *clustering*, podendo resolver problemas de desempenho e de entrega de dados.

No entanto, são os custos e a integração com plataformas de terceiros que podem ser considerados os pontos fracos desta arquitetura. Por conter um número considerável de funcionalidades para todos os dispositivos, existe a necessidade de construir manuais de utilização do CMS, para que os utilizadores possam conseguir utilizar todas as funcionalidades [8].

2.2.3. Arquitetura CMS *headless*

O modelo *headless* é considerado uma evolução do modelo *decoupled*. A sua principal diferença é a não predefinição do *front-end* que o CMS se destina. No *back-end*, realiza-se a gestão dos dados e também a estrutura onde o conteúdo é mostrado na aplicação cliente. As composições das páginas, com a informação incluída, são transmitidas através de API para as plataformas de *front-end* associadas.

O CMS *headless* torna-se uma solução mais flexível que o *decoupled*, porque os administradores podem escolher a forma como as secções são apresentadas, sem que seja necessário realizar alterações no *back-end* e na API. Além disso, a equipa de desenvolvimento pode escolher a tecnologia que melhor se adapta ao projeto, o que significa que estes CMS estão preparados para equipar futuras tecnologias. Isto permite que o CMS seja ainda mais escalável.

Quanto mais flexível é o CMS, mais complexo se torna a sua utilização. No caso de empresas, o custo de manutenção pode afetar o orçamento para este tipo de sistemas, porque tendem a ter preços mais exorbitantes. Além disso, é imprescindível uma equipa de desenvolvimento totalmente capaz de gerir o CMS. No caso da apresentação, a equipa especializada frequentemente utiliza JavaScript para a construção de *interfaces*, o que pode ser um problema para utilizadores que não tenham conhecimentos nesta linguagem de programação [8].

2.2.4. Arquitetura CMS híbrida

A abordagem híbrida em CMS combina as funcionalidades das arquiteturas *headless* e *decoupled*, criando assim um modelo que se beneficia das duas abordagens. Utilizam o conceito de API para facilitar a comunicação e transferência de dados do *back-end*, enquanto incorporam as capacidades *decoupled* para aumentar a flexibilidade e personalização do conteúdo a ser apresentado [8].

Essa arquitetura isola o *front-end* do *back-end*, permitindo que os programadores criem aplicações Web utilizando diversas tecnologias, sem uma imposição de *front-end* predefinido. A execução de processos em tempo real torna-se possível através do uso de API, proporcionando aos utilizadores finais a capacidade de gerir informações por meio de editores de conteúdo em tempo real [8] [9].

No entanto, a fusão dessas duas arquiteturas CMS também traz consigo desafios, como a alta dependência da equipa de TI responsável pelo produto CMS, o que pode resultar em complexidade adicional, além dos custos associados à própria arquitetura.

2.3.Funcionamento

Os sistemas de gestão de conteúdo (CMS) têm a capacidade de interagir diretamente com o servidor e gerir os ficheiros que compõem uma aplicação Web. CMS centraliza o controlo de pastas que armazenam conteúdo multimédia, como imagens e vídeos, em arquiteturas tradicionais, tornando mais fácil aceder esse conteúdo pelas páginas Web. No entanto, com a evolução de arquiteturas como os CMS *headless*, a gestão desse tipo pode ser feita através de API que ligam o sistema a vários serviços de armazenamento *cloud* ou externos. Assim, gerir e organizar bibliotecas multimédia é importante tanto para CMS tradicionais quanto *headless* para garantir um controlo eficaz e acessível de todo o conteúdo.

Os sistemas de controlo não estão preparados somente para acolher texto e imagens e facilitar na comunicação entre o utilizador e o servidor. Também deve ter a capacidade de construir exatamente a estrutura que esse conteúdo deve estar numa página Web [1]. Isto implica a estruturação integral de blocos específicos ou, num nível mais complexo, das próprias páginas.

Os detalhes descritos demonstram que um CMS é composto em duas partes [9]:

- Aplicação de gestão de conteúdos (CMA) – a secção onde existe a possibilidade de observar as informações guardadas nas bases de dados, sob a forma de tabelas ou outras apresentações;
- Aplicação de gestão de entrega (CDA) – local onde se desenvolve estruturas para as páginas Web, utilizando o conteúdo inserido, e é responsável em permitir o envio para sítios Web que as expõem.

A automatização é um fator chave que se destaca num CMS. De facto, uma simples alteração consegue modificar vários *scripts* de código informático instantaneamente, não necessitando de visitar cada um dos ficheiros que albergam a secção a ser alterada. A simplificação e uma interface visual mais apelativa e intuitiva descrevem as aplicações para gestão de conteúdos.

Na gestão de um CMS, alguns utilizadores ficam com a sua conta registada como administradores. No entanto, os indivíduos que apenas estão destinados a gerir o conteúdo não devem ter acesso a toda a plataforma. Isto significa que sistemas de permissões também são aplicados, controlando os acessos às contas registadas e aumentando a segurança do *back-end*. A utilização de regras para os utilizadores não se destina às contas registas dentro do CMS, mas também no lado do sítio Web, principalmente no acesso a áreas reservadas [10].

2.4.UX e UI

Os CMS manipuláveis pelos utilizadores necessitam de uma *interface*, com a intenção de facilitar a gestão dos dados. Um aspeto com contornos simples e que oferece um visual mais conciso são pormenores que todas as plataformas devem ter em conta. Por isso, interface de utilizador (UI) e experiência de utilizador (UX) também estão envolvidos nos sistemas de gestão de dados, não apenas na edição de páginas, como também na utilização do próprio CMS.

A experiência do utilizador é um fator importante para as *interfaces* das aplicações. Desta forma, pode-se afirmar que UI tem mais sucesso quando tem uma UX consistente e forte [11], e os *designers* possuem conteúdo capaz de produzir um visual mais atrativo, agradável e funcional para todas as situações.

Para a edição de páginas Web, um CMS contém vários pacotes de componentes. Foram desenvolvidos com o objetivo de evitar a construção total dos elementos Web. Além disso, estes estão adaptados com técnicas de UX, o que traz mais facilidades na aplicação das páginas em qualquer ambiente. "[…] o trabalho árduo em projetar UX foram trabalhadas no CMS […]" [11], mostrando que a personalização pode ser uma tarefa mais simples e rápida.

2.5.Suporte a multimédia

O conteúdo audiovisual é muito utilizado em plataformas Web, para representação visual e descritiva do conteúdo. Em páginas que apesentam instituições, as imagens e vídeos completam a explicação desse conteúdo. As lojas *e-commerce* guardam inúmeras imagens que apresentam os seus produtos registados.

Os CMS são vitais para a gestão de multimédia. A constituição destes sistemas permite encontrar diversas soluções que facilitam a integração com outras plataformas de conteúdo.

Uma das soluções mais utilizadas são as bibliotecas de multimédia embutidas. Nestes casos, todo o conteúdo audiovisual fica localizado na sua secção e pode ser acedido em todo o *website*, oferecendo a reutilização dos recursos, nomeadamente no espaço de armazenamento onde os ficheiros associados estão inseridos, e um melhor controlo. A galeria pode estar organizada em diferentes partes, de acordo com tipo de multimédia importado. Algumas funcionalidades, como a edição de imagens ou vídeos, também podem ser adicionadas no CMS, enriquecendo o poder dos utilizadores quanto ao controlo deste tipo de conteúdo Web. Outras ferramentas associadas à multimédia também interessantes é a habilidade em adaptar as imagens, por exemplo, para casos mais específicos. A utilização de redimensionamento automático pode ser útil para a seleção de uma figura para ícone ou para situações relacionadas com SEO.

Quando o CMS tem a sua própria galeria de multimédia diretamente inserida na plataforma, os ficheiros estão alocados dentro das pastas da aplicação *back-end*, mostrando que não existe uma barreira entre a sua estrutura principal e os ficheiros de multimédia.

Para resolver o problema de acumulação de conteúdo multimédia dentro das pastas do *back-end*, uma solução eficaz é a integração com outros serviços, principalmente externos, que conseguem armazenar ficheiros. Desta forma, a distinção é mais notória e existe uma maior independência, fornecendo condições de gestão tanto no CMS como na aplicação externa do serviço associado.

Os serviços disponibilizados em *cloud* mais comuns para armazenamento de ficheiros são adequados para as ocorrências anteriores, desde que ofereçam ferramentas para a comunicação com o CMS.

2.6. Integrações e extensões

Um CMS é construído com funcionalidades essenciais para a gestão de páginas Web. No entanto, nem sempre é considerado utilizar todas as ferramentas existentes, porque depende do contexto de todo o sítio Web. Portanto, existir módulos que são opcionais não apenas favorece quanto ao espaço de armazenamento que a plataforma ocupa, como também torna menos confusa a utilização do CMS.

São comuns as integrações com serviços externos, como acontece no caso da multimédia, estarem associados a extensões dos CMS, porque a maioria dos projetos não as necessitam. Além disso, também algumas ferramentas desenvolvidas pela equipa de desenvolvimento do CMS podem ser dispensadas durante a preparação da plataforma.

Cada CMS contém os seus próprios *plugins*, dependendo das linguagens de programação utilizadas e da estrutura adotada pelos sistemas de gestão. Isto apresenta a dificuldade na reutilização do mesmo módulo para outras plataformas de controlo de informação, obrigando a criar *wrappers* ou novas bibliotecas.

A existência de uma zona de *marketplace* para extensões, no interior do CMS, é uma das técnicas para a gestão destas ferramentas adicionais, conseguindo atualizá-las sem dificuldades, o que permite evitar o uso de linhas de comando para instalar bibliotecas.

Alguns sistemas de gestão de conteúdo permitem a adição de funcionalidades criadas pela comunidade e disponibilizadas na loja das extensões. Para que aconteça, são fornecidos materiais auxiliares que explicam as regras para a integração com o CMS. A utilização de *frameworks* para desenvolvimento de módulos são essenciais e facilitam o trabalho dos programadores.

As extensões não podem funcionar somente para completar o funcionamento do CMS. A aparência da i*nterface* das páginas Web ou da própria plataforma também podem ser personalizadas com pacotes de temas ou de *templates*.

Quando se desenvolve um componente adicional, deve-se ter em conta as tecnologias aplicadas no próprio CMS. Para comunicação com o exterior, API é adequado. Estas podem adotar várias arquiteturas, como o RESTful ou GraphQL, para realizar operações de leitura, escrita ou remoção. É importante considerar os meios para acontecer a comunicação por API, porque existem *frameworks* que facilitam todo o processo. Se as *frameworks* estiverem relacionadas com o *back-end*, melhora na organização do projeto e na reutilização das funcionalidades existentes. *Webhooks* também é visto como ótima ferramenta de integração com o exterior, visto que suporta comunicação em tempo real e interage, através de notificações, com outras aplicações sobre mudanças no sistema.

Com a utilização de diferentes técnicas de interligação com CMS, as extensões podem variar entre a interação com redes socias, o acesso a sistemas *e-commerce*, a gestão de *marketing*, entre outras.

2.7.Desempenho e Escalabilidade

Antecipar o futuro é essencial para a vida dos CMS. Não existir planos que permitem a adaptação constante relativamente a mudanças e melhorias tecnológicas pode colocar em risco a manutenção do CMS, obrigando, por vezes, a realizar modificações extremas ou até optar por construir algo de raiz capaz de resolver os problemas encontrados. Por isso, a escalabilidade e o desempenho têm um papel preponderante nos CMS e não devem ser esquecidos quando se está a escolher um adequado para as suas necessidades.

2.7.1. Desempenho

Ter um sistema que responde a todos os pedidos e que realiza controlos de tráfego consegue diminuir a ocorrência de erros sistemáticos e aumentar o desempenho e execução de tarefas nos CMS [12].

Existem vários pontos notórios relativamente ao desempenho e flexibilidade destes sistemas, destacando os seguintes:

- Arquitetura do CMS;
- Bases de dados e distribuição de conteúdo;
- Otimizações;
- Monitorização.

Arquitetura do CMS

As arquiteturas são projetadas com a intenção de melhorar a gestão e comunicação e reduzir falhas causadas durante a utilização. Algumas dependem da aplicação de serviços externos, obrigando a criar soluções que otimizam as consultas de dados [13].

Bases de dados e distribuição de conteúdo

Em diversos CMS, os dados podem ficar guardados em ficheiros de notação ou em bases de dados capazes de os controlar. A maioria da informação inserida é armazenada nesses locais, o que obriga a apostar em soluções que permitem melhor *performance* durante as consultas e, por conseguinte, uma maior rapidez no carregamento dessa informação obtida nas páginas Web [12] [13] [14].

A distribuição de conteúdo para diferentes localizações geográficas possibilita melhores velocidades de comunicação e uma menor latência, ao procurar pelo servidor mais próximo [12].

Otimizações

Em ocasiões específicas, o carregamento de ficheiros de grandes dimensões pode demorar alguns segundos e o utilizador é obrigado a esperar até que o processo fique concluído. Isto acontece não apenas no *upload* de conteúdo, como também no carregamento nos *websites*, em que fica apresentável depois de terminar todos os procedimentos de consulta de dados.

É essencial procurar formas que permitem o carregamento mais simples e rápidos, evitando a sobrecarga de pedidos e oferecendo uma boa experiência de utilizador [14]. Por exemplo, a utilização de *cache* é uma técnica que procura uma entrega de conteúdo mais rápida e menos acessos aos servidores [14].

Monitorização

Os CMS não devem possuir apenas ferramentas de gestão de conteúdo, mas também um painel de controlo do próprio sistema. As monitorizações são importantes para o desempenho, facultando melhores maneiras de configurar de acordo com as necessidades.

2.7.2. Escalabilidade

A escalabilidade dos CMS pode ser aplicada em duas maneiras similares: horizontalmente ou verticalmente. A primeira foca-se na expansão através de novos servidores ou instâncias, enquanto a segunda segue a ideologia da melhoria de *hardware* do único servidor que aloja o sistema [13].

Várias estratégias foram pensadas para cumprir os desafios da escalabilidade dos sistemas de gestão de dados. A aplicação de arquiteturas sem estado, que não necessitam de servidores e que se baseiam em serviços micro, e a utilização de *edge computing* são várias das propostas exploradas pelos especialistas para reduzir a latência e o efeito *bottleneck* na comunicação [13].

2.8.SEO e Acessibilidade

Para que um sítio Web se destaque entre os demais durante pesquisas na Internet, vários mecanismos são acionados, em que certas otimizações são montadas a pensar nos motores de pesquisa, vulgarmente conhecidas por SEO.

As plataformas CMS estão preparadas para a aplicação do SEO. A utilização de imagens e a capacidade constante em publicitar conteúdo favorecem as pesquisas, porque os motores de pesquisa procuram por informação mais recente [15].

Outras técnicas, como a utilização de *snippets*, definição de palavras-chave, otimização das imagens ou a adaptação dos *websites* para dispositivos móveis são facilmente aplicáveis a partir dos CMS, utilizando formulários como meio para inserir estes dados nas páginas Web [15].

Quanto à acessibilidade, nunca se deve esquecer os utilizadores com limitações nas suas capacidades. É vital que qualquer página esteja adaptada para estas situações, para reduzir drasticamente a exclusão social e aumentando o sucesso empresarial. Existem várias normas de acessibilidade Web que vários países implementaram e que são extremamente importantes para a concretização destas metodologias [16].

Um CMS acessível para todos gera conteúdo que seja percetível, operável, compreensível e robusto [16], podendo ser colaborado por ferramentas específicas fornecidas pelos navegadores Web.

2.9.Tendências e Inovações

O acompanhamento das novidades tecnológicas é importante para o futuro dos CMS. Não estar ciente da atualidade e das tendências futuras mostra o declínio constante, fazendo diminuir a popularidade face aos concorrentes e, mais tarde, descontinuar o produto.

Desenvolver soluções inovadoras tornam os CMS mais avançados e melhores relativamente aos demais, ocupando os lugares de topo quanto à tecnologia aplicada. Desta forma, outros produtos começam a aplicar funcionalidades semelhantes, aumentando a concorrência e a oferta de soluções robustas.

Das tendências tecnológicas associadas a CMS, destaca-se, principalmente, a implantação de inteligência artificial generativa. Esta tecnologia tem ganhado robustez nos últimos anos, em que várias gigantes tecnológicas estão a desenvolver e melhorar produtos que envolver este tipo de Inteligência Artificial (IA). Atualmente, estão a ser realizados testes de criação de conteúdo textual e visual para *landing pages* utilizando várias frases de linguagem natural. O texto gerado também pode ser editado, para se adaptar melhor ao contexto. Outro exemplo associado representa as recentes ferramentas de conversão de texto integral para diapositivos de apresentação [17] [18].

As aplicações que criam conteúdo artificial necessitam de cumprir normas para minimizar vários problemas complexos. A inteligência artificial deve estar em conformidade com normas quanto à confidencialidade dos dados. Nem toda a informação guardada deve ser acedida por todos, e, por isso, a IA tem de estar ciente destas situações e aplicar limitações a pessoas que estão restringidas a certo tipo de conteúdo. Falhas nos protocolos de confidencialidade podem tornar a ferramenta IA mais problemática do que benéfica [17].

Depois da inteligência artificial, também se destacam as habituais automatizações de processos de controlo de informação. A criação de procedimentos automáticos permite utilizadores com pouca ou nenhuma experiência naquele setor a criar sequências de trabalho mais simples, com o objetivo de anular a codificação de processos. Para ajudar, as funcionalidades são compostas por *interfaces* mais limpas e por técnicas que conseguem evitar o uso do teclado, como o arrasto de objetos [17].

A computação por *cloud* continua a ser uma das tendências mais populares nos CMS. A integração com outros serviços *cloud* permite o crescimento mais acessível para estas plataformas, tornando o processo de escalabilidade mais simples. O armazenamento de ficheiros, as bases de dados, os serviços de IA ou os sistemas de gestão de informações e eventos de segurança (SIEM) conseguem ficar agregados em apenas um *back-end*, utilizando a arquitetura de CMS *headless* e, por consequência, causando mais independência entre os módulos [17] [18].

As primeiras ferramentas de edição em tempo real estão a surgir nos CMS. Esta técnica permite a integração de várias pessoas em simultâneo na mesma área de trabalho, contribuindo em conjunto na melhoria da qualidade do conteúdo e rapidez nos processos [18].

2.10. Exemplos de aplicações CMS

CMS é descrito com uma das tecnologias onde se pode observar uma maior rivalidade de produtos, quanto às suas funcionalidades e adaptações. Os produtos que conseguem estar aptos durante as evoluções tecnológicas são os que estão mais suscetíveis de ter sucesso no mercado.

Nem todas as plataformas CMS oferecidas possuem os mesmos princípios. Alguns focam-se na gestão mais genérica de *landing pages*, outras no comércio eletrónico. Para compreender o mercado dos CMS, pesquisas na Internet podem mostrar as tendências dos CMS ao longo dos últimos quinze anos. Utilizou-se, como referência, o Google Trends [19] [20] para acompanhamento, como se verifica na Figura 1. O Anexo A – Gráfico de interesse de soluções CMS ao longo dos últimos vinte anos está disponível para observar os dados com mais detalhe.

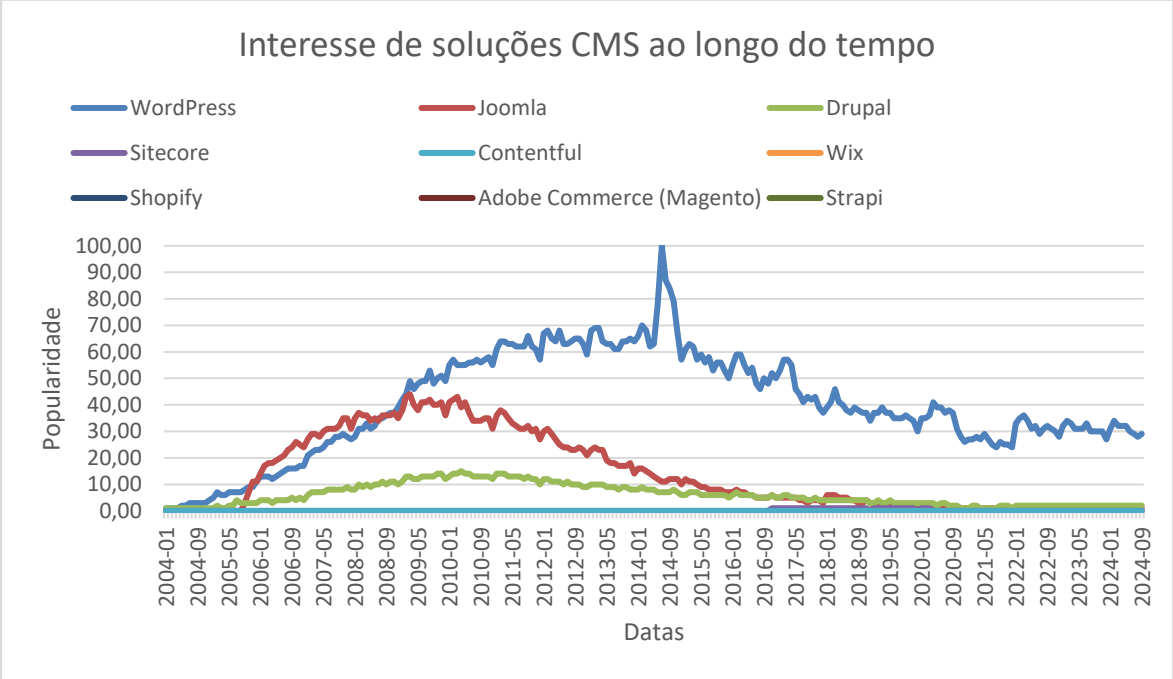


Figura 1 - Gráfico dos interesses de CMS ao longo dos últimos vinte anos.

O gráfico representa nove soluções que estão associadas a diversas arquiteturas. Foram escolhidas de acordo com a sua popularidade a nível pessoal e empresarial. Foram selecionadas Wordpress, Joomla, Wix, Adobe Commerce e Shopify para representar produtos com arquitetura tradicional, enquanto Drupal, Sitecore, Contentful e Strapi foram sorteados por serem soluções com arquitetura *headless*.

As soluções indicadas como tradicionais tendem a ser mais populares devido à sua disponibilidade para a comunidade, oferecendo ferramentas acessíveis e menos avançadas de criação de páginas. Os produtos que se mostram duráveis e escaláveis possuem menos popularidade (próxima de zero em alguns casos) comparativamente aos demais e geralmente são exclusivos para empresas.

Os seguintes subcapítulos têm uma análise mais precisa de alguns dos produtos representados, com o objetivo de destacar as funcionalidades únicas. Para além de estar relacionada com a sua popularidade no mercado, a escolha destes exemplos também permite descrever com mais detalhe as arquiteturas aplicadas.

2.10.1. Aplicações com arquitetura tradicional

Dois dos exemplos mais conhecidos de plataformas que utilizam as normas de uma arquitetura tradicional são o Wordpress e Joomla.

WordPress

O WordPress foi uma das primeiras plataformas comerciais a aplicar o conceito CMS, revolucionado a maneira como se gere um *website*. Desde o início do seu lançamento, no ano de 2003 [5], WordPress nunca largou as posições de topo [19], estando sempre na inovação, de acordo com as evoluções tecnológicas do momento. Ao longo destes vinte anos de existência, a comunidade tem desenvolvido ferramentas adicionais e temas que completem a plataforma.

No entanto, nem todos os negócios funcionam com este sistema, focando somente para pequenas e médias empresas ou para situações individuais. A sua escalabilidade é limitada para casos complexos, não sendo adequado para sítios Web que exigem complexidade [2]. Além disso, vários problemas de segurança foram detetados, colocando o conteúdo em risco de ser indevidamente acedido ou, numa situação mais grave, ser furtado.

Joomla

Como está descrito acima, os CMS não demonstram serem soluções para todos os sentidos. Uns focam-se nos casos gerais e em projetos simples, outros direcionam-se para áreas de negócio mais concretas. E o Joomla é distinguido como uma plataforma CMS com foco para comércio *online*.

Joomla apenas desenvolve páginas Web para *e-commerce*, funcionado da mesma forma que o WordPress. Uma das funcionalidades que se destaca é a utilização de diferentes idiomas dentro do *website*, permitindo-lhes uma abertura para mercados internacionais. Também possui uma biblioteca de *plugins* [7], mas a quantidade de *plugins* na comunidade não se aproxima do WordPress [2].

No Joomla, o que se pode salientar negativamente é o facto que não existem uma quantidade considerável de *plugins*, obrigando os programadores a criarem os seus quando não estão disponíveis, e a complexidade quanto à aprendizagem, fazendo com que a curva seja semelhante à da Drupal [2].

2.10.2. Aplicações com arquitetura *decoupled*

Como exemplo de plataformas que utilizam as normas de uma arquitetura *decoupled*, pode-se considerar o Drupal e o Brightspot.

Drupal

Quando se trata de implementar CMS para ambientes complexos, Drupal surge como uma solução tecnologicamente robusta e escalável [2]. Para além da escalabilidade e robustez, os seus pontos fortes incluem a segurança, fazendo-se diferenciar de outros produtos com a mesma arquitetura. Segundo a sua página Web [6], a equipa de desenvolvimento deste CMS pretende que a flexibilidade esteja sempre presente durante a gestão de plataformas empresariais, interligadas ao Drupal. Por isso, o leque deste CMS estende-se apenas para empresas de estatuto médio ou para grandes instituições.

Estando preparado para situações concretas ou de um nível de dificuldade acrescido, a sua montagem não aparenta ser simples. Para conseguir gerir todo o sistema do Drupal, é imprescindível realizar formações técnicas, o que exige ter experiência técnica. Estes pormenores apresentam que a curva de aprendizagem é acentuada [2].

Brightspot

O Brightspot é mais um exemplo de CMS *decoupled* que atende às necessidades de organizações com gestão de conteúdo complexa. Ele funciona bem tanto como CMS tradicional quanto *headless* graças à sua arquitetura adaptável, que o permite integrar facilmente vários serviços externos e API. A sua interface amigável e a facilidade de personalização destacam-no, mas mantém a robustez e a segurança [21].

O Brightspot difere-se de outras plataformas mais complexas, como o Drupal, por oferecer uma experiência de gestão simplificada e uma curva de aprendizagem mais curta. Isso torna-o uma opção viável para empresas que desejam crescer rapidamente sem grandes necessidades de conhecimento técnico.

2.10.3. Aplicações com arquitetura *headless*

No que refere a plataformas que utilizam as normas de uma arquitetura *headless*, tem-se como exemplo o Contentful e Prismic.

Contentful

Contentful é um CMS simples e moderno que se destaca por ser flexível e integrável com várias plataformas e serviços por meio de API. Por utilizar uma arquitetura *headless*, está projetado para facilitar a criação, gestão e distribuição de conteúdo para vários canais, como *websites*, aplicações móveis e dispositivos Internet of Things (IoT) [22].

Contentful concentra-se na agilidade e elimina os desafios da gestão de conteúdo tradicional com uma interface fácil de usar e uma API poderosa. Ele é ideal para empresas que precisam de uma solução escalável sem ficar restritas a uma estrutura CMS rígida, graças à sua flexibilidade. Embora seja bastante fácil de usar, especialmente para programadores, a implementação completa pode exigir conhecimento técnico especializado para configurar integrações e fluxos de trabalho personalizados [22].

Prismic

O Prismic é outro CMS *headless* que se destaca pela facilidade e simplicidade de uso. É uma solução perfeita para empresas que procuram uma implementação rápida e uma gestão de conteúdo simples. É flexível para funcionar em qualquer dispositivo ou tecnologia, permitindo que os utilizadores controlem conteúdo independentemente da apresentação [23].

O sistema de modelos de conteúdo da plataforma e a integração rápida com *frameworks* populares, como Next.js, Nuxt ou SvelteKit, são comuns entre os programadores que querem agilidade na entrega de conteúdo. A interface simples para editores não requer habilitações técnicas, o que a torna popular entre equipas de *marketing* e comunicação [23].

Apesar de ser simples, Prismic oferece funcionalidades importantes para grandes empresas, como suporte para várias linguagens e conteúdo com versões. Isso facilita a gestão de *websites* ou campanhas digitais complexas em todo o mundo.

2.10.4. Aplicações com arquitetura híbrida

Dois exemplos de plataformas que utilizam as normas de uma arquitetura híbrida são o Zesty.io e o Sitecore.

Zesty.io

Zesty.io é um CMS *hybrid headless* que se destaca por ser fácil de usar e oferecer uma experiência integrada para programadores e editores de conteúdo. Ele fornece uma plataforma adaptável que combina a flexibilidade de um CMS *headless*, permitindo a entrega de conteúdo por meio de API para vários canais, além da simplicidade de um CMS tradicional, que as equipas de *marketing* e conteúdo podem usar sem grandes dificuldades [24].

Zesty.io facilita a criação de *websites*, aplicações móveis e outros pontos de contacto digitais com as suas funcionalidades nativas para otimização de mecanismo de pesquisa, análise em tempo real e um fluxo de trabalho otimizado. *Designers* podem aproveitar o recurso *headless* para integrar API e personalizar profundamente o sistema, tornando-o ideal para empresas de médio porte que querem flexibilidade e eficiência. Isso pode ser feito sem precisar de conhecimento técnico profundo para as tarefas do dia a dia [24].

Sitecore

Grandes empresas que precisam de uma plataforma de gestão de conteúdo escalável e personalizada podem usar o Sitecore, um CMS híbrido confiável. Ao combinar a estrutura tradicional de CMS com a versatilidade sem esforço, permite a entrega de conteúdo para vários canais através de API, fornecendo uma solução integrada para *marketing* digital. Empresas podem personalizar experiências de utilizador em várias plataformas, como *websites*, aplicações móveis e outras interfaces digitais [25].

Sitecore permite a automatização de *marketing* e personalização em grande escala, pois pode unir dados de clientes e comportamento. No entanto, é mais adequado para grandes corporações que procuram uma solução de gestão de conteúdo poderosa, com *marketing* integrado e *headless*, que requer um alto nível de conhecimento técnico e um grande investimento [25].

2.11. Desafios e Futuro

De facto, os CMS são sistemas complexos e sempre dispostos a procurar soluções que adaptam à atualidade e a pensar no futuro. Pretende-se criar otimizações que possam minimizar os problemas causados nas comunicações ou na falta de acessibilidade para todas as pessoas.

Os principais desafios que enfrentam o crescimento são a escalabilidade, a flexibilidade e a curva de aprendizagem na utilização dos CMS [26]. A utilização de múltiplas API podem influenciar a utilização da arquitetura de CMS *headless*, enquanto a escalabilidade torna a curva de aprendizagem mais acentuada, atingindo níveis de complexidade avançados e, mais tarde, exigindo formações contínuas.

As tendências atuais conseguem mostrar o futuro dos sistemas de gestão de conteúdo. A inteligência artificial generativa marcará, certamente, o funcionamento destas plataformas. No entanto, também existem outros indícios que poderão estar inseridos nos CMS num futuro próximo.

IoT pode ter um papel importante para a gestão de conteúdo, relativamente aos utilizadores das ferramentas. Com a capacidade em obter grandes quantidades de informações e enviá-los, os CMS conseguem ter noções dos comportamentos das pessoas e, com base nisso, oferecer conteúdo mais personalizado e adequado [18].

Outra tecnologia, que está a emergir em conjunto com os dispositivos inteligentes IoT são as interfaces por voz [18]. Em combinação com a IA, os CMS podem estar equipados com assistentes de voz capazes de realizar tarefas pedidas pelos utilizadores, aplicando mais acessibilidade nas plataformas e manipulando conteúdo sem estar próximo de computadores ou dispositivos móveis.

3. A empresa e EasyWeb

Este projeto surgiu por parte da empresa LOBA, da qual eu sou colaborador, onde é necessário aplicar melhorias no EasyWeb, uma plataforma CMS proprietária da mesma. O projeto foi iniciado pela equipa da empresa Incentea Marketing e Inovação (MI) e, mais tarde, foi transferido para a empresa LOBA.

O EasyWeb tem uma história, no mínimo, interessante. A plataforma teve vários proprietários desde a sua origem, começando pela Elaconta, que a criou em 2005. Posteriormente, a Elaconta foi adquirida pelo grupo inCentea em 2012. Dentro da inCentea, o EasyWeb foi primeiramente gerido pela empresa Sente e, mais tarde, transitou para a inCentea Marketing e Inovação.

A inCentea Marketing e Inovação foi adquirida pela LOBA em setembro de 2023 e integrada na estrutura desta última, mantendo as suas atividades e continuando a desenvolver os projetos que tinha em curso. A LOBA, sediada em Oliveira de Azeméis, expandiu assim a sua presença, abrindo um novo espaço em Leiria após a aquisição. Neste momento conta com quase 200 colaboradores distribuídos por 7 escritórios em Portugal e 1 escritório em Londres.

3.1. O EasyWeb

O EasyWeb é um CMS que se tornou central nos projetos da inCentea Marketing e Inovação, especialmente no início do desenvolvimento das melhorias. A sua missão sempre foi oferecer melhores condições de gestão de *websites*, com funcionalidades de integração com plataformas Enterprise Resource Planning (ERP). De facto, a integração com *software* de gestão de empresas é o ponto forte da plataforma, o que tem contribuído para o seu sucesso.

Segundo a entrevista realizada com o criador do EasyWeb, detalhada no Anexo B - Entrevista sobre a plataforma EasyWeb, a plataforma começou como um conjunto de *scripts* com várias funcionalidades. Com o crescimento, surgiu a necessidade de criar algo mais estruturado, com melhor personalização entre projetos, mais robustez, estabilidade, segurança e também maior facilidade na implementação em alojamentos predefinidos, originando um portal de gestão de dados.

O CMS em questão é um portal para *back-end*, associado a uma página Web, desenvolvido em PHP, utilizando a *framework* Laravel. Tanto o *front-end* como o *back-end* estão agrupados num único projeto, com integrações locais entre ambos, tratando-se, desta forma, de um CMS tradicional, como se pode verificar na Figura 2.

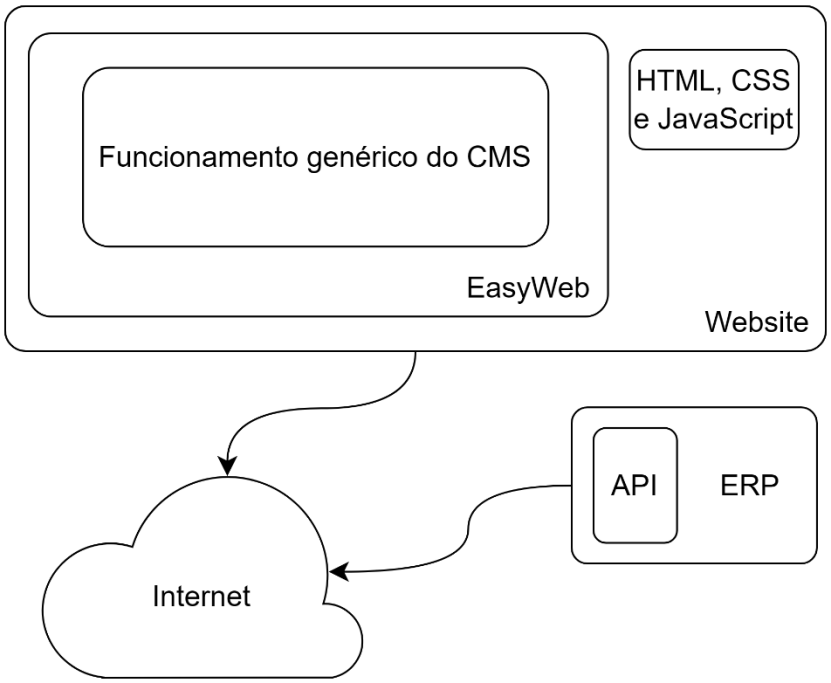


Figura 2 - A arquitetura do EasyWeb, considerado como um CMS tradicional e que interage com API de ERP.

A integração com ERP é feita através do uso de API fornecidas, permitindo gerir processos como a criação de encomendas ou a apresentação de dados em áreas restritas.

4. Análise de requisitos

Para que se consiga ter uma ideia mais clara sobre o que se pretende realizar neste projeto, é necessário compreender o que a comunidade consegue oferecer, porque, em alguns casos, não é necessário reinventar a roda. O objetivo principal, nesta fase, é realizar pesquisas sobre as ferramentas existentes no mercado, que sejam, na esmagadora maioria, *open-source*, que possam ser facilmente integráveis na plataforma e que oferecem facilidades na adaptação e modificação. Pretende-se, desta forma, implementar o sistema de construção de páginas através de arrasto e procurar ideias e/ou sugestões interessantes que possam ser úteis para o EasyWeb, podendo ser implementadas em futuros momentos.

No entanto, nem todas as funcionalidades para melhoria serão implementadas diretamente com base em projetos existentes, devido a certas limitações, principalmente na personalização. Portanto, é necessário realizar uma análise ao que se pretende realizar e verificar se é possível realizar a integração com o EasyWeb. Além disso, observar a composição de outros projetos oferece uma visão mais clara sobre o que se pretende realizar, podendo ser uma fonte de inspiração. Para este caso, as funcionalidades que irão ser analisadas são o sistema de carregamento de ficheiros para formulários e o sistema de gestão de tabelas.

Este capítulo está estruturado em três subcapítulos, começando com a apresentação dos requisitos funcionais. De seguida, os requisitos não funcionais são mostrados e, por fim, estão as restrições impostas.

4.1. Requisitos funcionais

Os requisitos funcionais são os que representam funcionalidades específicas que os componentes devem fazer.

No componente de carregamento de ficheiros:

- Suporte para diferentes navegadores Web;
- Integração com HTML simples;
- Utilização de API para HTML (*Drag and Drop* [27]);
- Validação dos ficheiros inseridos, através do arrasto e da forma convencional;
- Possibilidade de adicionar múltiplos ficheiros sem substituir os existentes (manter uma lista de ficheiros);

- Excluir ficheiros da lista;
- Restrições em casos de carregamento de um ou vários ficheiros.

No módulo de gestão de ficheiros de servidores FTP:

- Suporte para diferentes navegadores Web;
- Apresentação de ficheiros por pastas, se existirem;
- Visualização do conteúdo dos ficheiros;
- Filtros por ficheiros;
- Existência de ações, como transferir, copiar, mover ou eliminar;
- Possibilidade em carregar ficheiros, podendo utilizar o componente de carregamento de ficheiros.

No componente de gestão de tabelas:

- Suporte para diferentes navegadores Web;
- Integração com a estrutura das tabelas do HTML;
- Limitação da circulação da linha na tabela (não ultrapassar os limites do corpo da tabela);
- Limitação da circulação apenas na vertical;
- Atualização imediata na base de dados e na página Web após o arrasto.

No componente de construção de páginas Web:

- Compatibilidade com os componentes existentes no EasyWeb;
- Implementação de novos componentes;
- Compatibilidade com a base de dados (assumindo que o conteúdo da página seja guardado numa tabela);
- Pré-visualização da página Web em diferentes dimensões da janela.

4.2.Requisitos não funcionais

Os requisitos não funcionais são os que representam a comportamentos e qualidades dos componentes, com foco em temas como responsividade, personalização, tratamento de erros, entre outros. A segurança, o desempenho e a usabilidade são tópicos mais revelantes nos requisitos não funcionais.

No componente de carregamento de ficheiros:

- Interface visual personalizável para ambientes específicos;
- Representação imediata do *feedback*, através de mensagens ou notificações;
- Adaptações quando não se consegue realizar arrasto (por exemplo, em dispositivos móveis);
- Responsividade;
- Tratamento de erros e exceções, mostrando *feedback*.

No módulo de gestão de ficheiros de servidores FTP:

- Interface visual personalizável para ambientes específicos;
- Representação imediata do *feedback*, através de mensagens ou notificações;
- Responsividade;
- Tratamento de erros e exceções, mostrando *feedback*.

No componente de gestão de tabelas:

- Interface visual personalizável para ambientes específicos;
- Representação imediata do *feedback*, com mensagens ou notificações;
- Responsividade;
- Tratamento de erros e exceções, mostrando *feedback*.

No componente de construção de páginas Web:

- Facilidade na modificação e personalização dos formulários de edição de estilos, entre outros;
- Compatibilidade com bibliotecas populares de CSS e de JavaScript;
- Facilidade de uso e documentação consistente.

4.3.Restrições

Em todos os projetos, existem sempre restrições que causam limitações no processo de desenvolvimento. Neste, em concreto, as restrições destinam-se aos novos componentes a desenvolver. São um fator imprescindível para tornar viável o projeto, porque pode limitar a integração com o EasyWeb. Claramente que existe prioridade no seguimento destes requisitos, porque conseguem mudar estratégias durante o desenvolvimento. Pretende-se que as

novas funcionalidades sejam entregues com qualidade, sem atrasos causados por situações inesperadas.

As restrições colocadas para este projeto são as seguintes:

- Os componentes devem ser capazes de se integrarem com projetos PHP que seguem os ideais de *framework* Laravel, devido à base do projeto do CMS EasyWeb;
- Cada componente desenvolvido em JavaScript deve evitar o uso de componentes externos para minimizar a dependência;
- Os componentes devem totalmente ser desenvolvidos pela equipa, sem recurso a bibliotecas externas, com exceção do componente de construção de páginas;
- Os componentes, na programação em JavaScript, não podem utilizar *frameworks* disponíveis (React, Vue, entre outros);
- Relativamente a estilos, deve-se evitar o uso de *frameworks* CSS, com a exceção de Bootstrap, que está incluída no EasyWeb.

Por consequência, as restrições geram impactos que podem dificultar o processo de desenvolvimento, como acontece neste projeto. São os seguintes:

- O uso das mesmas linguagens de programação torna a programação mais consistente com o sistema, oferecendo compatibilidade e manutenção fácil. Contrariamente, mais limitações existem na utilização de pacotes que não sejam nativamente compatíveis com Laravel;
- O bloqueio de bibliotecas externas torna o desenvolvimento mais exigente, porque terá de ser implementado funcionalidades internamente, em vez de utilizar algo pronto. Mesmo que diminua a dependência de bibliotecas de terceiros, limita a agilidade no desenvolvimento;
- O desenvolvimento total dos componentes mostra que existe total controlo e permite mais personalização, de acordo com os requisitos pedidos. Contudo, implica mais tempo para programar;
- A utilização de JavaScript sem *frameworks* obriga a utilizar o JavaScript puro para construir as funcionalidades necessárias, oferecendo mais controlo no componente. Mas, como não existe simplificação existente nas *frameworks*, mais

complexo se torna a implementação dos componentes. O mesmo acontece com a utilização de CSS puro.

Geralmente, as restrições impostas gerem diferentes desafios, como o aumento no tempo de desenvolvimento, o aumento de complexidade e o cuidado nas atualizações e manutenções. A manutenção é necessária porque, como não se utiliza funcionalidades externas, que contêm atualizações, os programadores são obrigados a fazer essas tarefas manualmente, podendo ter de reescrever o código para adaptações.

5. Processo de Desenvolvimento

Construir uma página ou uma aplicação Web envolve uma variedade de processos e competências especializadas, por isso não é uma tarefa fácil. A maioria desses projetos exige uma interface visual, que frequentemente requer um grupo de *designers*. Como resultado, uma grande quantidade de pessoas que trabalham em grupos ou equipas acaba por participar num único projeto, o que torna mais difícil organizar e gerir o trabalho.

A implementação de métodos ágeis de desenvolvimento de *software*, como o Scrumban, em conjunto com a gestão eficaz de projetos, [28] é essencial para melhorar a organização e a gestão de projetos. O Scrumban, que combina elementos do Kanban e do Scrum, permite flexibilidade e foco no fluxo de trabalho e garante que as tarefas sejam concluídas dentro do prazo e com a qualidade esperada [28]. A popularidade deste método cresce gradualmente a nível mundial e, por isso, a empresa LOBA, nos últimos tempos, tem vindo a aplicar o Scrumban nos projetos mais recentes.

Através da aplicação do Scrumban no projeto, conseguiu-se não apenas melhorar a organização das tarefas, mas também ajustar rapidamente o fluxo de trabalho às alterações de prioridades e requisitos. Esta abordagem demonstrou ser particularmente eficiente na preservação da produtividade e na entrega constante de valor, garantindo a adaptabilidade necessária para lidar com os obstáculos encontrados durante o processo de desenvolvimento. A opção pelo Scrumban possibilitou a manutenção de um equilíbrio entre planeamento estruturado e capacidade de adaptação. Mais detalhes da utilização do podem ser vistos na secção 5.2 (Metodologia no projeto).

5.1. Metodologia Scrumban no desenvolvimento de aplicações Web

Ao iniciar o desenvolvimento de uma aplicação Web, muitas coisas devem ser consideradas, como arquitetura, segurança, UI e UX, entre outras coisas. A aplicação depende da arquitetura, que é fundamental. Como aplicações Web são frequentemente alvos de ciberataques, a segurança deve ser selecionada como prioritária. UI e UX devem ser bem planeadas, porque são essenciais para a interação dos utilizadores.

Para lidar com essas questões de maneira eficiente, a equipa deve ser multidisciplinar, garantindo que cada setor receba a atenção e o conhecimento necessários. O processo começa definindo a arquitetura do projeto e as linguagens de programação. Essas definições geralmente estão estabelecidas nas empresas.

Uma reunião de *brainstorming* inicial ajuda na definição da arquitetura do projeto. Em seguida, uma reunião de lançamento é realizada para definir os requisitos do cliente. A equipa de design pode usar essas informações para criar esboços e protótipos funcionais da aplicação, dando ao cliente uma visão clara de como será a plataforma.

O sistema Kanban do Scrumban organiza as tarefas em colunas, como "Para Fazer", "Em Progresso" e "Concluído". Cada equipa, incluindo as de *back-end* e *front-end*, tem uma visão clara de suas funções e deveres, o que permite que o trabalho flua sem problemas [28].

A equipa de *back-end* pode criar algoritmos de gestão de dados e comunicação entre plataformas enquanto a equipa de *design* trabalha nos protótipos, sem depender do design final. A organização, a gestão de dados e a compreensão do código são facilitadas pelas ferramentas de gestão de servidores e de gestão de versões de código integradas ao quadro Scrumban.

É de a responsabilidade das equipas especializadas manter o *backoffice* de gestão de conteúdo da plataforma. Quando o *backoffice* e o *design* do *website* estiverem a funcionar, a equipa de *front-end* pode começar a trabalhar na interface gráfica da aplicação. Essa será a parte da aplicação com a qual os usuários interagirão.

Além disso, o Scrumban permite que a equipa de testes comece a testar a aplicação assim que uma funcionalidade é concluída, permitindo que erros sejam encontrados e corrigido rapidamente. Isso evita atrasos e mantém a qualidade do projeto [28].

Finalmente, com a ajuda das práticas de Scrumban, a equipa de gestão do projeto monitora o andamento do projeto e ajusta as prioridades para garantir que o trabalho seja concluído no prazo e sem comprometer a qualidade.

5.2.Metodologia no projeto

A metodologia em questão está estruturada para projetos grandes e, para aplicar no projeto das melhorias de EasyWeb, foram realizadas diversas adaptações.

Os primeiros passos estiveram associados na seleção das melhorias a implementar, através de uma reunião. Nessa fase, foram escolhidas as que eram consideradas como necessárias, de acordo com as opiniões dos clientes.

A partir da reunião inicial, foram definidas as tarefas relacionadas com o desenvolvimento. A LOBA possui vários gestores de projetos que organizam as tarefas utilizando a metodologia Kanban, separando-as em colunas. Na Figura 3, consegue-se observar que foram criadas diversas tarefas para cada componente, organizadas por etapas de desenvolvimento. As tarefas que estão na fase "Aguarda documentação" correspondem a melhorias que não estão no âmbito deste projeto. A aplicação que faz a gestão pertence a um pacote de aplicações empresariais Zoho, sendo esta denominada como Zoho Projects.

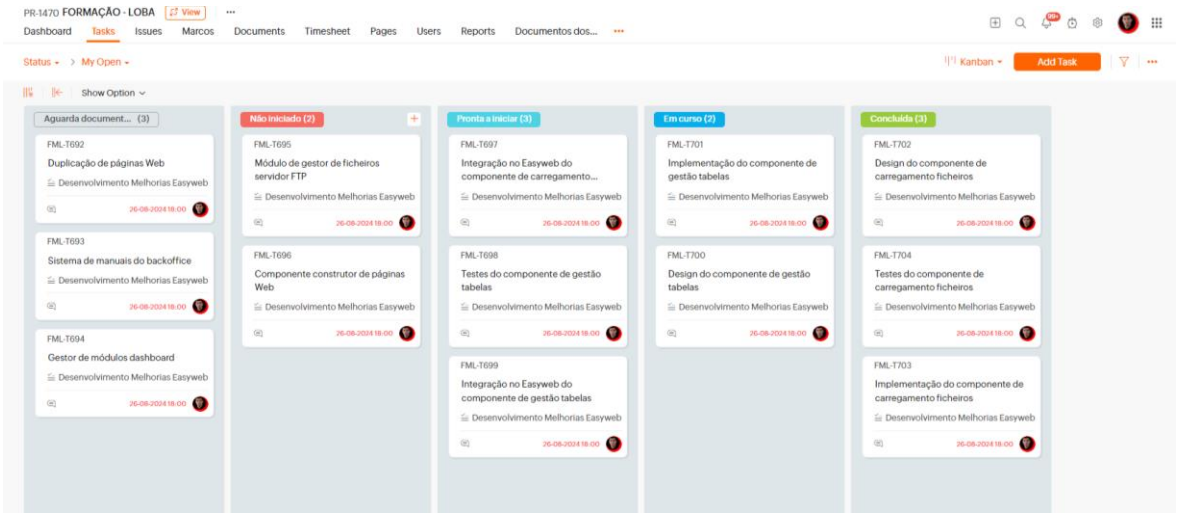


Figura 3 - Organização de tarefas relacionadas com as melhorias no EasyWeb, em janeiro de 2024.

Os detalhes das tarefas associadas ao desenvolvimento podem ser analisados na Figura 4, onde há a indicação do estado, do tempo estipulado e também das pessoas envolvidas.

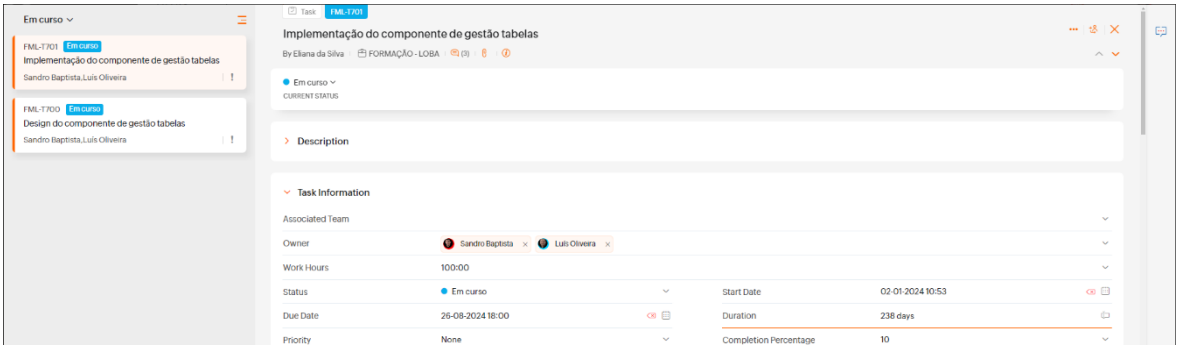


Figura 4 - Detalhes de uma tarefa relacionada com as melhorias no EasyWeb.

Durante a realização das tarefas atribuídas, várias reuniões mensais foram realizadas, com o objetivo de partilhar o estado da implementação, de mostrar alguns esclarecimentos quanto a pormenores técnicos e também de preparar para implementações em CMS de testes e em projetos reais, posteriormente.

6. Arquitetura e Tecnologias

O EasyWeb é uma plataforma CMS tradicional, amplamente utilizada para o gestão e publicação de conteúdo *online*. O EasyWeb desempenha um papel crucial ao permitir a integração com sistemas de gestão empresarial, como os ERP, oferecendo uma solução versátil para empresas.

No entanto, com a evolução contínua das tecnologias empresariais, nomeadamente na gestão, a plataforma necessita de melhorias frequentes, implementando novas ferramentas desenvolvidas ou integrando com algumas existentes.

Neste capítulo, estão apresentados dois subcapítulos, que separam a arquitetura do EasyWeb das tecnologias associadas aos requisitos analisados no capítulo 4. No subcapítulo das tecnologias, vários produtos existentes são apresentados para, mais tarde, serem comparados e retirar conclusões antes de iniciar o processo de implementação.

6.1. Arquitetura

A plataforma EasyWeb está integrada diretamente com o *front-end* do projeto, sem a necessidade de uma API para a transmissão do conteúdo HTML gerado. Os ficheiros e dados são armazenados numa plataforma de gestão de servidores, que também hospeda outros serviços essenciais, como bases de dados, servidores de e-mail e FTP, conforme ilustrado na Figura 5.

O EasyWeb foi projetado para ser compatível com a integração de sistemas API fornecidos pelas plataformas de ERP. Isso permite que o sistema se comunique de maneira eficiente com outros sistemas empresariais, facilitando a gestão de processos internos.

O conteúdo gerido no portal EasyWeb resulta na criação de páginas Web genéricas e responsivas, adaptadas para diversos dispositivos, como computadores, *tablets* e *smartphones*. Em projetos específicos, o EasyWeb permite a criação de áreas reservadas para o controlo de informações dos ERP, podendo a *interface* ser otimizada para um tipo específico de dispositivo.

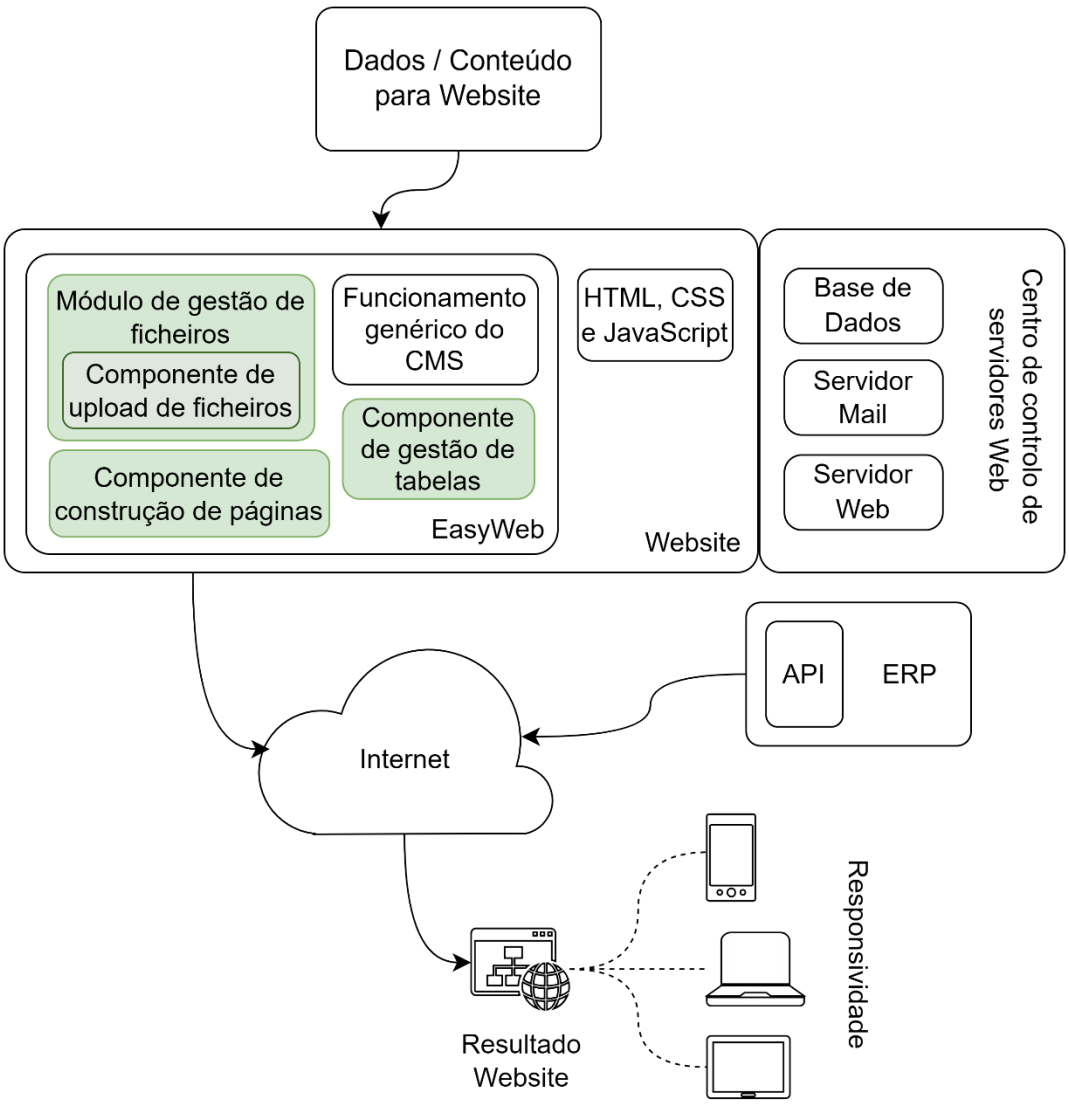


Figura 5 - Arquitetura do EasyWeb com os novos módulos.

Observando a figura, as regiões coloridas a verde representam as melhorias a implementar neste projeto. Estas melhorias pertencem à parte interna do CMS, podendo, em algum momento, interagir com outros servidores que o centro de controlo disponibiliza.

6.2.Tecnologias concorrentes

Dado que o capítulo 4 (Análise de Requisitos) aborda os critérios rigorosos para o desenvolvimento integral dos produtos, é importante compreender como esses componentes funcionam na prática. É importante lembrar que as tecnologias selecionadas, com exceção de editores de páginas Web, servem como exemplos capazes de fornecer funcionalidades e

métodos de implementação. No caso de editores, o objetivo é utilizar uma dessas tecnologias, devido à sua complexidade de implementação de raiz. Nos seguintes tópicos, são destacadas as áreas específicas em que esses componentes podem ser aplicados.

Carregamento de Ficheiros

Para o componente de *upload* de ficheiros, foram identificadas várias bibliotecas que podem ajudar na criação de componentes para o EasyWeb, cada uma mostrando funcionalidades distintas para melhorar a experiência do utilizador e a eficiência do sistema:

- **Dropzone.js**: Uma biblioteca popular que facilita o *upload* de ficheiros com suporte a arrastar e largar;
- **FilePond**: Uma solução flexível e extensível para *upload* de ficheiros, com uma interface moderna;
- **Uppload.js**: Um *uploader* de ficheiros com suporte para múltiplos fornecedores de armazenamento;
- **React-Dropzone**: Uma biblioteca baseada em React para implementar funcionalidades de arrastar e largar ficheiros.

Gestão de ficheiros de servidores FTP

Como o módulo corresponde a uma secção de visualização de ficheiros existentes em servidores FTP, foram identificadas várias ferramentas que podem ser úteis na implementação:

- **Webix File Manager**: uma biblioteca simples e prática para gestão de pequenos sistemas de ficheiros;
- **DevExtreme FileManager**: um explorador de ficheiros com funcionalidades avançadas e preparado para diferentes *frameworks*;
- **Syncfusion File Manager**: uma solução completa e funcional para gerir ficheiros.

Gestão de Tabelas (Foco no arrasto de linhas)

Na gestão de tabelas, especialmente na funcionalidade de arrastar e largar linhas, foram identificadas as seguintes bibliotecas que podem fornecer bases para o desenvolvimento:

- **table-dragger**: Um *plugin* leve que permite a reorganização de tabelas através de arrastar e largar;

- **SortableJS**: Uma poderosa biblioteca para arrastar e reorganizar itens numa interface, incluindo tabelas;
- **Material React Table**: Uma tabela baseada em React que oferece funcionalidades avançadas, como arrasto de linhas, com uma estética baseada no Material Design.

Construção de Páginas Dinâmicas

Para a construção de páginas dinâmicas, que é uma das áreas mais críticas para o EasyWeb, foram considerados os seguintes produtos:

- **Laravel GrapesJS**: Um construtor de páginas visual, desenvolvido em JavaScript e montado em Laravel, que oferece uma interface intuitiva para criação de páginas;
- **Laravel PageBuilder**: Um construtor de páginas com foco em flexibilidade e integração fácil com Laravel;
- **VvvebJS**: Uma ferramenta de edição de páginas visual que permite a criação rápida de layouts dinâmicos.

Esses componentes foram escolhidos com base na sua capacidade de fornecer ideias interessantes para os novos componentes a desenvolver, atendendo aos objetivos do projeto em termos de funcionalidade e usabilidade.

6.2.1. Dropzone.js

Dropzone.js é uma biblioteca feita em JavaScript com a intenção de fornecer funcionalidades de *upload* de ficheiros através de *drag-and-drop*. O seu *design* é minimalista e é apreciado por diversos programadores que procuram por soluções fáceis de implementar e eficazes. Pode ser aplicado em qualquer elemento HTML visível, embora não sendo recomendado, convertendo num componente mais robusto, e capaz de interagir com servidores, através de AJAX. Além disso, tem ferramentas de personalização, tanto visualmente como na validação dos ficheiros.

A Figura 6 mostra a interface inicial que a biblioteca fornece, podendo ser personalizada ao editar, em JavaScript, as configurações do componente. Através da documentação existente [29], consegue-se observar as opções a utilizar na configuração [30], podendo ser úteis para organizar as configurações do componente de *upload* a desenvolver.

Figura 6 - Apresentação da interface personalizada da biblioteca, onde se vê mensagens de aviso quando certos ficheiros, proibidos na configuração, são inseridos.

6.2.2. FilePond

FilePond é uma biblioteca desenvolvida em JavaScript que permite gerir os ficheiros existentes no componente de *upload* para formulários. Este possui uma interface mais moderna, funcionando em todos os cenários que envolvem armazenamento de ficheiros no servidor. Desta forma, torna-se ideal para situações mais complexas de carregamento de ficheiros. Por isso, é considerado um produto flexível e completo.

A Figura 7 representa a interface criada por omissão. Nela, é possível observar múltiplos ficheiros a ser carregados para o servidor em simultâneo e com a indicação da percentagem. Estes pormenores são nativos, mas podem ser editados nas configurações do componente. A personalização é aceite e podem ser vistos o processo de implementação na sua documentação [31].

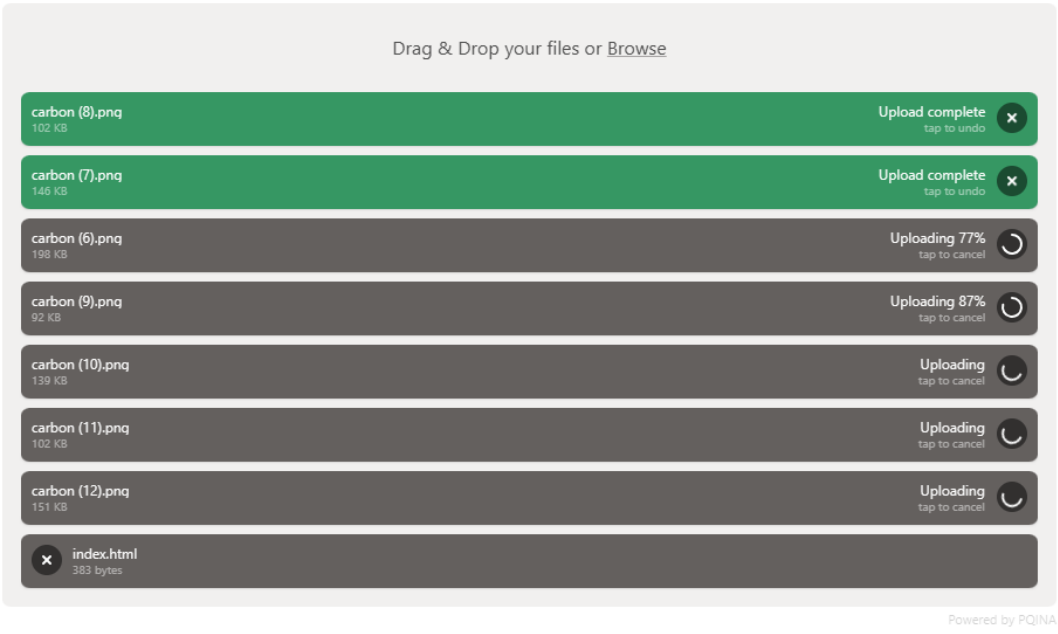


Figura 7 - Apresentação da interface inicial da biblioteca.

6.2.3. Uppload.js

Uppload.js é uma biblioteca feita em JavaScript que contém funcionalidades avançadas de *upload* de imagens. Mostra-se bastante flexível e personalizável, podendo integrar com muitos *plugins* desenvolvidos pela comunidade [32].

Para além de conter o recurso *drag-and-drop*, consegue pesquisar, importar e editar imagens de alguns serviços externos, como Pixabay [33] e Unsplash [34] (Figura 8), facilitando a gestão de ficheiros em diversos ambientes. A edição de imagens é realizada depois de escolher a imagem. Além disso, possui *plugins* de idiomas, permitindo adaptar o componente de acordo com o idioma da página Web.

Dentro da secção da edição de imagens, o *plugin* está disponível apenas quando uma única imagem é selecionada, tanto localmente como através de outros *plugins*. Como está disponível o carregamento de múltiplos ficheiros no serviço local, fica desativada a edição se forem adicionados em simultâneo.

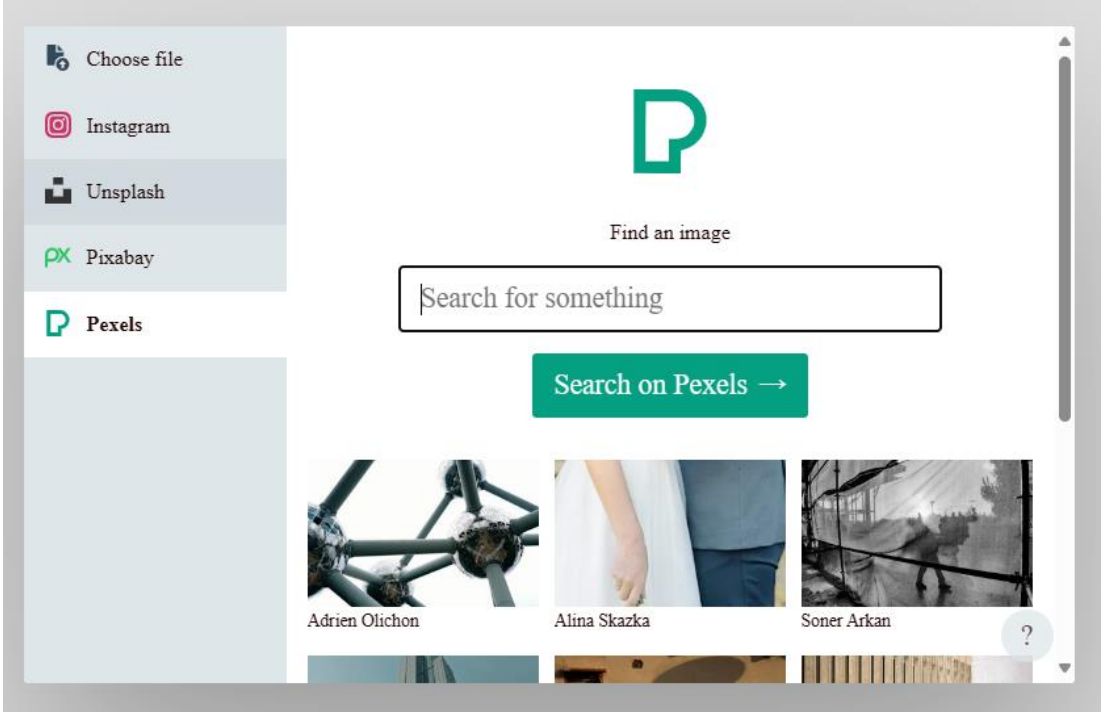


Figura 8 - Como o Uppload.js está preparado para imagens, alguns *plugins* foram criados, com foco principal na integração com ferramentas externas.

6.2.4. React-Dropzone

React-Dropzone é uma biblioteca desenvolvida em JavaScript, baseada em React. É um *hook* que cria uma zona de carregamento de ficheiros simples com *drag-and-drop* [35].

Como é considerado um pacote React, está preparado para interligar com outros pacotes da mesma *framework*, facilitando a personalização e expansão das funcionalidades de *upload*, como se pode observar na Figura 9. Um dos pacotes é Pintura [36], um componente de edição de imagens desenvolvido pela mesma equipa do FilePond (capítulo 6.2.2), também utilizado no componente Uppload.js (capítulo 6.2.3).

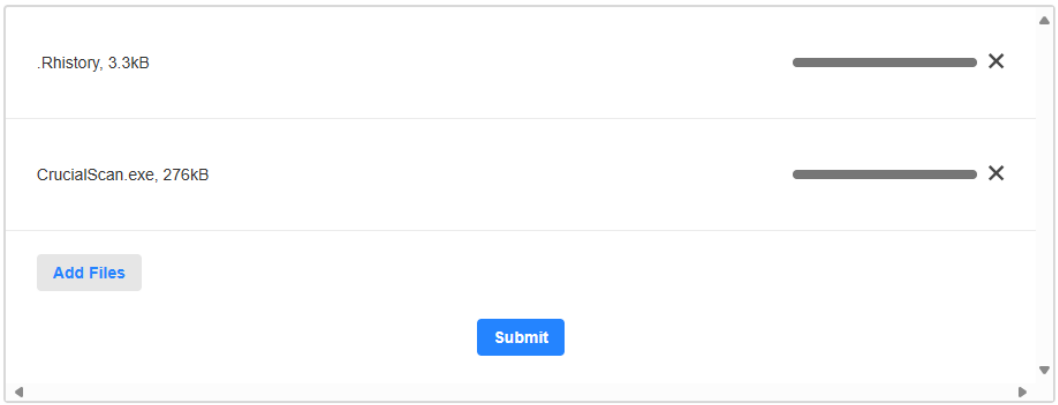


Figura 9 - Apresentação da interface personalizada da biblioteca [37].

6.2.5. Webix File Manager

Webix File Manager é uma ferramenta aplicada para páginas Web que realiza a gestão de ficheiros de uma pasta específica. Apresenta uma interface simples e compreensível, podendo ser personalizável com temas específicos. No entanto, apenas está disponível de acordo com a licença adquirida [38].

As operações nos ficheiros podem ser acedidas ao selecionar a pasta ou o ficheiro e permite a visualização da lista de diversas formas. O conteúdo e os detalhes dos ficheiros podem ser vistos numa secção lateral, que se encontra desativada por omissão [38].

A biblioteca também fornece funcionalidades de visualização do conteúdo de cada ficheiro, nomeadamente em ficheiros de texto e imagens. A edição de texto também está disponível durante a visualização. No caso de imagens, é possível pré-visualizar o conteúdo a partir da lista de ficheiros. Além disso, permite reproduzir ficheiros de áudio sem necessitar de transferir [38].

A adição de ficheiros ou de pastas pode ser feito através de arrasto diretamente para a lista, facilitando o processo de *upload*.

A Figura 10 contém uma representação da interface da biblioteca, onde também se consegue observar um conjunto de operações associado ao ficheiro selecionado.

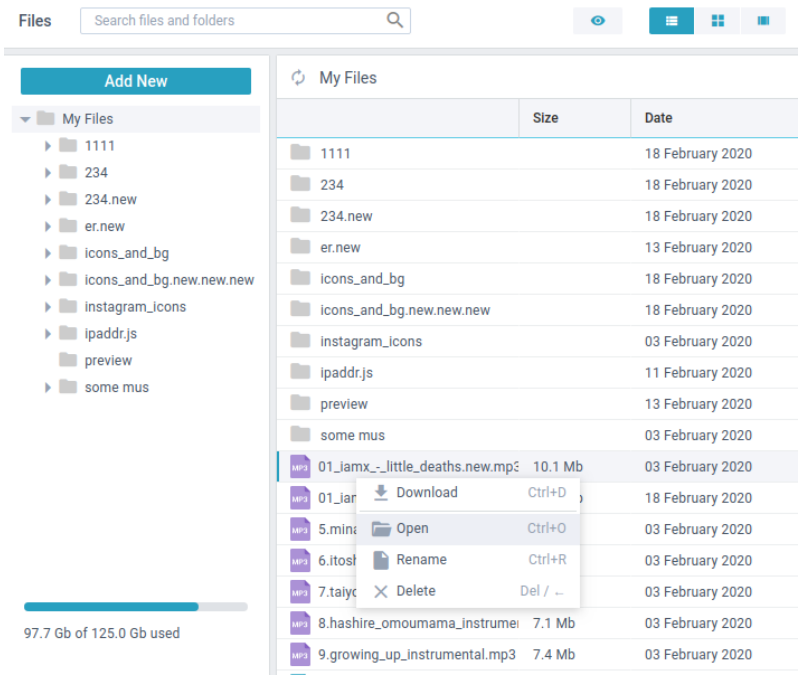


Figura 10 - Interface do explorador de ficheiros, com a representação de menu de operações rápidas [38].

6.2.6. DevExtreme FileManager

DevExtreme FileManager é um componente da DevExpress, conhecida por fornecer bibliotecas para desenvolvimento de interfaces de utilizador. A biblioteca é um produto pago, obrigando a ter uma licença DevExpress para adquirir este componente [39].

A solução contém funcionalidades mais avançadas relativamente ao anterior e possui uma interface moderna, com a utilização de *frameworks* de componentes e de temas mais recentes. Pode ser configurado para funcionar com sistemas de ficheiros virtuais, utilizando integrações com serviços remotos via API.

Observando o explorador de ficheiros (Figura 11), a organização é semelhante ao Webix File Manager, bem como o acesso ao menu de ações para ficheiros e pastas. Contudo, na questão de visualização do conteúdo dos ficheiros, apenas se consegue ver ao clicar no ficheiro, e os *thumbnails* das imagens não são carregados na lista dos ficheiros.

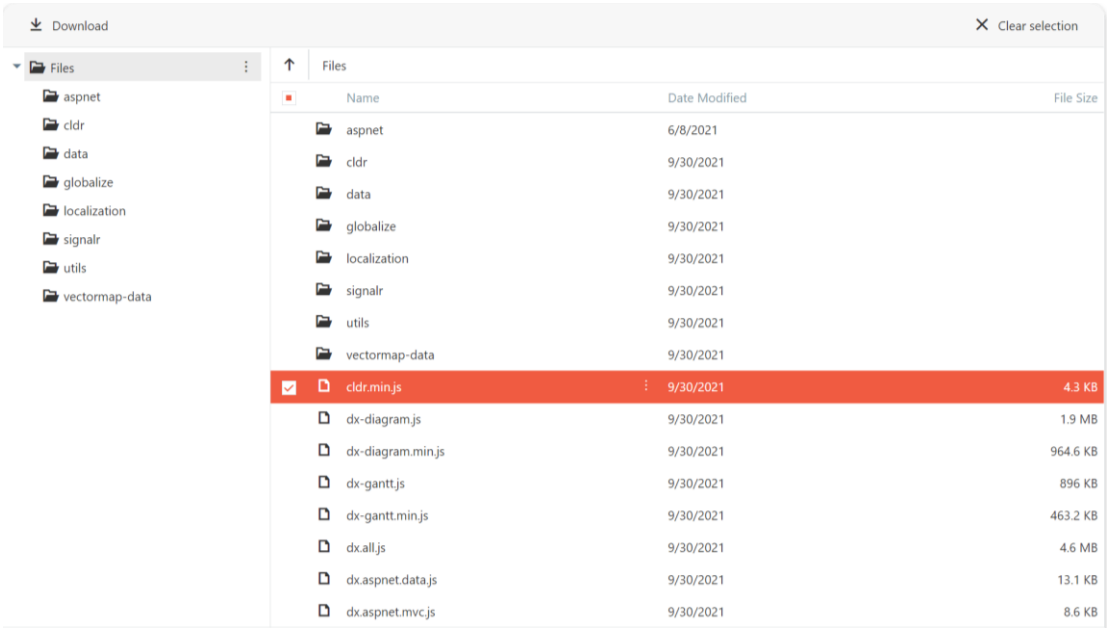


Figura 11 - Interface do explorador de ficheiros [39].

6.2.7. Syncfusion File Manager

Syncfusion File Manager faz parte de um pacote de componentes de Syncfusion, reconhecido por conter pacotes para desenvolvimento de aplicações. Também é considerado um produto licenciado, possuindo licenças comunitárias e comerciais [40].

Neste produto, destacam-se a interface familiar, baseado em exploradores de ficheiros tradicionais, a personalização, a existência de operações mais avançadas, a possibilidade em integrar com serviços externos e a compatibilidade em diversas *frameworks* JavaScript.

A Figura 12 apresenta o gestor de ficheiros, com uma interface semelhante ao produto DevExtreme FileManager. O menu de operações encontra-se no topo, mas pode ser acedido ao clicar no ficheiro ou pasta. Dentro de uma pasta, é possível arrastar ficheiros para a área da lista, realizando *upload* desses ficheiros.

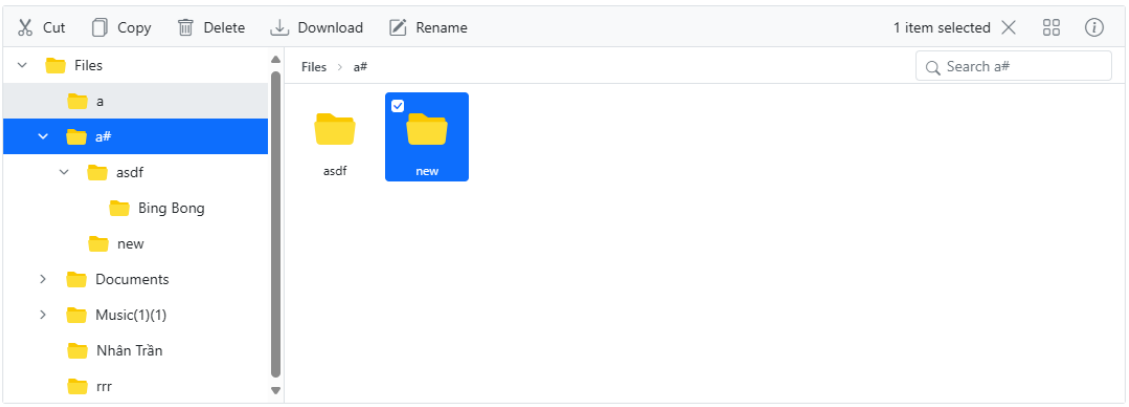


Figura 12 - Interface do explorador de ficheiros Syncfusion [40].

6.2.8. table-dragger

O table-dragger é um pacote JavaScript leve que contém todas as ferramentas necessárias para tornar uma tabela simples numa tabela ordenável através de arrasto. A implementação é simples e possui alto desempenho, o que torna esta solução indicada para situações onde a manipulação de tabelas é imprescindível [41].

De acordo com a sua documentação [41], por omissão, a implementação atribui as células da primeira linha como arrastáveis, movendo a coluna associada, mostrando na Figura 13 um exemplo. Além disso, com as devidas configurações, é possível definir arrastos de linhas.

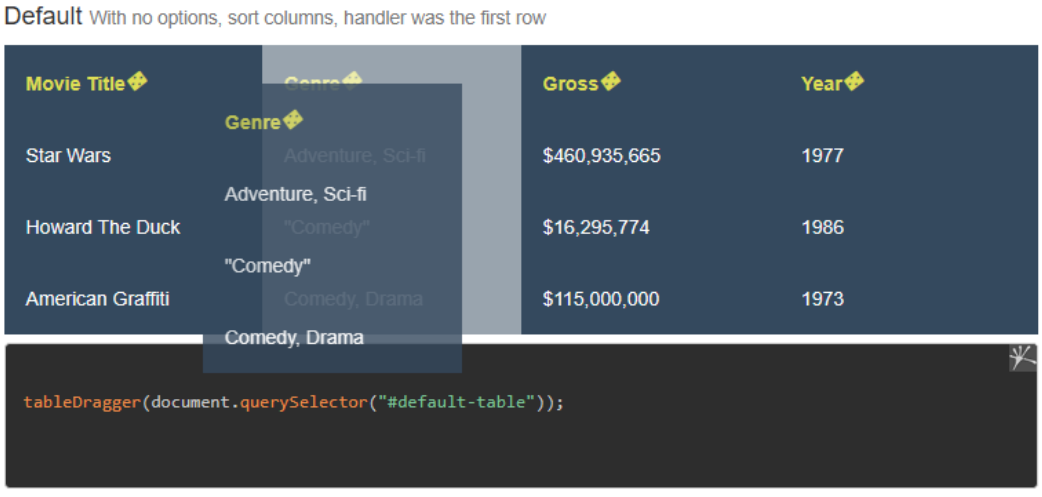


Figura 13 - Apresentação da interface inicial da biblioteca [41].

6.2.9. SortableJS

SortableJS é uma biblioteca de JavaScript com muitas funcionalidades para arrastar e reorganizar itens numa interface. A sua flexibilidade permite a aplicação numa variedade de contextos, incluindo tabelas, listas e grades. Com um *design* modular, SortableJS é facilmente personalizável e extensível [42].

A biblioteca destaca-se pela quantidade de funcionalidades relacionadas com o arrasto de linhas de uma lista. Além disso, possui alguns *plugins* interessantes, como o MultiDrag, cuja demonstração está representada na Figura 14.

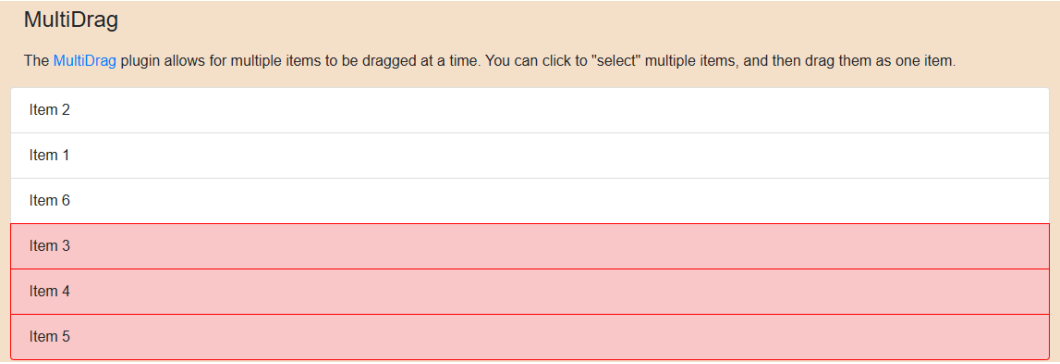


Figura 14 - SortableJS permite ordenar múltiplas linhas em simultâneo [42].

6.2.10. Material React Table

Material React Table oferece uma interface moderna baseada em Material Design, com funcionalidades avançadas como arrasto de linhas e personalização de colunas. É ideal para aplicações React que requerem uma solução de tabela robusta e agradável ao nível estético [43].

De acordo com a Figura 15, o arrasto encontra-se aplicado nas colunas, cuja referência é o ícone com duas barras na horizontal. Além disso, a coluna tem funcionalidades de ordenação, outras formas de filtros e também a gestão de visualização da coluna na tabela. As tabelas podem ser integradas com serviços externos de dados, disponibilizando a funcionalidade de *infinite scroll*, e também possuem botões de ação, geralmente associados a edição ou eliminação do registo.

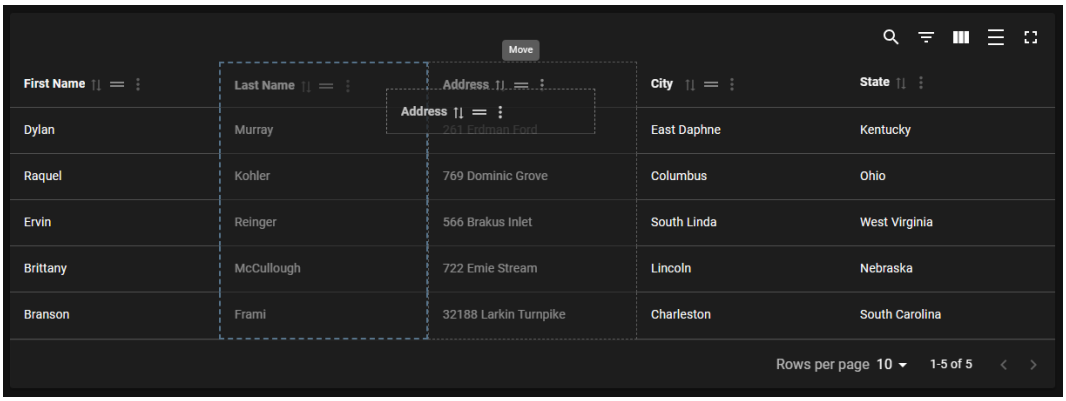


Figura 15 - A biblioteca possui muitas funcionalidades que permitem personalizar totalmente as tabelas [44].

6.2.11. Laravel GrapesJS

LaravelGrapesJs é um *wrapper* para integrar a biblioteca JavaScript GrapesJS com o Laravel. GrapesJS é uma das bibliotecas mais populares para a construção de páginas web através do arrasto de componentes, resolvendo limitações dos editores WYSIWYG tradicionais, especialmente na criação de estruturas HTML [45].

GrapesJS oferece uma interface intuitiva, onde os componentes podem ser arrastados para uma área de construção visual, que representa o código HTML e CSS gerado automaticamente. O painel direito permite selecionar e modificar estilos dos componentes, enquanto o painel esquerdo contém opções de configuração da página, como título e descrição. Alterações no código HTML e CSS podem ser feitas diretamente através de um editor de texto simples. A Figura 16 dá destaque à área de construção da página no editor.

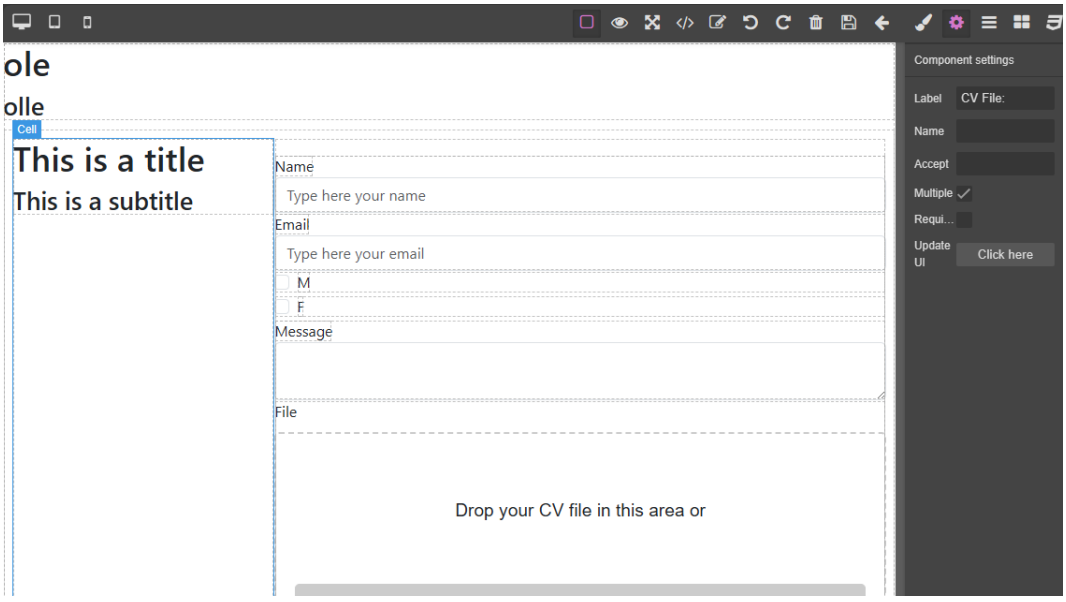


Figura 16 - A interface do editor GrapesJS, adaptado num pacote Laravel.

A documentação de GrapesJS é completa e está disponível na sua página Web [46]. A instalação é simples e requer apenas algumas configurações, permitindo uma implementação rápida e eficiente. O *wrapper* e os detalhes de integração estão disponíveis no GitHub [47].

A comunidade Laravel tem desenvolvido vários pacotes para ultrapassar as limitações dos editores WYSIWYG, muitos dos quais baseados em GrapesJS, destacando a sua popularidade.

6.2.12. Laravel Pagebuilder

Laravel PageBuilder é uma ferramenta que cria páginas utilizando componentes reutilizáveis. Este pacote Laravel, baseado no PHPageBuilder (PHP) [48] e no GrapesJS (JavaScript) [45], oferece uma interface intuitiva para a criação de páginas visuais [49].

Além de ter uma interface simples, permite a personalização do editor, tanto visualmente quanto na configuração de módulos, e inclui uma página no portal CMS com todas as páginas criadas.

A Figura 17 mostra o editor de páginas, que possui um *design* semelhante ao do GrapesJS. A barra lateral é bastante abrangente, com uma grande coleção de *templates* e componentes. As ferramentas de construção são semelhantes ao editor desenvolvido em JavaScript.

Figura 17 - O editor PHPageBuilder, baseado na biblioteca GrapesJS [50].

Para mais informações acerca deste produto, está disponível documentação no repositório GitHub [49] [48] e na página Web [50].

6.2.13. VvvebJS

VvvebJS é uma ferramenta de edição de páginas visual que permite a criação de *layouts* dinâmicos num curto período. Ele oferece uma interface gráfica amigável, onde os utilizadores podem arrastar e largar componentes para construir as suas páginas, sem a necessidade de programar [51].

Esta biblioteca possui diversos componentes pré-definidos que ajudam a criar páginas simples, como formulários, *sliders*, estruturas e tabelas, entre outros. No entanto, caso seja necessário editar algum componente ou aplicar um novo componente, existe a possibilidade de editar o HTML e CSS diretamente no editor. Neste caso, a ferramenta destina-se a indivíduos com capacidades técnicas. A Figura 18 contém um exemplar de um editor VvvebJS, onde se destacam as páginas criadas à esquerda, a estrutura à direita e o resultado, em HTML e CSS, por baixo.

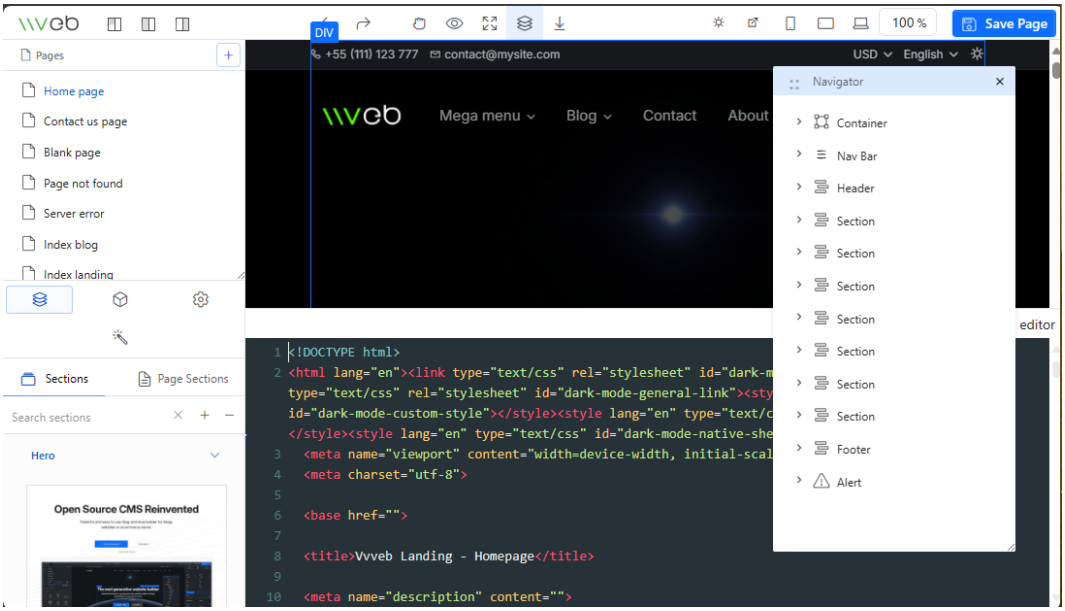


Figura 18 - Apresentação do editor [52].

Outras funcionalidades interessantes são o suporte a diversas *frameworks* de CSS e a exportação de código para outras plataformas. Também permite personalização, mas entende-se com limitada relativamente a GrapesJS.

Os detalhes podem ser vistos no exemplo gerado na sua página Web [52] e no repositório GitHub [53], onde são aplicadas atualizações regulares recentemente.

6.3. Tabelas de comparação

Terminada a apresentação dos produtos capazes em ajudar a melhorar o EasyWeb, é necessário indicar os destaques de cada componente e, de seguida, mostrar uma tabela final para comparação direta, de forma a obter uma visão mais clara sobre as diferenças entre eles.

Estas tabelas revelam um resumo das capacidades que as bibliotecas oferecem, podendo ser uma fonte de ajuda na decisão de qual a melhor solução para o EasyWeb.

6.3.1. Componentes para carregamento de ficheiros

Como o componente de *upload* de ficheiros tem de ser desenvolvido de raiz, os outros componentes analisados servem para fornecer informações e funcionalidades importantes para o desenvolvimento.

No Dropzone.js, a base da biblioteca fornece as ideias essenciais para começar o desenvolvimento do componente. Por estar desenvolvido em JavaScript e sem usar *frameworks*, destaca-se relativamente aos demais quanto à facilidade na programação do componente.

Em FilePond, o destaque vai para a interface, onde se explora as tendências mais modernas do *design* Web neste tema. Além disso, a indicação da percentagem de carregamento e a possibilidade em remover da lista são outros pontos fortes deste componente, podendo ser novas ideias para o novo componente a desenvolver.

Em Uppload.js, a aplicação de *plugins* no componente é uma funcionalidade que mais se sobrepõe, principalmente no carregamento de imagens provenientes de serviços externos e na capacidade em editar imagens. A integração de *plugins* é uma funcionalidade não pensada para a primeira fase de desenvolvimento deste projeto, mas é considerada um bom tópico a apostar noutros momentos.

A biblioteca React-Dropzone é utilizada apenas em projetos React. Além disso, não domina novas funcionalidades relacionadas com o carregamento de ficheiros, o que torna a biblioteca apenas um exemplo para este projeto.

Para realizar uma comparação mais detalhada, uma tabela pode ajudar. Na Tabela 1, existe um resumo das capacidades dos componentes deste tema, realizando comparações relativamente a algumas funcionalidades.

Tabela 1 - Tabela de comparação entre produtos que permitem carregamento de ficheiros.

Critério	Dropzone.js	FilePond	Uppload.js	React-Dropzone
Facilidade de Integração	Alta	Média	Alta	Alta
Suporte a múltiplos ficheiros	Sim	Sim	Sim (mas local)	Sim
Interação com serviços externos	Não	Sim	Sim (uma imagem por vez)	Não
Edição de imagem	Não	Sim (mais avançado)	Sim (*plugin*, uma imagem por vez)	Não
Compatibilidade com outras *frameworks*	Sim (JavaScript)	Sim (JavaScript)	Sim (JavaScript)	Sim (React)
Personalização	Alta	Alta	Média	Alta
Licença	MIT	MIT	MIT	MIT

6.3.2. Componentes para gestão de ficheiros de servidores FTP

Este módulo, tal como o anterior, tem de ficar implementado sem dependências externas, com apenas programação pura desde o início. Para tal, as soluções apresentadas têm o intuito de ajudar a identificar funcionalidades essenciais na interação com os ficheiros existentes nos servidores FTP.

Em Webix File Manager, as funcionalidades básicas existentes mostram que a biblioteca está preparada para soluções simples, sem a adição de complexidades.

Observando a biblioteca DevExtreme FileManager, destacam-se a sua interface mais moderna e personalizável e a sua riqueza em funcionalidades. As vantagens existentes estão disponíveis de acordo com o impacto da licença.

Na biblioteca Syncfusion File Manager, apresenta um equilíbrio entre as bibliotecas anteriores, nomeadamente nas funcionalidades avançadas e a facilidade de utilização. Possui também licença, podendo aproveitar a comunitária.

A Tabela 2 contém várias informações importantes dos componentes para gestão de ficheiros, realizando comparações seguindo as funcionalidades apresentadas à esquerda.

Tabela 2 - Tabela de comparação entre produtos que permitem gerir ficheiros de servidores FTP.

Critério	Webix File Manager	DeveExtreme FileManager	Syncfusion File Manager
Interface de utilizador	Intuitiva	Moderna	Familiar
Funcionalidades avançadas	Básicas	Avançadas	Avançadas
Integração com *frameworks*	Uso de API	*Frameworks* JavaScript	*Frameworks* JavaScript, outras
Suporte e Documentação	Variável	Extenso	Abrangente
Personalização	Simples	Extensa	Extensa
Operações de ficheiros	Simples	Avançadas	Avançadas
Licenciamento	Diversos planos	Comercial	Comunitária

6.3.3. Componentes para gestão de tabelas (foco no arrasto de linhas)

Tal como o componente de carregamento de ficheiros e o módulo de gestão de ficheiros de servidores FTP, a base do componente de gestão de tabelas tem de ser desenvolvida desde o início, sem a utilização de bibliotecas. Por isso, os produtos aqui analisados são exemplos que podem fornecer ideias para o componente a desenvolver.

No table-dragger, a sua simplicidade na interface e na programação são dois pontos que se destacam e que são úteis para a criação das funcionalidades gerais de arrasto na tabela.

Sobre SortableJS, esta biblioteca não se foca diretamente nas tabelas, mas em listas. Contém todas as funcionalidades de *drag-and-drop*, mas está preparado para ordenar listas e grelhas. No entanto, os conceitos usados também podem ser úteis.

Material React Table é a biblioteca mais completa desta lista e contém variadíssimas funcionalidades, para além do arrasto. Destaca-se pela interface mais moderna e na aplicação de outras funcionalidades, por vezes vindas por *plugins*.

A Tabela 3 contém várias informações importantes dos componentes para gestão de tabelas, realizando comparações seguindo as funcionalidades apresentadas à esquerda.

Tabela 3 - Tabela de comparação entre produtos que permitem gerir tabelas com arrasto de linhas.

Critério	table-dragger	SortableJS	Material React Table
Facilidade de Integração	Alta	Alta	Média
Suporte de *drag-and-drop*	Sim	Sim	Sim
Compatibilidade com *frameworks*	Sim (JavaScript)	Sim (JavaScript)	Sim (React)
Personalização	Média	Alta	Alta
Desempenho	Alta	Alta	Média
Licença	MIT	MIT	MIT

6.3.4. Componentes para construção de páginas dinâmicas

Devido à complexidade em construir um componente de criação de páginas através de arrasto de blocos, neste caso concreto, o objetivo passa pela utilização e adaptação de um componente já existente. Para tal, é importante realizar uma análise entre os três produtos para saber as vantagens e desvantagens e, mais tarde, decidir qual utilizar.

Ao explorar a interface do editor de Laravel-GrapesJS, visível na Figura 16, nota-se a simplicidade de uso: à direita, há um painel de componentes que podem ser arrastados para a área de construção visual à esquerda, onde o HTML e o CSS são gerados automaticamente. Os componentes podem ser reorganizados facilmente e os seus estilos podem ser personalizados diretamente na interface, sem necessidade de editores de código tradicionais (mesmo estando lá para algo mais complexo).

O Laravel PageBuilder, embora semelhante ao Laravel GrapesJS, diferencia-se pela sua estrutura e aparência. Visualmente, é baseado na biblioteca Bootstrap e incorpora uma versão modificada do GrapesJS para o controlo e construção das páginas. Além disso, o pacote inclui uma secção dedicada à gestão de páginas, facilitando a criação e edição de conteúdo diretamente na interface. Contudo, no caso do EasyWeb, a funcionalidade tem de ficar desativada, porque já possui uma secção própria para gestão de páginas [48].

Por fim, VvvebJS é uma biblioteca de JavaScript que possui algumas diferenças relativamente ao GrapesJS e está mais focada para projetos mais simples e rápidos. Uma das funcionalidades que mais se destaca é o acesso às páginas criadas, cuja secção no editor se encontra à esquerda. Todavia, não existe nenhum pacote Laravel que inclua o VvvebJS, o que torna a integração mais complexa e demorada.

A Tabela 4 aparenta ser um excelente exemplo para observar comparações entre os produtos para construção de páginas Web.

Tabela 4 - Tabela de comparação entre produtos que permitem criar páginas dinâmicas.

Critério	Laravel-GrapesJS	Laravel Pagebuilder	VvvebJS
Facilidade de Uso	Média (requer conhecimento técnico)	Alta (focado na simplicidade)	Alta (simples e intuitivo)
Flexibilidade e Personalização	Média (com foco em Laravel)	Média (focado em componentes Laravel)	Média
Componentes pré-definidos	Avançados e extensíveis	Média (baseado em Laravel)	Básicos
Edição de código	Sim (HTML, CSS, JavaScript)	Sim (HTML, CSS)	Sim (HTML, CSS)
Suporte a *frameworks* CSS	Bootstrap, Foundation, personalizado	Bootsrap, personalizado	Bootstrap
***Plugins* e extensões**	Extensível com ampla biblioteca e Laravel	Limitado (focado em Laravel)	Limitado
Exportação de código	Sim	Sim	Sim
Comunidade e suporte	Menor (no GrapesJS, é maior)	Maior	Maior
Integração com CMS	Avançada	Avançada	Básica
Licenças	MIT	MIT	MIT

Depois de estudar as principais características entre os produtos apresentados, considera-se que o pacote Laravel-GrapesJS oferece melhores condições para se adaptar à plataforma

EasyWeb, porque, para além de ser uma biblioteca adequada para o projeto EasyWeb (uso de Laravel como *framework*), apenas fornece o editor e permite uma maior personalização dos blocos para as páginas Web e da própria interface do editor. Para esta fase, a interface é o tema onde o desenvolvimento não irá influenciar, mas poderá ser em momentos futuros.

7. Implementação

A parte prática está concentrada neste capítulo, que delineia os processos de prototipagem, as tecnologias de programação utilizadas e a integração com o EasyWeb. Representa a fase crucial em que os conceitos teóricos e planos estratégicos se transformam em soluções tangíveis e funcionais.

Cada subcapítulo representa um componente desenvolvido, nomeadamente o componente de *upload* de ficheiros no capítulo 7.1, o módulo de gestão de ficheiros no 7.2, o componente de gestão de tabelas no 7.3 e o componente de construção de páginas Web no 7.4. Para cada subcapítulo relativo a cada um dos componentes, será descrito o processo de prototipagem, a implementação, a integração no CMS EasyWeb e a utilização do componente.

Prototipagem

É essencial a existência de protótipos no desenvolvimento e implementação de conteúdo, visto que permite validar requisitos, fornecer *feedback* antecipado, acelerar determinados processos, diminuir riscos, entre muitos outros. Portanto, pretende-se divulgar diversos *mockups* para cada requisito do projeto, com a intenção de pré-visualizar interfaces visuais selecionadas para a implementação.

Implementação

Após a apresentação das interfaces dos componentes, as secções de implementação irão mostrar as formas utilizadas para a implementação em ambientes locais e fechados, como também alguns pormenores.

Integração com o EasyWeb

As secções de integração com o EasyWeb irão ilustrar as técnicas utilizadas para a integração dos componentes, desenvolvidos num ambiente separado, no CMS, porque, após o desenvolvimento dos componentes, estes encontram-se prontos para serem integrados e testados. Devido a processos específicos existentes, possíveis adaptações poderão ser feitas para aceitar minimamente todas as funcionalidades requeridas.

Utilização dos componentes

O objetivo é mostrar aos utilizadores finais como usar os componentes, principalmente para os programadores. Pretende-se apresentar um manual, expondo, de forma resumida, os principais pontos.

Testes de carga

Por fim, este bloco ilustra diversos testes realizados no componente de *upload* de ficheiros, com a intenção de obter conclusões relacionados com o desempenho tanto a nível local como em servidores *online*.

7.1.Componente de *upload* de ficheiros

Neste subcapítulo, irá ser descrito o componente de *upload* de ficheiros desenvolvido, mostrando a prototipagem, a implementação, a utilização e alguns testes de carga. O componente também é designado por Drag-Drop-Files-Upload.

7.1.1. Prototipagem

Foi construído um esboço da interface visual que cumprisse os requisitos necessários, cujo protótipo se apresenta na Figura 19.

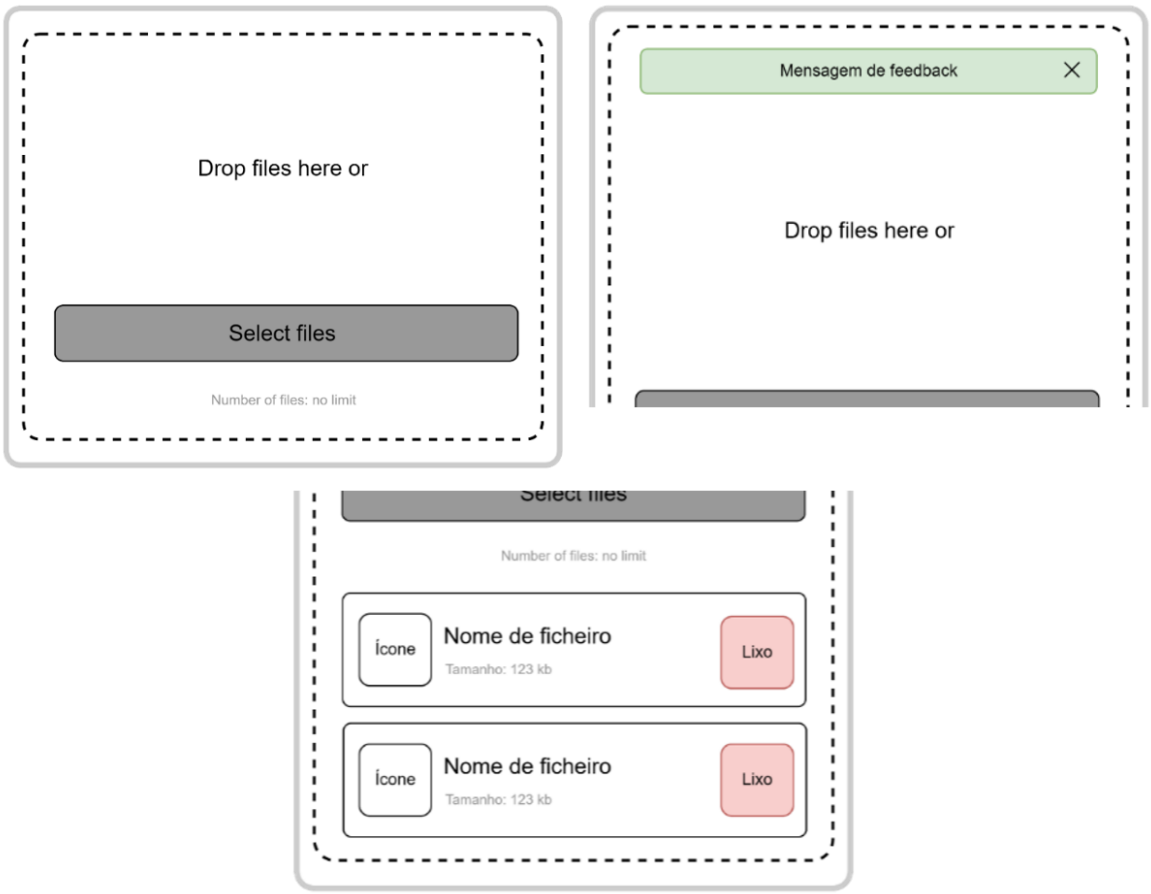


Figura 19 - Páginas que compõem o protótipo do componente.

Observado as imagens da Figura 19, o componente está dividido em duas partes: a área de importação dos ficheiros (em cima) e a representação daqueles que foram inseridos, tanto

na forma manual como através do arrasto. O botão de carregamento, que se encontra na divisão das duas secções, apenas permite o carregamento manual, com a utilização de caixas de diálogo do navegador Web. Mais abaixo, há uma indicação sobre o número máximo de ficheiros que podem ser inseridos.

Na secção dos ficheiros carregados, em cada um dos blocos que representam as informações essenciais, existe um botão que oferece a funcionalidade de remoção na lista.

Durante o processo de inserção, o componente apresentará informações sobre o estado, para que o utilizador perceba, em tempo real, se o ficheiro está a ser ou não aceite, porque existem verificações relacionadas com a duplicação, com as extensões aceites (indicadas no componente de carregamento), entre outros.

7.1.2. Implementação

O elemento HTML *<input type="file">* [54] é responsável por gerir o carregamento de ficheiros. Este elemento é limitado, porque possui uma interface visual que não permite observar facilmente múltiplos ficheiros inseridos, como está descrito na Figura 20.

Figura 20 - O componente de upload de ficheiros por omissão.

Passando o ponteiro do rato por cima, é possível observar a lista dos ficheiros, conforme visível na Figura 21.

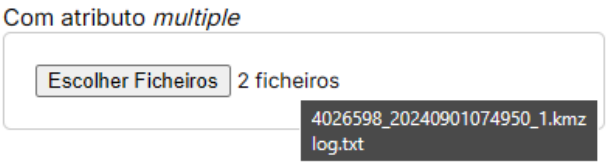


Figura 21 - Os nomes dos ficheiros ficam visíveis ao passar o ponteiro do rato por cima do elemento.

As limitações podem ser ultrapassadas com a aplicação de código JavaScript e de algum CSS, aumentando a capacidade em manipular a parte visual. Mantendo o funcionamento nativo do elemento (*drag-and-drop*, eventos, etc.), é possível ver mais detalhes de cada ficheiro, permitir a remoção na lista ou até mostrar o progresso do *upload* em tempo real.

Organização dos ficheiros utilizados

A Figura 22 mostra a estrutura do sistema de ficheiros que engloba o componente, sob a forma de *plugin*.

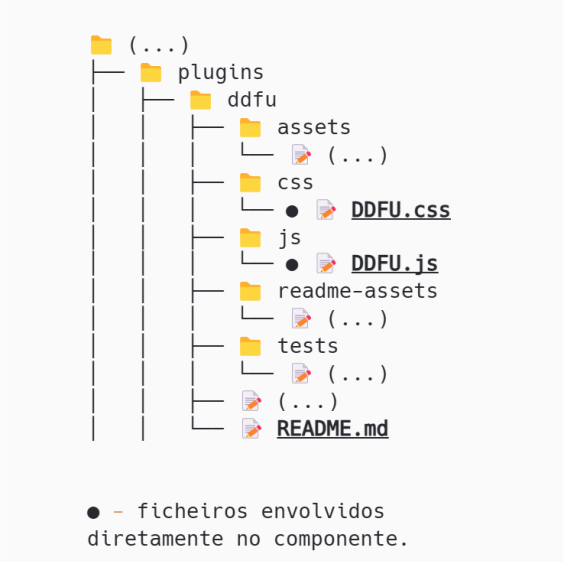


Figura 22 - Organização dos ficheiros associados ao componente.

De realçar que estes ficheiros foram inseridos dentro da pasta de *plugins* do projeto EasyWeb, em que os ficheiros indicados com carater ● são os ficheiros envolvidos na construção do componente.

O componente é constituído por um ficheiro de estilos (com a extensão .css), por um ficheiro de código JavaScript (extensão .js) e por um manual de utilização (extensão .md), para além de outros menos importantes.

O manual tem o objetivo de fornecer informações sobre as características do *plugin*, do seu funcionamento e da instalação em projetos Web. Os detalhes podem ser observados em Anexo C – Manual de Utilizador do Componente *Upload* de Ficheiros.

Desenho

Para o funcionamento do componente DDFU, é necessário implementar uma estrutura HTML específica, porque é exigida a utilização de classes especiais, um ID específico e novos atributos criados. A Figura 23 mostra toda a estrutura HTML que o componente necessita, onde são representadas duas secções, separando o que é obrigatório inserir durante a implementação e o que é gerado pelo JavaScript.


```
<div class="ddfu_drop-area" id="(...)" ddfu-action="(...)" ddfu-required="">
    <p>Drop files in this area or</p>
    <input type="file" name="(...)" id="(...)"
           title="(...)" accept="(...)" disabled="">
    <label for="(...)" class="ddfu_button">Select File</label>
    <p class="ddfu_counter" style="color: green;">Max files number: 1</p>
    <div class="ddfu_gallery">
        <div id="file-preview-0" class="ddfu_file-preview">
            <div title="pdf_title.pdf">
                <svg (...)>(...)</svg>
            </div>
            <div class="file-data">
                <span class="filename" title="File name: pdf_title.pdf\nSize: 2.40 MB">
                    pdf_title.pdf
                </span>
                <span class="filesize">Size: 2.40 MB</span>
            </div>
            <span class="delete-btn" title="Delete file">
                <img src="../assets/Trash_Can.png" alt="Trash can" width="20">
            </span>
        </div>
    </div>
</div>
```

Figura 23 - Estrutura HTML que define o desenho do componente.

A classe *ddfu_drop-area* é obrigatória, porque permite aplicar os estilos nos elementos sucessores. O ID serve para distinguir diferentes instâncias do componente existentes numa página Web e é utilizado para iniciar o funcionamento da sua instância. O atributo *ddfu-action* indica o URL que dá início ao processo de *upload* no servidor, depois de submeter.

Segundo a Figura 23, a secção a cor verde contém apenas os elementos <*input*> e alguma decoração que identifique a zona como espaço de *drop* de ficheiros. A outra zona mostra a lista dos ficheiros e indica o número máximo de ficheiros que o DDFU aceita.

A Figura 24 mostra um exemplo de estilos CSS aplicados no elemento HTML com a classe *ddfu_drop-area*.

```css
.ddfu_drop-area {
  border: 2px dashed #ccc;
  border-radius: 5px;
  font-family: sans-serif;
  text-align: center;
  font-size: 1.2rem;
  margin: 10px auto;
  padding: 20px;

  display: flex;
  flex-direction: column
  justify-content: center;
  align-items: stretch;
  gap: 1rem;
}

(...)
```

Figura 24 - Estilos CSS aplicados ao elemento HTML com a classe *ddfu_drop-area*.

Observando a Figura 24, os estilos não envolvem apenas na formatação dos elementos, mas também no *layout* no componente.

O resultado visual do componente DDFU acaba por tornar o componente mais único, como está visível na Figura 25.

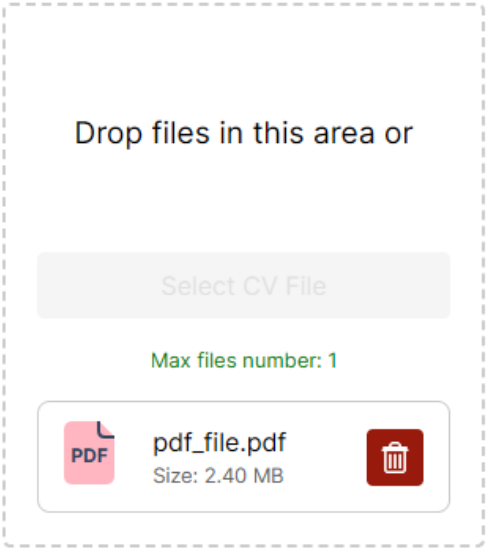


Figura 25 - Resultado visual, após aplicar os estilos.

Suporte a drag-and-drop

A implementação do *drag-and-drop* envolve a aplicação de modificações ao elemento *<input type="file">*, para permitir a adição de novas funcionalidades, como a lista dos ficheiros. Para que tal aconteça, alguns eventos que estão associados ao *drag-and-drop* sofrem bloqueios quanto aos comportamentos nativos, como mostra a Figura 26.

```
["dragenter", "dragover", "dragleave", "drop"].forEach((eventName) => {
    this.selectedDropArea.addEventListener(eventName, this.preventDefaults, false);
});
```

Figura 26 - A desativação dos comportamentos nativos dos eventos indicados é feita desta forma, no código.

Ao inserir um ficheiro, através do arrasto ou do método tradicional, vários processos são executados para realizar validações. Primeiro, verifica se o <*input*> aceita múltiplos ficheiros; depois, verifica se a extensão ou tipo de ficheiro é aceitável para o formulário em questão. Se passar nas validações do componente, o ficheiro é registado e apresentado na lista.

Processo de upload

Todo o processo de *upload* é tratado dentro do JavaScript, dando início imediatamente após o clique no botão de submissão do formulário. No entanto, os valores inseridos nos campos do formulário apenas são enviados para o servidor depois de terminar o *upload*.

É essencial implementar o atributo *ddfu-action* no componente, porque indica o URL que recebe os ficheiros carregados. O atributo *ddfu-required* não é obrigatório, mas foi necessário para substituir o *required* do elemento <*input type=”file”*>, porque não consegue ser focalizável durante as validações. Esta situação acontece devido ao facto de o elemento HTML original tem de estar escondido, para aplicar novos estilos ao componente. A Figura 27 exemplifica uma situação que não permite aplicar o atributo *required* como pretendido.

```
An invalid form control with name='file[]' is not focusable.    localhost/:1
<input type="file" name="file[]" id="file1" title="CV File" accept=".pdf,.do
c,.docx" multiple required style="display: none;">
```

Figura 27 - Mensagem de erro ao aplicar o atributo *required* no elemento forçadamente escondido.

Para que o *upload* seja executado antes de realizar a submissão do formulário, a função que inicia o funcionamento do JavaScript tem de bloquear os comportamentos nativos de submissão e aplicar novos, onde se inclui a parte de *upload* dos ficheiros.

Como é removida a validação que traz por padrão, o código tem de validar cada um dos campos que tenham o atributo *required*, incluindo o próprio componente. Ou seja, tem de replicar o comportamento nativo com o código que Figura 28 mostra. Primeiro, verifica os outros campos e, mais tarde, verifica o componente.

```
this.selectedForm.onsubmit = (e) => {
  this.preventDefaults(e);   <-- Remoção do comportamento nativo da submissão

  const allRequiredFields = this.selectedForm.querySelectorAll(
    "input[required], textarea[required], select[required]"
  );
  for (const field of allRequiredFields) {
    if (!field.validity.valid) {
      field.reportValidity();
      (...)
      break;
    }
  }                          Validação dos outros campos obrigatórios de um formulário

  if (this.fileInputFiles.length == 0 && this.hasRequiredAttribute) {
    (mensagem de erro)
  } else {
    if (this.fileInputFiles.length > 0) {
      (processo de upload)
    } else {
      this.selectedForm.submit();
    }
  }                                        Validação do componente de upload
};
```

Figura 28 - Implementação da validação do formulário através de JavaScript.

Quando o formulário está validado e correto, é iniciado o processo de *upload*. Através da tecnologia API XMLHttpRequest [55], os ficheiros são separados em pedidos, ficando cada um associado a um pedido HTTP. A Figura 29 mostra como foi feita a separação.

```
for (let i = 0; i < numberFiles; i++) {
  (...)

  let formData = new FormData();
  formData.append("file", files[i]);

  let request = new XMLHttpRequest();

  request.open("POST", this.actionURL);

  (...)

  request.beforeSend = (xhr) => {
    xhr.setRequestHeader(
      "X-CSRFToken",
      document.querySelector("meta[name='csrf-token']").content
    );
  };

  (...)

  };

  request.send(formData);
}
```

Figura 29 - Implementação da criação de um pedido para cada ficheiro.

O XMLHttpRequest tem um evento que permite saber, em tempo real, a percentagem de *upload* realizado. Como oferece *feedback* de cada um em simultâneo, o componente gera uma barra de progresso para cada ficheiro e apresenta essa percentagem, como se pode ver na Figura 30.

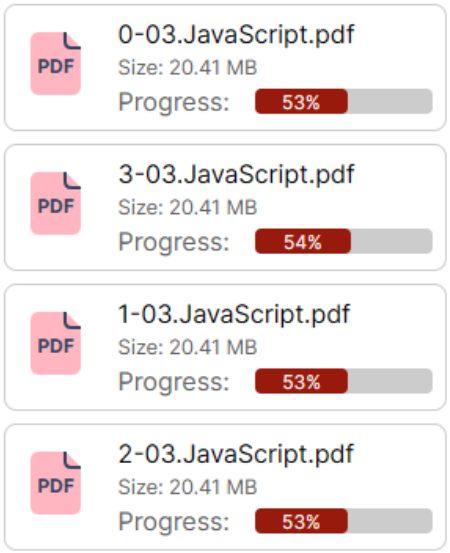


Figura 30 - As barras de progresso são criadas quando o *upload* é iniciado.

O valor da percentagem é obtido com o seguinte código (Figura 31):

```
for (let i = 0; i < numberFiles; i++) {
  (...)

  let formData = new FormData();
  formData.append("file", files[i]);

  let request = new XMLHttpRequest();

  (...)

  request.upload.onprogress = (e) => {
    if (e.lengthComputable) {
      let percentage = (e.loaded / e.total) * 100;
      this.onUploadProgress(i, percentage);
    }
  };

  (...)

  request.send(formData);
}
```

Figura 31 - Obtenção do valor da percentagem de cada pedido criado.

Depois de obter as respostas de todos os pedidos gerados, o processo de *upload* está terminado. Desta forma, o formulário está pronto para ser submetido. O *upload* é executado

inicialmente, porque permite ter mais controlo dos ficheiros inseridos. Assim, caso haja algum erro durante o carregamento, a submissão do formulário não é realizada. Caso os ficheiros tenham grandes dimensões e sejam utilizados imediatamente após a submissão, poderá ocorrer problemas.

De notar que, mesmo que realize duas submissões distintas, os valores originais do formulário são mantidos, incluindo os ficheiros. Portanto, os nomes dos ficheiros podem novamente ser usados, não havendo a necessidade de realizar outra vez o *upload*.

Processo de upload *no servidor*

Quando há a inserção de múltiplos ficheiros, apesar do processo de *upload* no JavaScript ser assíncrono, cada pedido originado apenas inclui um ficheiro. Desta forma, os ficheiros ficam independentes entre si. Portanto, no servidor, a função PHP associada apenas trata um ficheiro de cada vez, verificando isso na Figura 32.

```php
function uploadFiles($location, $files)
{
    $results_array = null;
    if (is_array($files['tmp_name'])) {
        $results_array = [];
        $number_files = count($files['tmp_name']);
        for ($i = 0; $i < $number_files; $i++) {
            $newPath = $location . '/' . $files['name'][$i];
            $result = move_uploaded_file($files['tmp_name'][$i], $newPath);
            if ($result) {
                $results_array[] = $result;
            } else {
                return false;
            }
        }
    } else {
        $newPath = $location . '/' . $files['name'];
        $results_array = move_uploaded_file($files['tmp_name'], $newPath);
    }

    if (is_array($results_array)) {
        if (in_array(false, $results_array)) {
            return false;
        }
        return true;
    }
    return $results_array;
}
```

Figura 32 - Código PHP que define, no servidor, o processo de *upload*.

Segundo a Figura 32, não foi utilizada nenhuma *framework* PHP para a implementação do processo de *upload* no servidor, mostrando que o componente aceita qualquer programação de *back-end*.

7.1.3. Utilização do componente

O componente DDFU contém um manual de utilização, com a intenção de fornecer todos os pormenores existentes no código, bem como explicar o método de implementação em projetos. Para ver mais detalhes, a informação está disponível em Anexo C – Manual de Utilizador do Componente *Upload* de Ficheiros.

Para que o componente DDFU fique acessível na página Web, é necessário aplicar elementos HTML com atributos específicos, como a Figura 33 sugere.

```
<form (...)>

    (...)

    <div id="(...)" class="ddfu_drop-area" ddfu-action="(...)">
        <p>Drop files in this area or</p>
        <input type="file" name="file[]" id="file" title="File" accept="(...)">
        <label for="file" class="ddfu_button">Select file</label>
    </div>

    (...)

</form>
```

Figura 33 - Implementação do componente na estrutura HTML de uma página.

Dentro do elemento <*div class=" ddfu_drop-table"*>, é possível adicionar mais elementos, desde que os elementos <*label*> e <*input*> estejam sempre presentes.

Esta estrutura HTML é insuficiente para dar início ao funcionamento do componente DDFU. Para tal, é necessário executar algum código JavaScript, como está na Figura 34.

```
import DDFU from "./js/DDFU.js";

const index = (function () {
  (...)

  // initialize DDFU.js
  const ddfu_form = new DDFU("<id do componente>");
  ddfu_form.init();

  (...)
})();
```

Figura 34 - Execução do DDFU é feita invocando a sua classe e iniciando o funcionamento.

Dentro do código base, alguns atributos aplicados no elemento <*input*> podem modificar a interface dinamicamente, como é o caso do atributo *multiple*. Deve-se utilizar o *ddfu-*

required no elemento antecessor quando se pretende tornar obrigatório o preenchimento do campo.

As configurações gerais são feitas dentro do ficheiro JavaScript, podendo ser necessário aceder para adaptar o componente em cenários específicos.

7.1.4. Testes de carga

Tendo em conta a importância do desempenho, em especial no processo de *upload*, é importante realizar testes de carga para verificar o comportamento do componente.

Implementação dos testes

Os testes foram baseados no tempo de duração do *upload* total. Para os obter, foi aplicado algum código JavaScript no componente, como aparece na Figura 35.

```
uploadFiles = () => {
    (...)

    // get time start in seconds
    let timeStart = Math.floor(new Date().getTime());

    for (let i = 0; i < numberFiles; i++) {
        (...)

        let request = new XMLHttpRequest();

        (...)

        request.onloadend = () => {
            let timeEnd = Math.floor(new Date().getTime());

            this.showAlert(
                this.selectedDropArea,
                "Total time: " + (timeEnd - timeStart) + " milliseconds",
                "information",
                10
            );

            (...)
        };

        request.send(formData);
    }
};
```

Figura 35 - Implementação do tempo de duração de cada pedido criado para cada ficheiro.

Existem diversas formas para obter os valores de tempo, mas, como estes testes são simples, esta foi uma maneira mais fácil para os conseguir ter, porque já vem incluído na biblioteca nativa do JavaScript [56]. Houve um cuidado em colocar o tempo final depois de terminar cada *upload*, visto que os pedidos são assíncronos por padrão.

Para os testes em questão, interessa saber a duração total, necessitando apenas do último tempo obtido. A diferença obtida é entre o tempo antes de iniciar a criação dos pedidos e o tempo depois de terminar o *upload* do último ficheiro.

Execução dos testes

Para a realização de testes antes de ser implementado no módulo do EasyWeb, uma página Web simples pode ser essencial para verificar o seu funcionamento em ambientes de carregamento de ficheiros. Para tal, um servidor local foi iniciado, utilizando um comando PHP, como está na Figura 36.

```
php -S localhost:5500 -t tests
```

Figura 36 - Comando PHP que permitiu a inicialização de um servidor local.

A Figura 37 apresenta a página de testes, que contém um formulário com vários campos de texto, opções de seleção e o tal componente. Pretende-se realizar testes num cenário próximo de casos reais, como o carregamento de ficheiros de CV em formulários de recrutamento.

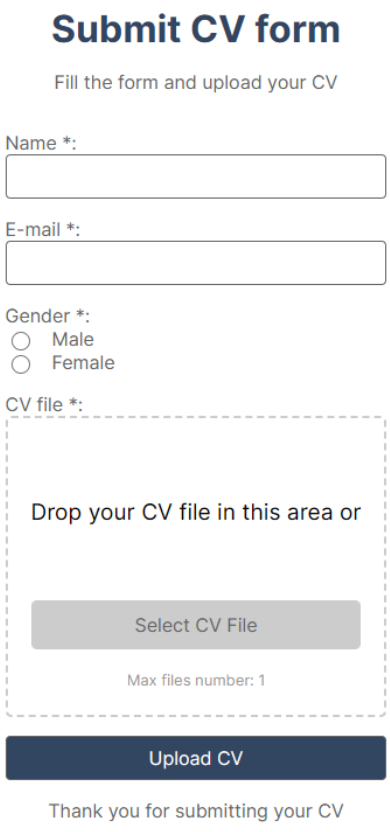


Figura 37 - Representação da página de testes do DDFU.

Carregamento de ficheiros com o mesmo tamanho

A fase de testes é iniciada, utilizando ficheiros PDF. Estes possuem distintas dimensões, para conseguir compreender como o componente reage. Inicialmente, começou-se a obter valores de duração do tempo do carregamento de um e de vários ficheiros de igual dimensão, como se pode ver na Tabela 5. O valor da dimensão do ficheiro PDF é de 20,41 MB.

Tabela 5 - Dados obtidos durante testes de carregamento de vários ficheiros de igual dimensão.

Número de ficheiros	Tamanho total (MB)	Tempo 1 (s)	Tempo 2 (s)	Tempo 3 (s)	Tempo médio (s)	Velocidade média (MB/s)
1	20,4	6,130	6,465	6,625	6,407	3,184
2	40,8	13,366	13,378	14,373	13,706	2,977
4	81,6	25,936	25,768	27,506	26,403	3,091
5	102	32,808	32,377	32,025	32,403	3,149
8	163,2	56,335	53,953	55,68	55,323	2,950
10	204	65,208	65,812	66,502	65,841	3,098

Para calcular a velocidade média de cada registo, foi aplicada a seguinte equação:

$$v = \frac{A}{t}$$

em que A corresponde ao tamanho do ficheiro e t ao tempo decorrido.

Para melhor compreensão da evolução do upload, a Figura 38 mostra a posição do tamanho total de cada carregamento face ao tempo demorado a realizar a tarefa num servidor local. A linha de tendência, marcada a tracejado, mostra uma aproximação média dos valores obtidos. Desta forma, o tempo é linear em comparação com o tamanho dos ficheiros.

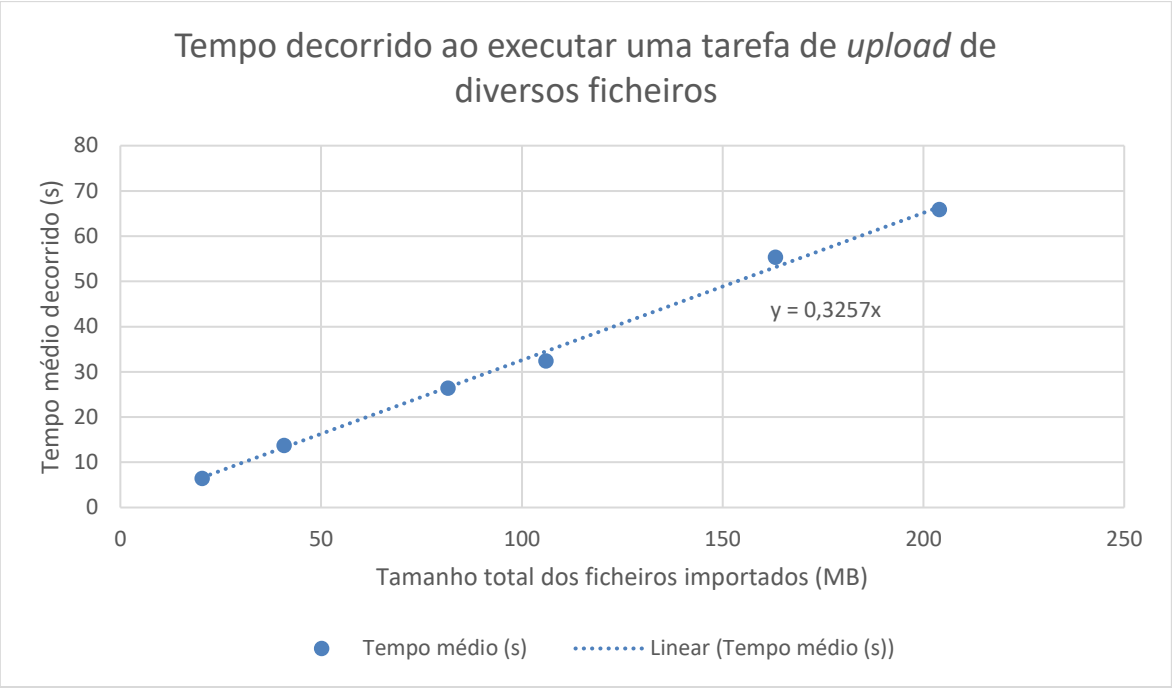


Figura 38 - Gráfico da comparação entre o tamanho total dos ficheiros e o tempo demorado a terminar a tarefa num servidor local.

Observando a equação da linha de tendência, é possível obter a velocidade, que demonstra ser um relativamente constante.

Nesta situação, o valor da velocidade média de carregamento em todas as situações é obtido na seguinte forma:

$$t = 0{,}3257A \iff 1 = 0{,}3257\frac{A}{t} \iff \frac{1}{0{,}3257} = v \iff v \approx 3{,}070\ MB/s$$

De realçar que também foi registado um valor máximo de ficheiros carregados em simultâneo. Era possível realizar *upload* de 6 ficheiros em simultâneo, em que a velocidade de *upload* de cada um diminuía quanto mais ficheiros estivessem em simultâneo. Se houver mais ficheiros na lista, esses ficam em espera.

Carregamento de ficheiros de diferentes tamanhos

Outro teste foi feito utilizando ficheiros de diferentes dimensões, cuja diferença entre tamanhos é grande. A Tabela 6 contém dados que foram obtidos a partir de dois ficheiros PDF, de 259KB e de 20,41MB. Em cada teste, existia o mesmo número de ficheiros de menor e de maior dimensão.

Tabela 6 - Dados obtidos durante testes de carregamento de vários ficheiros de diferentes dimensões.

Número de ficheiros	Tamanho total (MB)	Tempo 1 (s)	Tempo 2 (s)	Tempo 3 (s)	Tempo médio (s)	Velocidade média (MB/s)
2	20,66883	6,618	5,884	6,181	6,228	3,319
4	41,33766	12,907	12,009	12,026	12,314	3,357
6	62,00649	18,619	18,621	18,441	18,560	3,341
8	82,67532	24,374	25,871	26,088	25,444	3,249
10	103,34415	33,033	30,841	30,154	31,343	3,297
12	124,01298	35,458	36,347	36,077	35,961	3,449
14	144,68181	42,319	44,427	42,343	43,030	3,362

A Figura 39 descreve o gráfico com os dados da Tabela 6. Tal como no teste anterior, existe uma tendência linear entre o tempo e o valor total de carregamento. Este teste também foi realizado num ambiente local.

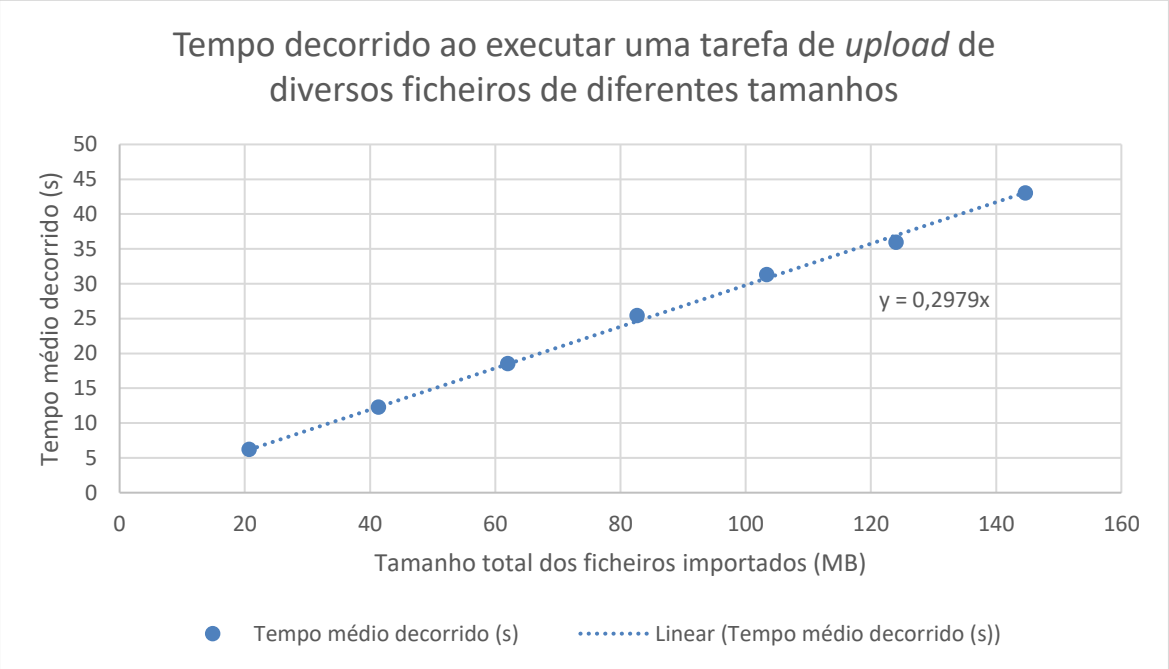


Figura 39 - Gráfico da comparação entre o tamanho total dos ficheiros e o tempo demorado a terminar a tarefa num servidor local, usando ficheiros de diferentes dimensões.

O valor da velocidade média de carregamento é obtido na seguinte forma:

$$t = 0{,}2979A \iff 1 = 0{,}2979\frac{A}{t} \iff \frac{1}{0{,}2979} = v \iff v \approx 3{,}357\ MB/s$$

Conclusões

A linearidade, em ambos os casos, tem mostrado que existe consistência ao nível do desempenho do carregamento dos ficheiros. No entanto, há sempre um certo limite, delimitado quando o servidor deixa de ter capacidade de processamento. É importante não esquecer que os testes foram realizados num ambiente controlado, num servidor local.

7.2.Módulo de gestão de ficheiros

Para que o EasyWeb consiga também ter controlo de ficheiros externos, foi necessário desenvolver um módulo para fazer a gestão de ficheiros com funcionalidades semelhantes a uma ferramenta externa. O módulo foi desenvolvido no contexto deste projeto e designou-se "File Gallery".

Neste subcapítulo, será descrito o módulo de gestão de ficheiros desenvolvido, mostrando a prototipagem, a implementação, a integração e a utilização.

7.2.1. Prototipagem

Depois de realizada uma análise de requisitos, começou-se a desenvolver um esboço da interface visual que melhor condiz com os requisitos imprescindíveis. O protótipo criado contém três páginas, cada uma com funcionalidades específicas. Contém uma página que representa o conteúdo de uma pasta (Figura 40), uma que mostra os resultados de pesquisa (Figura 41) e uma com um visualizador de ficheiros (Figura 42).

Como o CMS suporta diversos idiomas, este módulo foi preparado para funcionar em qualquer idioma selecionado no EasyWeb.

Página inicial / página com os ficheiros de uma pasta selecionada

A página principal do módulo apresenta os ficheiros e subpastas existentes na pasta raiz. A sua interface visual encontra-se visível na Figura 40.

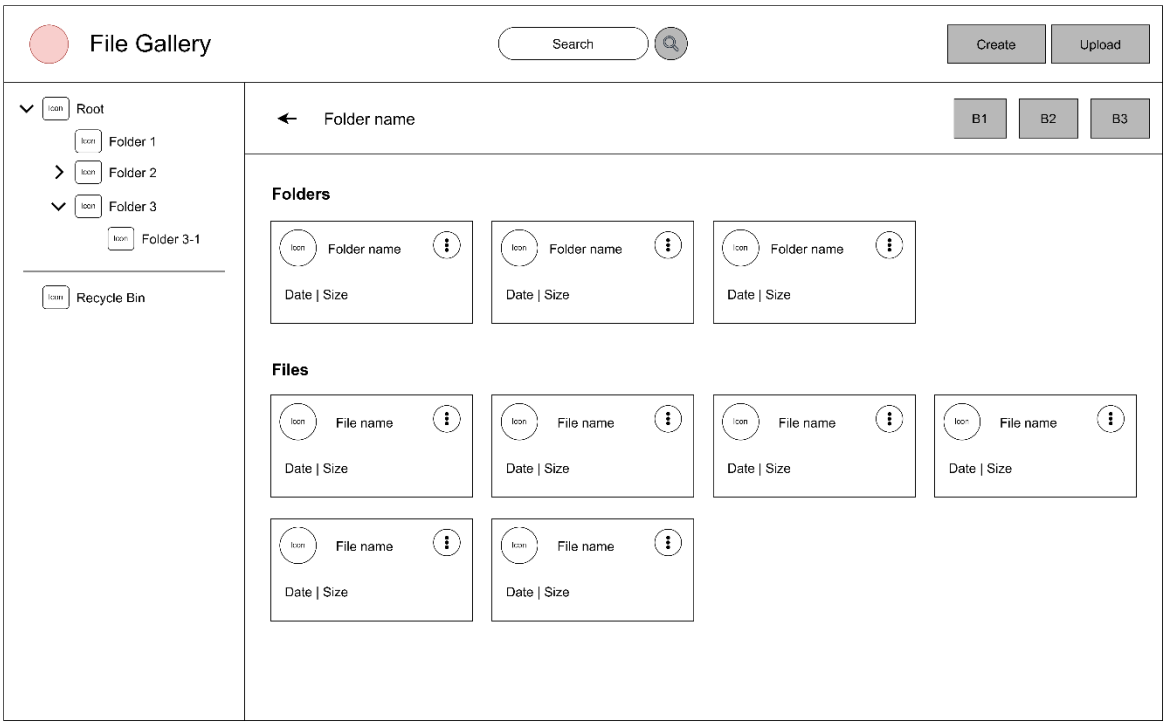


Figura 40 - Representação da estrutura dos ficheiros e pastas existentes no diretório selecionado.

Todas as páginas do gestor de ficheiros contêm uma estrutura semelhante a esta. O cabeçalho, que inclui o título, a pesquisa e um conjunto de botões, e a barra lateral, que representa a hierarquia das pastas, são as secções cuja estrutura e funcionamento são transversais a todas as páginas.

Na zona de apresentação do conteúdo existente na pasta selecionada, existem duas secções: a lista de subpastas e ficheiros existente e a zona das operações de gestão da pasta.

Página de pesquisa

Esta página é acedida durante a pesquisa por pastas ou ficheiros que possam ter, no nome, a expressão inserida no formulário.

A Figura 41 apresenta o resultado de uma pesquisa, em que a resposta da pesquisa está organizada por tipos de ficheiro, aparecendo primeiro as pastas e depois os ficheiros.

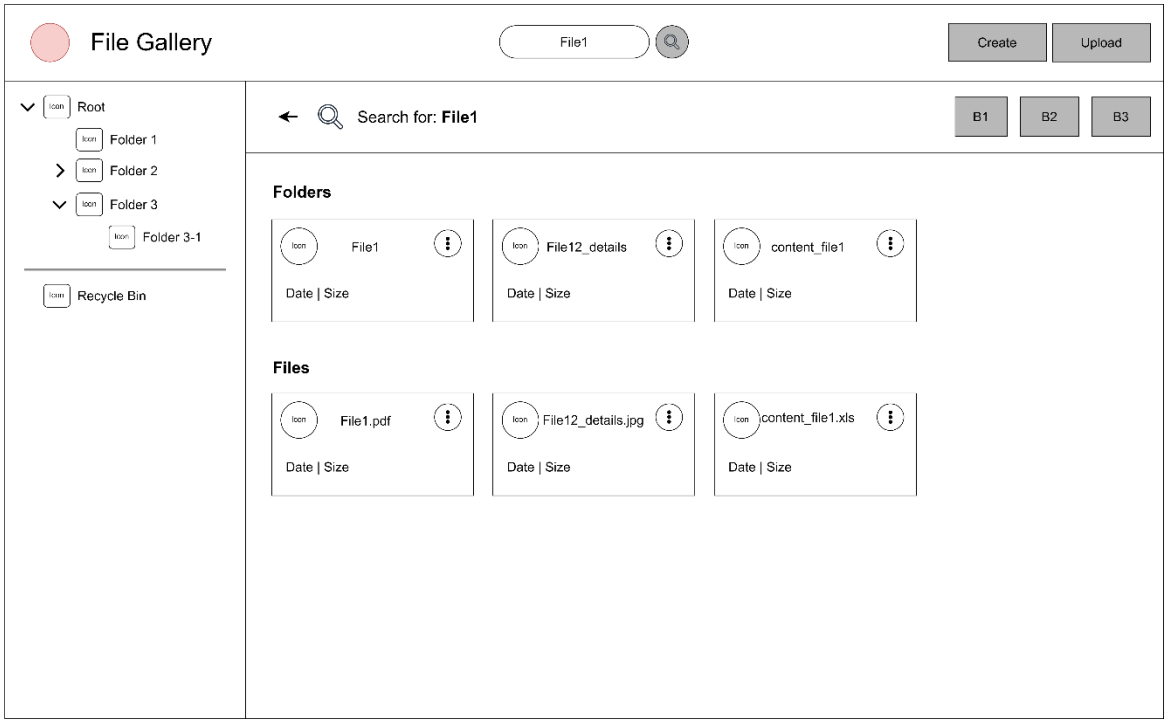


Figura 41 - Representação da estrutura da pesquisa.

Página do visualizador de ficheiros

Nesta página, os ficheiros que consigam ser interpretados num navegador Web são visualizados, por exemplo as imagens, vídeos e ficheiros de texto. Os outros ficheiros terão de ser transferidos para visualizar numa plataforma mais adequada. A Figura 42 mostra o protótipo da página destinada à visualização do ficheiro selecionado.

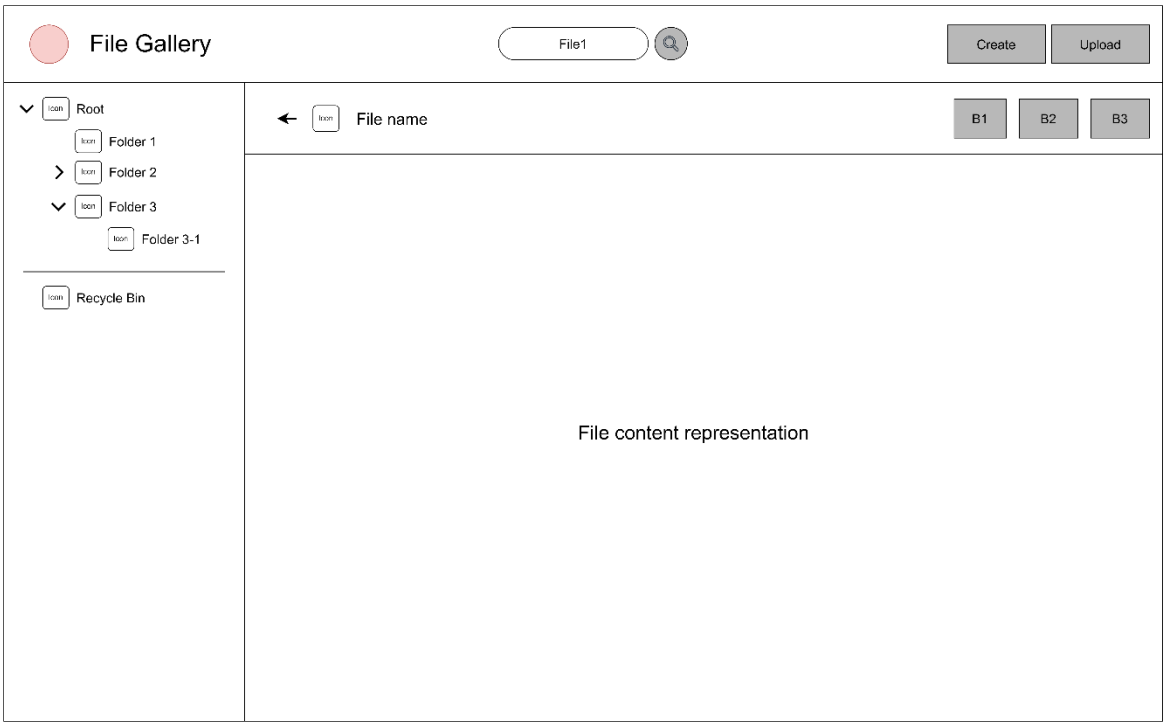


Figura 42 - Representação da estrutura do visualizador de ficheiros.

Outros pormenores/funcionalidades

Os eventos, iniciados após o clique em botões específicos, são acompanhados por caixas de diálogo (*modal*) com formulários, cujos protótipos se apresentam na Figura 43.

Figura 43 - Representação da estrutura dos diálogos de determinados eventos.

O conteúdo de cada formulário pode variar consoante o evento, porque, em algumas situações, os diálogos são apenas questões de confirmação, acontecendo quando se pretende eliminar ou transferir ficheiros.

Focando apenas no formulário de carregamento de ficheiros (imagem no canto superior esquerdo da Figura 43), observa-se que se pretende usar o componente de *upload* de ficheiros (capítulo 7.1 - Componente de *upload* de ficheiros).

7.2.2. Implementação

Visto que o EasyWeb está desenvolvido a partir da *framework* Laravel, a implementação deste módulo tem como base essa *framework* e está dividida em partes, envolvendo o servidor e a apresentação visual no navegador Web.

Organização dos ficheiros utilizados

A Figura 44 apresenta os ficheiros criados e utilizados para o funcionamento do módulo. Estes serão usados no processo de integração, podendo sofrer modificações.

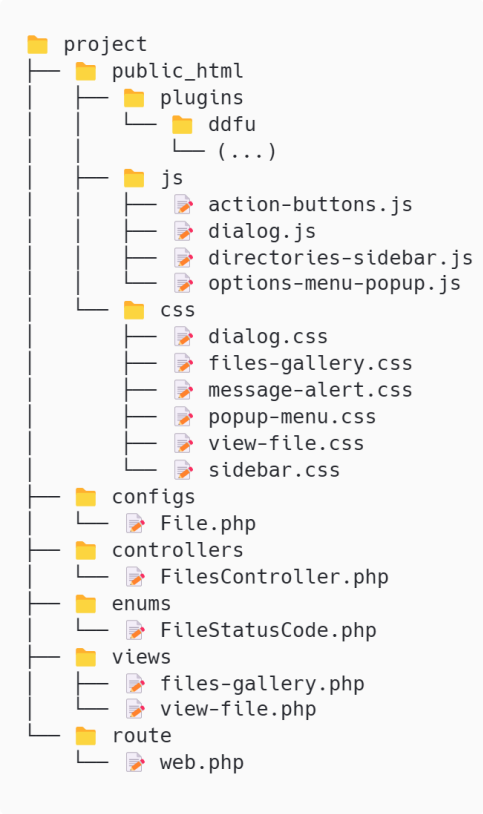


Figura 44 - Ficheiros utilizados para o desenvolvimento do módulo em ambiente local.

O módulo que gere os ficheiros foi construído sob diversos ficheiros Blade (a tecnologia para a criação de *views* do Laravel) para a página da listagem do conteúdo e para a página da visualização. Os controladores funcionam em conjunto com as *views*, fornecendo funções essenciais para os eventos, geridos a partir do código JavaScript. Os eventos JavaScript são responsáveis em criar as caixas de diálogo e gerir processos de controlo de dados em segundo plano.

A pasta *public_html* possui o conteúdo disponível publicamente, incluindo o componente de *upload* de ficheiros e outros ficheiros JavaScript e CSS. Cada um desses ficheiros tem uma funcionalidade específica e estão separados em pastas por questões de organização da programação.

Algumas rotas foram criadas para serem utilizadas entre páginas e em eventos de JavaScript, nomeadamente em operações existentes no módulo.

Desenho

A estrutura HTML é gerida no servidor, nos ficheiros Blade do Laravel, e também gerida a partir dos ficheiros de estilos (CSS) e de JavaScript. Os estilos estão organizados em distintos ficheiros, para haver mais organização e carregar o essencial à medida que as páginas são utilizadas. O mesmo acontece na programação do código JavaScript.

Na pasta *views*, o ficheiro *files-gallery.php* é responsável em mostrar as páginas dos ficheiros existentes na pasta selecionada e também a página com os resultados da pesquisa. A Figura 45 mostra a implementação da hierarquia das subpastas existentes.

```
(...)
<div class="sidebar__list">
    <div class="sidebar__folder_sub">
        <i class="fas {{
              isset($sidebar_state['root']) && $sidebar_state['root'] === 'hidden'
              ? 'fa-chevron-right'
              : 'fa-chevron-down'
            }}"></i>
        <i class="fas fa-folder"></i>
        @php $params = ['path' => 'root']; @endphp
        <a href="{{ $url }}&{{ http_build_query($params) }}" title="Root">Root</a>
     </div>
     {{ buildNestedList($all_directories, $sidebar_state, $url) }}
</div>
<hr>
<div class="sidebar__list">
    <div class="sidebar__folder_sub">
        <i></i>
        <i class="fas fa-trash"></i>
        @php $params = ['path' => 'recycle-bin']; @endphp
        <a href="{{ $url }}&{{ http_build_query($params) }}" title="Recycle Bin">
          Recycle Bin</a>
    </div>
</div>
(...)
```

Figura 45 - Implementação da barra lateral com a hierarquia de pastas.

O ficheiro *view-file.php* apenas aplica na gestão de apresentação dos ficheiros compatíveis no navegador Web. A Figura 46 apresenta o código utilizado para adaptar a região do visualizador de acordo com o tipo de ficheiro. Todos os ficheiros de texto que não estejam codificados podem ser observados no visualizador, evitando a sua transferência.

```
(...)
<div class="file-viewer">
    @php (...) @endphp
    @switch($viewer_type)
        @case('image')
            <img src="{{ $url }}" alt="{{ $filename }}">
        @break
        (...)
        @case('pdf')
            <iframe src="{{ $url }}" title="{{ $filename }}"></iframe>
        @break
        (...)
        @default
            <p>File type not supported</p>
        @break
    @endswitch
</div>
(...)
```

Figura 46 - Implementação da zona de visualização dos ficheiros.

O resultado visual do módulo pode ser visível na Figura 47, onde está representada a página de arranque do Files Gallery.

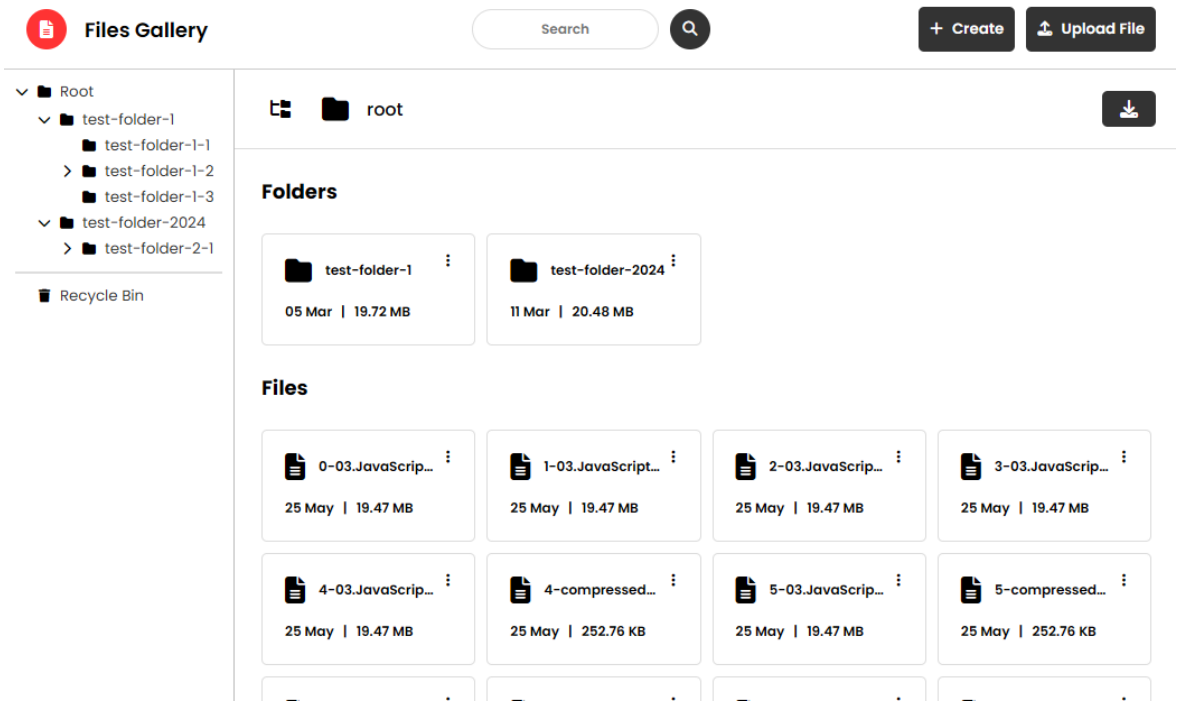


Figura 47 - Resultado da interface visual da página inicial do módulo.

Processo de interação entre páginas

Cada acesso a uma página do explorador contém um conjunto específico de parâmetros no URL. Esses parâmetros, conhecidos como *query strings*, estão relacionados com o nome

dos ficheiros e das pastas associadas. Com esses valores existentes no URL, um conteúdo específico de um ficheiro Blade é carregado.

A Figura 48 mostra os caminhos de URL necessários para aceder a uma página do módulo. Observando a figura, existem quatro exemplos de caminhos utilizados. Nos três primeiros, utilizou-se o mesmo caminho ("*/files-gallery*"), porque a rota associada está apontada para o ficheiro Blade *files-gallery.php*. No último exemplo, o caminho é diferente, porque a rota direciona para o ficheiro *view-file.php*.

```
(1) /files-gallery

(2) /files-gallery?
        path=/test-folder-1/test-folder-1-1

(3) /files-gallery?
        search=search_text

(4) /files-gallery/view-file?
        filename=new-pdf-name-2.pdf&
        location=/test-folder-1
```

Figura 48 - Os caminhos e parâmetros utilizados para conseguir aceder a conteúdo específico de uma página Web.

Nos caminhos URL, as *query strings* são imprescindíveis para o acesso a pastas e ficheiros específicos. O primeiro exemplo acede à página inicial, onde é apresentado o conteúdo existente na pasta raiz, não necessitando de utilizar a *query* 'path'. No caso da pesquisa (terceiro exemplo), os resultados são sempre visíveis de acordo com a expressão a pesquisar, especificada no parâmetro 'search'. O último exemplo, que representa o caminho para a visualização do conteúdo do ficheiro, obriga a especificar a localização da pasta com o parâmetro 'location' e o nome do ficheiro selecionado com o parâmetro 'filename'.

Processo de controlo de erros e configurações

Para um controlo mais rigoroso dos erros que possam originar durante a utilização do File Gallery, as funções utilizadas para a gestão dos ficheiros possuem vários códigos e mensagens de erro.

Na pasta *configs*, o ficheiro *File.php* contém as funções genéricas que envolvem ficheiros, como a edição de nome ou a eliminação, funcionando em conjunto com o ficheiro enumerador *FileStatusCodes.php*. A Figura 49 contém um exemplo de um método da classe *File*, criado para mover ficheiros entre pastas.

```
(...)
class Files {
    (...)
    public static function moveFile($old_path, $new_path)
    {
        (...)
        /* some validations */
        (...)

        if (rename($old_path, $new_path) === false) {
            return [
                'status' => 'error',
                'code' => ErrCodes::MOVE_FILE_ERROR_FAILED_TO_MOVE,
                'message' => trans('file_messages.MOVE_FILE_ERROR_FAILED_TO_MOVE')
            ]; // Error code 3: Failed to move file
        }

        return [
            'status' => 'success',
            'code' => ErrCodes::MOVE_FILE_SUCCESS,
            'message' => trans('file_messages.MOVE_FILE_SUCCESS')
        ]; // Success
    }
    (...)
}
```

Figura 49 - Uma das funções genéricas associadas aos ficheiros.

O enumerador *FileStatusCodes.php* é o responsável por indicar os códigos de cada processo, seja de sucesso ou de erro. A Figura 50 descreve os códigos associados às suas chaves, utilizadas em conjunto com o sistema de tradução das mensagens, para que o módulo aceite todos os idiomas implementados na plataforma.

```
(...)
enum FileStatusCodes: int {
    // MOVE DIRECTORY FUNCTION (CODES 1XX)
    case MOVE_DIRECTORY_ERROR_SOURCE_NOT_DIRECTORY = 101;
    case MOVE_DIRECTORY_ERROR_DESTINATION_EXISTS_NOT_DIRECTORY = 102;
    case MOVE_DIRECTORY_ERROR_FAILED_TO_RENAME = 103;
    case MOVE_DIRECTORY_ERROR_PARENT_DIRECTORY_CREATION_FAILED = 104;
    case MOVE_DIRECTORY_SUCCESS = 100;
    (...)
}
(...)
```

Figura 50 - O enumerador com os códigos associados a erros.

A Figura 51 mostra as traduções aplicadas às mensagens de estado dos processos executados.

```php
<?php

return [
    'MOVE_DIRECTORY_ERROR_SOURCE_NOT_DIRECTORY'
        => 'A pasta de origem não existe.',
    'MOVE_DIRECTORY_ERROR_DESTINATION_EXISTS_NOT_DIRECTORY'
        => 'A pasta de destino já existe e não é uma pasta.',
    'MOVE_DIRECTORY_ERROR_FAILED_TO_RENAME'
        => 'Falha ao renomear a pasta de origem.',
    'MOVE_DIRECTORY_ERROR_PARENT_DIRECTORY_CREATION_FAILED'
        => 'Falha ao criar a pasta pai da pasta de destino.',
    'MOVE_DIRECTORY_SUCCESS'
        => 'A pasta foi movida com sucesso.',
    (...)
]
```

Figura 51 As traduções implementadas para as mensagens.

O controlador *FilesController.php* trabalha em conjunto com as *views* e com as funções de *File.php*, estando responsável em mostrar ao utilizador da plataforma as mensagens de estado. A Figura 52 apresenta um método que permite mover pastas, podendo ser usado em código JavaScript através de rotas.

```php
(...)
class FilesController {
    (...)
    public static function moveDirectory(Request $request)
    {
        $request_body = $request->all();
        $req_location = $request_body['location'];
        $req_old_location = $request_body['old-location'];
        $req_old_directory = $request_body['old-directory'];

        (...)

        $status = File::moveDirectory($old_path, $new_path);

        return redirect()->back()
            ->with('status', $status['status'])
            ->with('message', $status['message'])
            ->with('code', $status['code']);
    }
    (...)
}
```

Figura 52 - A função do controlador que permite mover pastas a partir de JavaScript.

7.2.3. Integração

A implementação do gestor de ficheiros foi realizada numa estrutura um pouco diferente da do EasyWeb, obrigando a ajustes aos nomes dos ficheiros para seguir os padrões do EasyWeb.

O CMS contém os componentes e os módulos organizados nas pastas das *views* do Laravel. Os módulos ficam dentro da pasta *EWApps*, enquanto os componentes estão guardados em *EWElements*. Portanto, a interface visual foi configurada dentro de uma pasta específica de *EWApps*.

A Figura 53 identifica todos os ficheiros que estiveram envolvidos na implementação do módulo, em que os que estão assinalados com ● são os que estão envolvidos na integração.

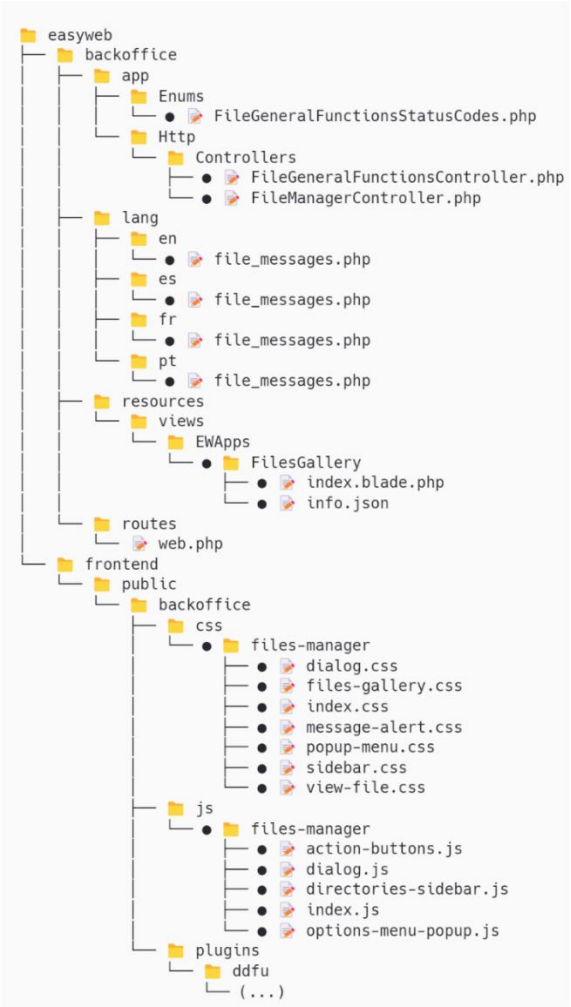


Figura 53 - Os ficheiros (e respetivas localizações) envolvidos na integração do novo módulo.

Os ficheiros públicos da plataforma EasyWeb estão localizados no lado do *front-end*, evidenciando que os ficheiros JavaScript e CSS do módulo e do componente estão armazenados na pasta *public* existente.

A configuração da parte visual do módulo é implementada nos ficheiros existentes na pasta *FilesGallery*. O conjunto de páginas que compõe o módulo estão preparadas, para além do ficheiro JSON, em apenas um ficheiro de *views* Blade. Como a implementação contém dois ficheiros Blade em vez de apenas um, é necessário aplicar alterações e, por conseguinte, integrar a parte do visualizador no ficheiro que lista os ficheiros e pastas.

Devido à forma como os módulos, numa ideia geral, estão construídos e são executados, a melhor forma de transitar entre páginas é utilizar *query strings* no URL, em vez do uso de caminhos. Desta maneira, quando a rota que acede ao módulo chamar o conteúdo HTML, cada conteúdo é carregado conforme o valor dos outros parâmetros, como a Figura 54 apresenta.

```
(1) /backoffice/runewapp?
        runid=FilesGallery

(2) /backoffice/runewapp?
        runid=FilesGallery&
        path=%2Fpdfs

(3) /backoffice/runewapp?
        runid=FilesGallery&
        search=search_text

(4) /backoffice/runewapp?
        runid=FilesGallery&
        pagename=view-file&
        filename=new-pdf-name-2.pdf&
        location=%2Ftest-folder-1
```

Figura 54 - As *query strings* utilizadas para diferentes páginas do módulo.

Observando a Figura 54, o parâmetro *runid* indica o ID do módulo a mostrar e, como está clarificado na figura, deve estar sempre representado cada vez que se pretende aceder ao gestor de ficheiros. O primeiro *endpoint* encaminha para a página inicial, mostrando o conteúdo da pasta raiz. O segundo mostra uma página com os ficheiros e/ou pastas existentes no diretório '/pdfs'. O terceiro representa uma pesquisa por pastas ou ficheiros que tenham

aquela expressão no seu nome. Por fim, o seguinte *endpoint* apresenta a página de visualização de ficheiros, apenas visível com o parâmetro *pagename* e com a indicação do nome do ficheiro (*filename*) e do caminho relativo da sua pasta (*location*).

7.3. Componente de gestão de tabelas

Neste subcapítulo, irá ser descrito o componente de gestão de tabelas desenvolvido, mostrando a prototipagem, a implementação e a utilização. O componente também é designado por Drag-Drop-Table-Rows-Sort.

7.3.1. Prototipagem

O componente tem a missão de gerir as tabelas, focando principalmente na posição que as linhas ocupam. Desta forma, pretende-se aplicar uma coluna que facilite o processo de ordenação através de arrasto. A Figura 55 mostra o processo de arrasto da linha a partir da coluna de ordenação, que se encontra à direita. Esta coluna distingue-se das outras por estar a representar um ícone propício ao arrasto.

Figura 55 - Processo de arrasto de linhas numa tabela.

O arrasto é simples e suficiente para uma melhor interação nas tabelas. No entanto, gere uma questão quanto à ordenação manual, a partir da edição do conteúdo. Antigamente, o processo era complexo e oferecia uma experiência de utilizador menos agradável. Então, para conseguir minimizar o problema, a tabela deixaria de estar paginada e, por consequência, pensou-se em apresentar, na página dos detalhes, a posição da linha na tabela e o valor da ordem, como está na Figura 56.

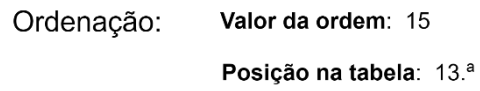


Figura 56 - *Mockup* da ordenação manual entre páginas, no editor de conteúdo.

De acordo com a Figura 56, a lista é calculada observando as configurações da tabela. O valor da ordem, calculado depois do arrasto da linha, indica a posição na tabela. Quanto maior for o valor da ordem, menor é o valor da posição.

7.3.2. Implementação

A ordenação através de arrasto tem de ser feita sem a necessidade de interagir diretamente com o servidor, através da atualização das páginas Web. No entanto, é importante haver uma interação com a base de dados, para registar a nova posição da linha em tempo real. Por isso, os navegadores Web disponibilizam uma maneira de interagir com a página Web no lado do cliente, utilizando JavaScript como linguagem de programação. Ao executar um *script*, vários eventos podem ser ativados de acordo com as interações.

Em concreto no que se refere à ordenação, os eventos são executados quando inicia arrasto, quando é arrastado e quando é largado. No final do processo, a base de dados precisa de saber o novo valor e, portanto, um pedido HTTP é iniciado. Após receber a resposta, é apresentada a mensagem de *feedback*.

Além disso, para que se mantenha o visual das células da coluna em qualquer tabela e que os seletores CSS estejam em consistência com a programação do JavaScript, foi desenvolvido um ficheiro de estilos CSS.

A organização dos ficheiros JS e CSS está representada na Figura 57, onde também se destaca um ficheiro Markdown, com intenções semelhantes ao componente de carregamento de ficheiros. O carater ● representa os novos ficheiros que o compõem.

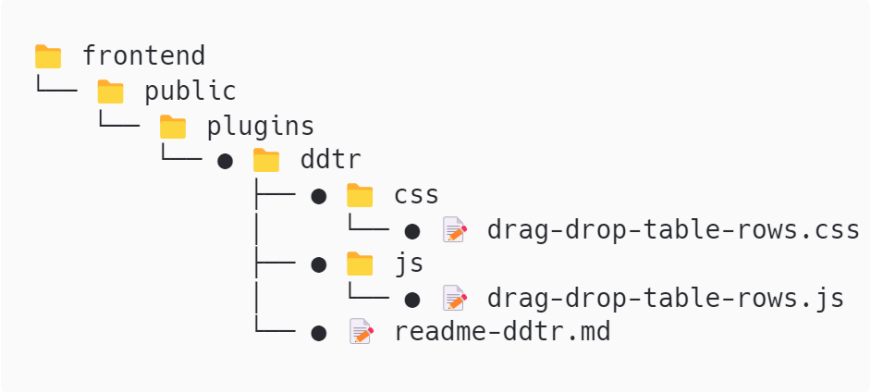


Figura 57 - Organização dos ficheiros associados ao componente.

As rotas utilizadas para guardar o valor da posição são geridas a partir do sistema do CMS, seguindo os métodos fornecidos pelo Laravel, como acontece no componente acima.

7.3.3. Integração

Sabendo a organização dos ficheiros do componente, apresentados na Figura 57, a aplicação é feita nas pastas públicas do CMS, como acontece em qualquer biblioteca JavaScript importada.

Pretende-se aplicar o componente nas tabelas que listam os registos de cada módulo, principalmente naquelas que fornecem a capacidade em ordenar. Antes da aplicação da nova ferramenta de ordenação, as linhas eram ordenadas a partir de cliques de botão que incrementavam ou decrementavam o valor da ordem. A célula responsável possuía os dois botões e um campo de texto com o número, que poderia ser utilizado na ordenação manual, durante o processo de edição. A Figura 58 apresenta, na coluna "Ordem", o formato das células de ordenação, em que os dois botões com cor amarela realizam a incrementação ou decrementação do valor.

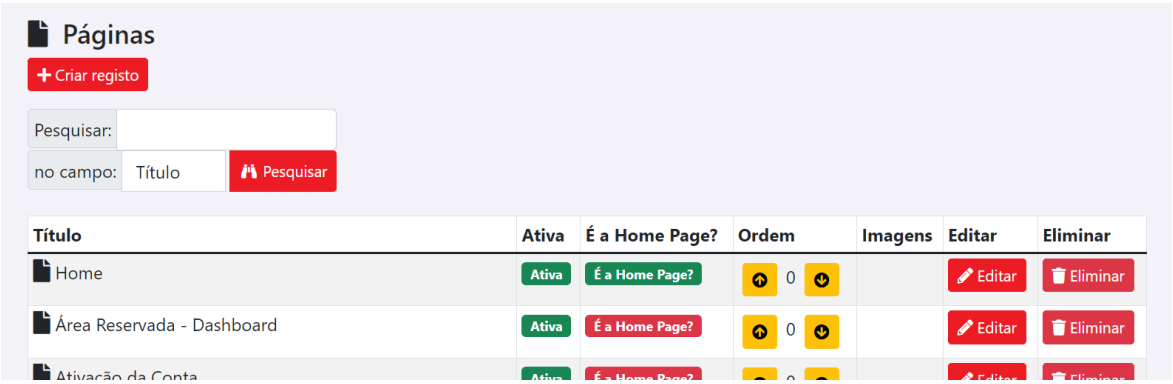


Figura 58 - Antiga representação da ordenação das tabelas no módulo das páginas.

A Figura 59 exemplifica a forma manual da aplicação do valor da ordem, num campo de número, durante o processo de edição do conteúdo adicionado no CMS.

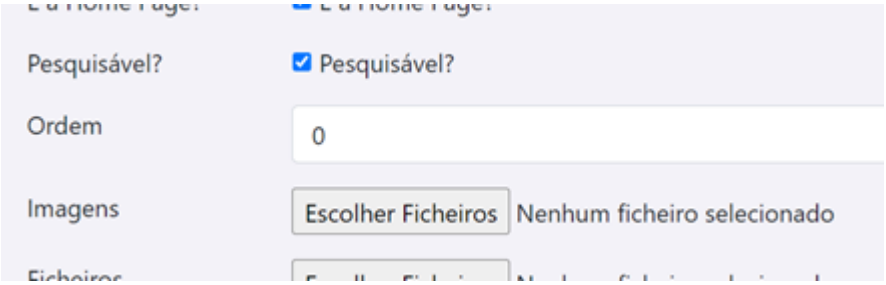


Figura 59 - Processo manual de inserção do valor da ordem, durante a edição do registo.

Para aplicar a nova coluna com células de arrasto, é necessário compreender como são geradas as páginas com as tabelas, porque, no CMS, todas são baseadas num único ficheiro *view* e geridas dinamicamente através de *scripts* PHP. Este ficheiro Blade está seccionado em várias partes, que podem representar as regiões do cabeçalho ou da importação de CSS e JavaScript, sendo estes essenciais para aplicar o *script* JavaScript e os estilos para aquelas células.

Depois de importar a biblioteca, as tabelas ainda não possuem a nova coluna e, portanto, é importante analisar como os *scripts* PHP as povoam. Cada módulo é considerado um "item" do CMS e estão guardados numa pasta específica das *views* Blade. Ao analisar cada um dos ficheiros existentes em cada "item", o sistema está preparado com um conjunto de configurações para cada campo na fase de edição. A Figura 60 mostra as definições aplicadas a um campo de texto "Título", que determina as funções a executar para a situação de listagem (tabela), na criação ou na edição, entre outros. No caso da ordem, Figura 61, as chaves *list_function*, *insert_function* e *edit_function* são as que estão mais envolvidas no processo de aplicação da nova coluna de arrasto.

```
'title' => array(
    'datatype' => 'string',
    'include_on_create' => 1,
    'include_on_modify' => 1,
    'multilanguage' => 1,
    'name' => array(
        'en' => 'Name',
        'pt' => 'Nome',
        'es' => 'Nombre',
    ),
    'list_function' => array(
        'function' => 'multilang_list_name_display',
        'parameters' => array(),
    ),
    'on_create_function' => array (),
    'after_create_function' => array (),
    'on_modify_function' => array(),
    'after_modify_function' => array(),
    'insert_function' => array(
        'function' => 'standard_textfield_input',
        'parameters' => array(
            'required' => 1,
            'multilanguage' => 1,
        ),
    ),
    'edit_function' => array(
        'function' => 'standard_textfield_input',
        'parameters' => array(
            'required' => 1,
            'multilanguage' => 1,
        ),
    ),
),
```

Figura 60 - A estrutura das variáveis que compõem os formulários de criação ou de edição.

```
'ordem' => array(
    'datatype' => 'integer',
    'include_on_create' => 1,
    'include_on_modify' => 1,
    'multilanguage' => 0,
    'name' => array(
        'en' => 'Order',
        'pt' => 'Ordem',
        'es' => 'Orden',
    ),
    // 'list_function' => array(
    //  'function' => 'standard_ew_list_ordem_display',
    //  'parameters' => array(),
    //  ),
    'list_function' => array(),
    'on_create_function' => array (),
    'after_create_function' => array (),
    'on_modify_function' => array(),
    'after_modify_function' => array(),
    'insert_function' => array(
        'function' => 'standard_ew_order_input',
        'parameters' => array(
            'multilanguage' => 0,
            'field_value' => 0,
        ),
    ),
    'edit_function' => array(
        'function' => 'standard_ew_order_input',
        'parameters' => array(
            'multilanguage' => 0,
        ),
    ),
),
```

Figura 61 - A estrutura da variável 'ordem', modificada para a nova coluna.

Tendo conhecimento de que o processo de ordenação é feito em segundo plano, através de JavaScript, uma nova rota foi instaurada no sistema de rotas do EasyWeb, conforme está visível na Figura 62.

```
Route::middleware('auth')->group(function () {
    // drag drop table sort
    Route::post(
      'drag-drop-sort-table',
      'App\Http\Controllers\TableManagementController@dragDropTableSort'
    )->name('drag-drop-sort-table');

    (...)
});
```

Figura 62 - Rota criada para que o JavaScript envie dados para o servidor.

De acordo com a Figura 62, a aplicação do valor da ordem é feita numa função de um controlador – *TableManagementController*. A função associada na rota mostra que o valor de ordenação é obtido consoante o valor de uma constante, o valor da quantidade total de registos e a lista da ordem dos registos (vinda da biblioteca), como está descrito na Figura 63.

```php
class TableManagementController extends Controller
{
    public function dragDropTableSort(Request $request)
    {
        if ($request->orderarray) {
            $order_array = $request->orderarray;

            $model_name_tmp = '\\App\\Models\\' . $request->table_model;

            $constant = 2; // valor constante para o cálculo da ordem
            $order_max_value = count($order_array) * $constant;
            $order_min_value = 0;

            foreach ($order_array as $key => $value) {
                $model = $model_name_tmp::findOrFail($value);
                $model->ordem = ($constant * $key);
                $model->save();
            }

            return [
                "status" => "success",
                "message" => "Table order updated successfully",
                "order_min_value" => $order_min_value,
                "order_max_value" => $order_max_value,
            ];
        }

        return [
            "status" => "error",
            "message" => "Table order was not updated"
        ];
    }
}
```

Figura 63 - Programação da atribuição de valor de ordenação de cada linha.

Depois de resolver todos os procedimentos necessários, os resultados estão à vista em, pelo menos, num módulo, como se pode ver na Figura 64. O objetivo é implementar em todos os módulos que contêm tabelas de registos.

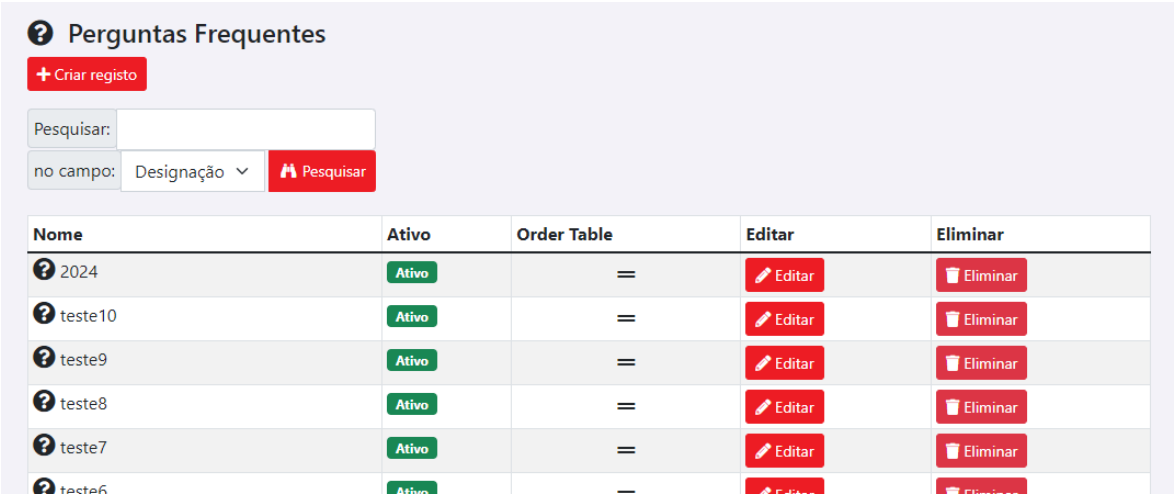


Figura 64 - Resultado da implementação da nova coluna de ordenação na tabela do módulo 'FAQs'.

Na criação de um registo ou na edição, o resultado encontra-se na secção da ordem, como está apresentado na Figura 65.

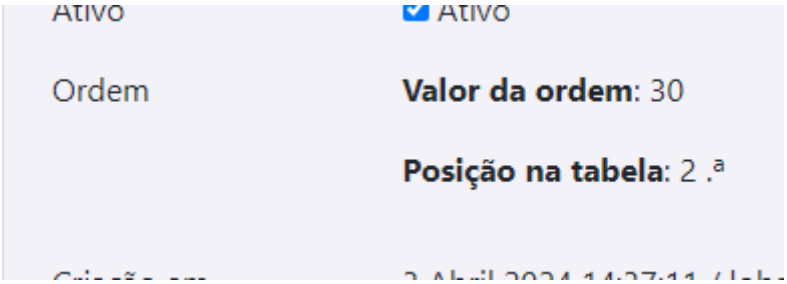


Figura 65 - Resultado da implementação dos valores na edição do registo.

7.3.4. Utilização do componente

O código JavaScript, que define todo o funcionamento do componente, é constituído por uma função de inicialização, por um conjunto de configurações e pelas funções criadas, com o objetivo de ter um maior controlo da programação do gestor de ordenação. Este desenvolvimento teve, como base, o artigo de Phuoc Nguyen [57], que possibilitou a alavancagem deste componente. Algumas adaptações foram implementadas, para que uma grande parte da programação fosse integrada com as tabelas do EasyWeb.

Para que fique aplicado a funcionalidade em cada linha de uma tabela, é necessário implementar, em HTML, a célula que permite o arrasto da linha.

Para mais informações sobre o componente e sua utilização, aconselha-se visualizar o Anexo D – Manual de Utilizador do Componente de Gestão de Tabelas.

7.4. Componente de construção de páginas Web

Este subcapítulo visa mostrar a implementação, a integração e a utilização do componente de construção de páginas Web no CMS, utilizando a biblioteca Laravel-GrapesJs.

7.4.1. Implementação

A implementação do pacote Laravel no projeto requer a compreensão do funcionamento, através da leitura de documentos fornecidos, como também a adaptação para situações mais específicas. Este subcapítulo está dividido em duas partes, onde, inicialmente, se apresenta o procedimento da inserção da biblioteca no projeto e, depois, o estudo e alterações necessárias para ultrapassar diversas limitações.

Instalação

A utilização da biblioteca requer a criação de um projeto Laravel específico, que seja um CMS genérico com capacidade em gerir páginas, não necessitando de outros módulos.

Depois de criado o projeto Laravel, é necessário realizar um conjunto de comandos PHP na linha de comandos, para instalar a biblioteca. Todos a informação utilizada encontra-se na documentação do repositório GitHub [47].

A Figura 66 mostra uma parte da documentação, inclusive alguns comandos PHP indicados para usar durante a inserção da biblioteca no projeto.

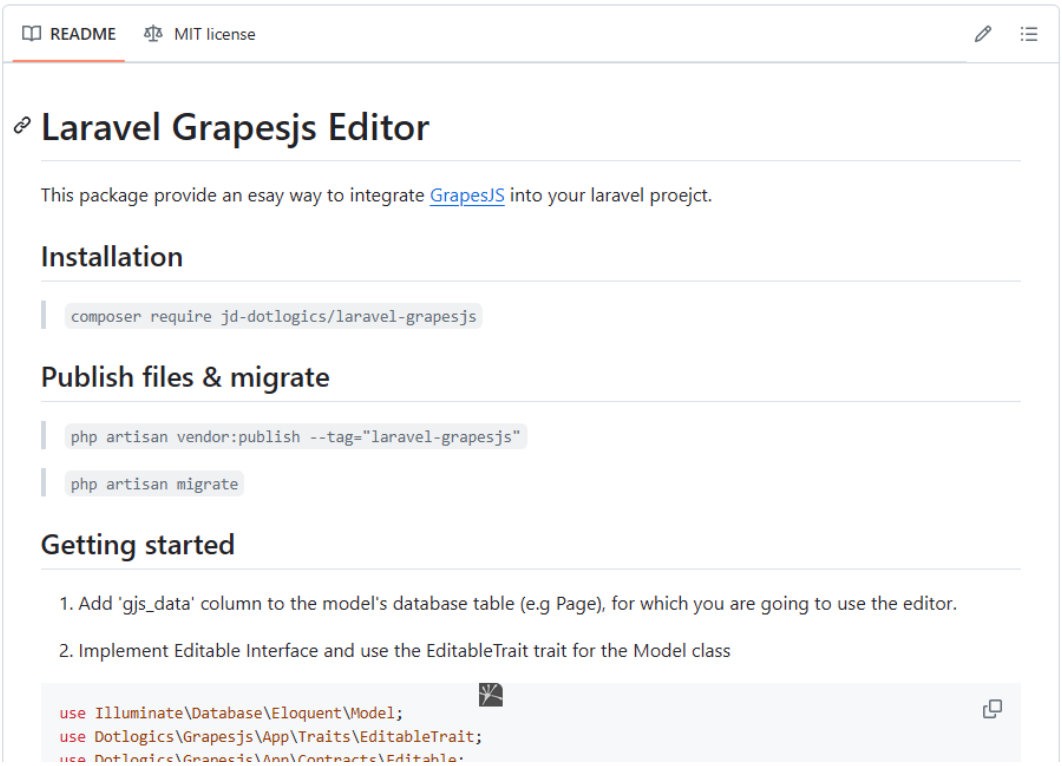


Figura 66 - Alguns procedimentos realizados para a instalação da biblioteca.

Durante o processo de integração do pacote ao projeto, o controlador das páginas fica preparado com um editor para cada página, acedido através de uma rota. Além disso, outra rota fica disponível para conseguir visualizar o conteúdo da página sem a necessidade de aceder ao editor. A Figura 67 mostra as rotas criadas que interagem com o pacote Laravel.

```
Route::prefix("/backoffice")->middleware("auth")->group(function () {

    (...)

    // pages
    Route::resource("pages", "PagesController");

    Route::get('/pages/{page}/editor', 'PagesController@editor')
        ->name('pages.editor');
    Route::get('/pages/{page}/show-page', 'PagesController@showPage')
        ->name('pages.showPage');

    (...)
});
```

Figura 67 - Lista de rotas implementadas para a gestão de página no projeto.

A biblioteca também fornece um conjunto de métodos e variáveis que permitem aceder ao HTML e CSS da página sem dificuldades, sendo possível observar na documentação.

No ficheiro Blade associado à visualização do conteúdo da página, está implementado o código da Figura 68. Nele, `$page->css` e `$page->html` são duas propriedades do módulo *Page* que são fornecidas diretamente da biblioteca, enquanto as outras provêm de desenvolvimento do CMS.

```
<html lang="pt">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Page Preview: {{ $page->name }}</title>

    (...) alguns links de CSS gerais (...)

    @if (isset($page->gjs_data['links']))
        @foreach (json_decode($page->gjs_data['links']) as $link)
            <link rel="{{ $link->rel }}" href="{{ $link->href }}">
        @endforeach
    @endif

</head>

<style type="text/css">
    {!! $page->css !!}
</style>

{!! $page->html !!}

@if (isset($page->gjs_data['scripts']))
    @foreach (json_decode($page->gjs_data['scripts']) as $link)
        <script type="{{ $link->type }}" src="{{ $link->src }}"></script>
    @endforeach
@endif

(...) alguns scripts JavaScript gerais (...)

</html>
```

Figura 68 - Estrutura Blade de Laravel utilizada para pré-visualizar páginas geridas pelo editor.

Finalizando a instalação, o pacote cria um conjunto de pastas na pasta Views do projeto, onde são representados os ficheiros Blade associados a cada novo bloco ou *template* criado. O editor tem uma secção, nos componentes, que disponibiliza o conteúdo de cada um desses ficheiros.

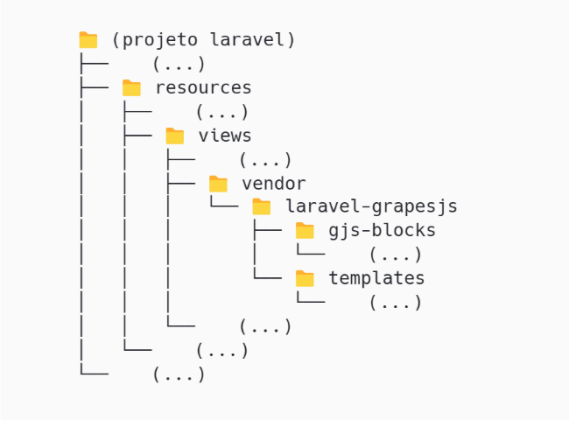


Figura 69 - Hierarquia de pastas que guardam os novos componentes/blocos ou *templates*.

Adaptação

O sistema de componentes de páginas que o EasyWeb contém apresenta diversas limitações quanto à praticidade e evolução relativamente a novas tendências tecnológicas. E o pacote Laravel-GrapesJS vem para minimizar os problemas, mas também para cumprir os seguintes objetivos:

- Conseguir integrar com todos os módulos existentes no CMS;
- Facilitar na criação de novos componentes;
- Facilitar integrações de *scripts* externos para cada componente;
- Manter a gestão do componente o mais próxima possível do sistema existente.

É evidente que todos os pontos acima são muito específicos, devido à implementação do EasyWeb. Por isso, a biblioteca precisa de algumas modificações.

Começando pelo editor, é composto por diversas secções, como se observa na Figura 70. A lateral direita está focada para os componentes, enquanto a de topo contém botões de ação relacionados com o editor. A área a branco é onde são adicionados os componentes da página, sendo conhecida por *canvas*. As imagens numeradas mostram o conteúdo do separador quando selecionado na aba acima.

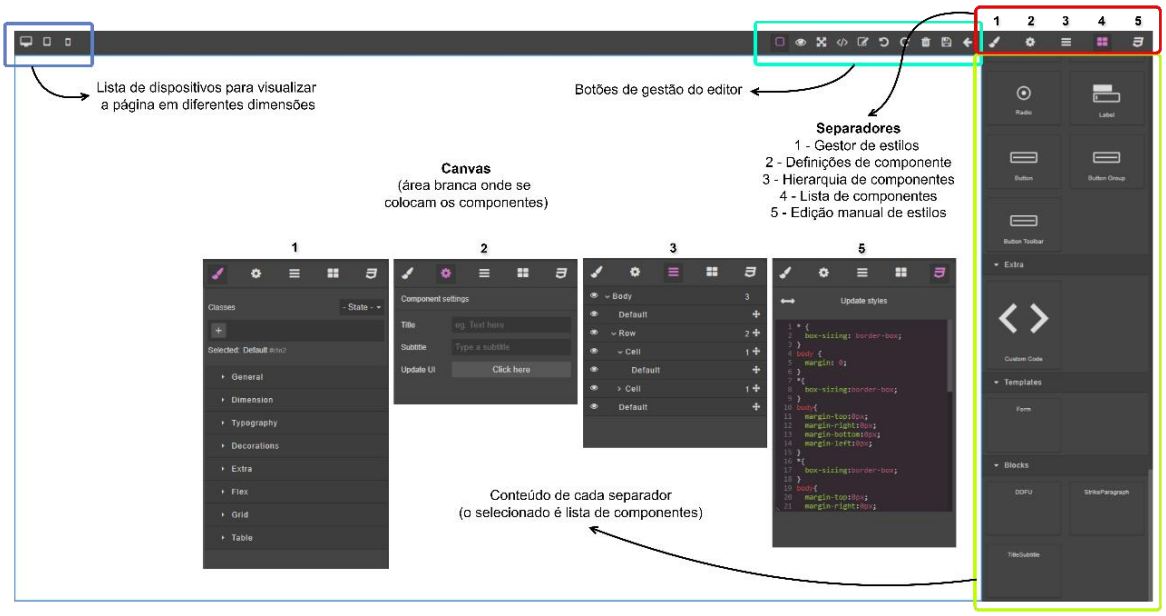


Figura 70 - Composição do editor GrapesJS.

A secção lateral contém diversos separadores que podem ser úteis para a criação de componentes com configurações. O GrapesJS oferece um separador focado para as definições do componente selecionado no *canvas*, caracterizado com número 2 (Figura 70) e conhecido por *traits*. Segundo a documentação [58], os *traits* definem os comportamentos e parâmetros específicos do componente.

No final da lista de componentes, existem dois grupos (*Blocks* e *Templates*) que representam os componentes criados, cuja organização depende da pasta onde estão inseridos os ficheiros Blade.

A observação do editor, estudando as funcionalidades existentes, permitiu definir alguns caminhos para o desenvolvimento de componentes personalizados, aproveitando o que já se encontra ativo.

Com o estudo do funcionamento do pacote e do editor, percebe-se que existem algumas limitações quanto ao desenvolvimento dos componentes de página e dos *templates*. Como o editor funciona somente em JavaScript, é importante haver um mecanismo que aceite ler ficheiros Blade no editor.

O sistema de leitura dos ficheiros Blade encontra-se implementado nativamente no pacote, mas não está adaptado para funcionar com controladores. Os controladores, neste caso, são a peça chave dos componentes, porque permitem criar as suas configurações, não interferindo com outros componentes. Além disso, facilitam na renderização do ficheiro Blade associado e na escalabilidade, caso se pretende criar outros desenvolvimentos nesta área.

Para conseguir associar um controlador a um ficheiro Blade, o Laravel está preparado para ter algo semelhante a componentes. Com um simples comando Artisan [59], um componente pode ser adicionado ao projeto, criando automaticamente um ficheiro Blade e um controlador. Assim, a parte da construção do componente está tratada.

No entanto, a pasta que recebe os ficheiros Blade é diferente da pasta utilizada pelo pacote. Portanto, várias alterações ao código são necessárias para começar a ler os ficheiros da pasta adequada. A Figura 71 mostra as novas pastas criadas para albergar os ficheiros Blade dos componentes, sendo necessário gerar outras para se assemelhar ao que já existe.

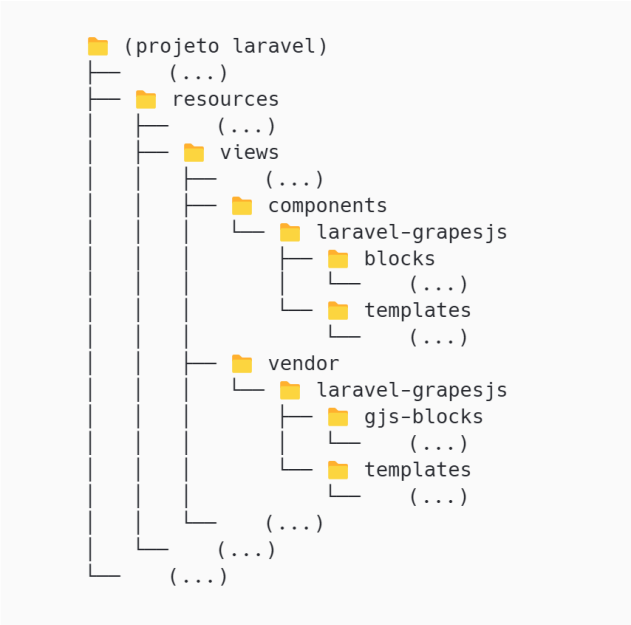


Figura 71 – Hierarquia de pastas após a adição de componentes Laravel no projeto.

Os controladores dos componentes não necessitam de ser movidos para outras pastas, porque a programação do pacote não vai interagir diretamente nos ficheiros. Desta forma, ficam na pasta que foi criada automaticamente durante a execução do comando Artisan.

Estando compreendida a montagem de um componente Laravel no editor, a programação do pacote sofreu algumas alterações, principalmente na leitura e nas configurações genéricas dos componentes.

Depois da montagem dos componentes no editor (aparecimento na lista), é preciso montar os *traits* de cada um, se necessário. Essa configuração é feita no controlador, cuja estrutura segue a documentação do editor.

Devido a limitações que o pacote de Laravel fornecia quanto a personalizações, o processo de aplicação de *traits* para cada componente necessita de *scripts* de JavaScript e de mais programação no servidor, que permitissem editar ou criar depois de inicializar o editor. Além disso, os dados preenchidos nos *traits* não conseguiam ser guardados a partir das configurações predefinidas do pacote, sendo necessário aplicar soluções alternativas.

Cada componente tem os seus *traits*, mas existe um botão que está sempre disponível em todos os que são personalizados. A Figura 72 mostra esse botão, cuja função é aplicar a atualização da parte visual do componente, sendo visível no *canvas*.

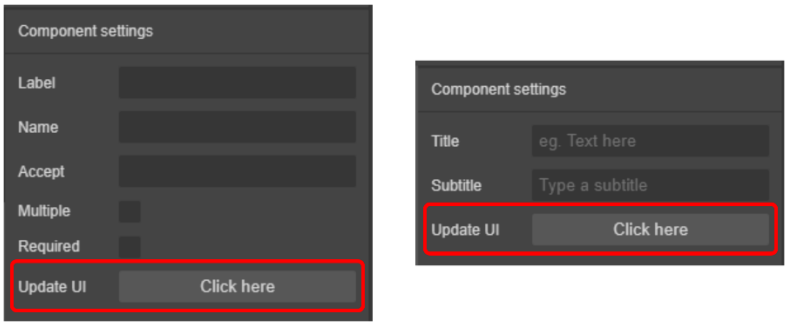


Figura 72 - Todos os componentes criados têm um botão que permite atualizar o componente.

O botão criado está associado a uma funcionalidade que o GrapesJS disponibiliza – os comandos [60]. Com os comandos, a configuração de eventos de clique nos botões dos *traits* torna-se mais fácil e podem ser criados depois de executar o editor. Para este caso, o comando corresponde a um método que acede o controlador do componente e, de seguida, devolve o conteúdo HTML do ficheiro Blade atualizado.

O editor também fornece um conjunto de eventos que podem ser úteis em casos onde se pretende aplicar algumas modificações nos componentes. Um desses eventos é *component:add* [61], que é chamado à medida que se adicione um componente no *canvas*. Para este caso, é um momento indicado para atribuir os *traits* ao tal componente adicionado. Outro evento utilizado é *storage:after* [62], cuja função é executar algo depois de guardar as alterações feitas na página. Pode ser usado para guardar, na base de dados, as alterações feitas nos *traits* dos componentes da página e os *scripts* externos usados.

As modificações referidas estão aplicadas num ficheiro JavaScript, que é sempre executado à medida que o editor é iniciado.

7.4.2. Integração

Para integrar a biblioteca no EasyWeb, o processo foi muito semelhante ao referido anteriormente, mas foi necessário adaptar alguns pontos, devido à sua unicidade. Houve a necessidade de:

- criar controladores, para manter as funcionalidades dos controladores usados na implementação;
- definir uma combinação temporária entre o conteúdo das páginas e o idioma do *website*, que envolvesse o editor e a base de dados. Neste momento, funciona apenas com o português.

Depois de integrar o editor de páginas no CMS, o acesso ao editor era conseguido através de rotas, cujo processo não é adequado para os utilizadores que recorrem diariamente esta funcionalidade. Então, para que cada página tenha acesso direto ao seu editor, foi implementado um botão no formulário de edição do registo da página. O botão somente está disponível durante a edição da página, porque necessita do ID da página para carregar o editor.

A função do botão é reencaminhar para a página do editor, sem precisar de digitar o URL na barra de pesquisa do navegador Web. O botão possui o título "Open GrapesJs Editor" e encontra-se por baixo do campo "Descrição" (Figura 73).

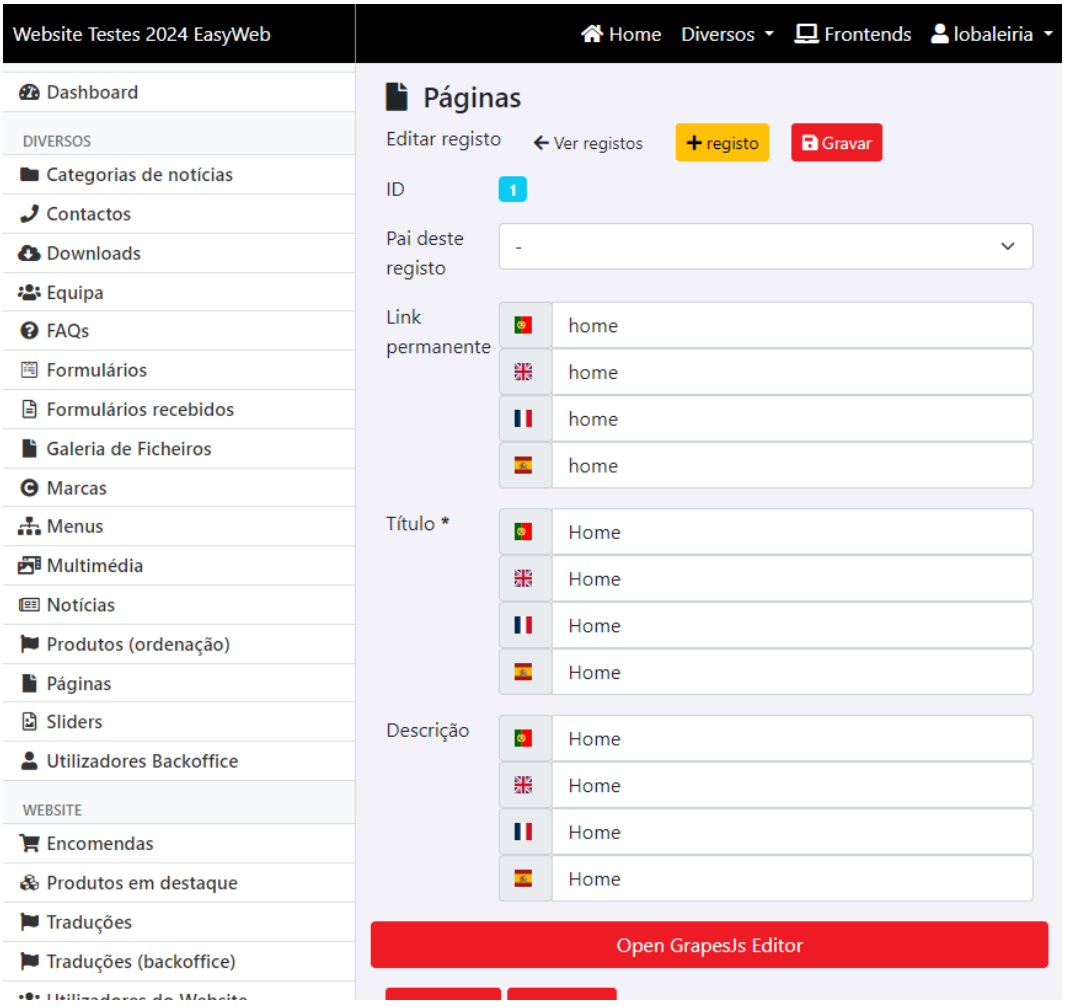


Figura 73 - Para aceder ao editor da página, basta clicar no botão que se encontra por baixo do campo 'Descrição'.

Com a existência de uma maneira mais simples para aceder à plataforma GrapesJS, a técnica utilizada na gestão dos componentes criados mantém-se, oferecendo condições mais adequadas para a comunicação com outros módulos do EasyWeb.

7.4.3. Utilização do componente

A utilização do editor de páginas Web é relativamente simples, onde apenas se torna ligeiramente mais complexo na gestão dos estilos dos componentes e na montagem na área de construção da página, como se vê na Figura 70.

No entanto, um dos pontos cruciais do editor é a adição de componentes de Laravel. Para criar os ficheiros de um componente ou de um *template*, só é necessário executar um comando na linha de comandos do projeto. A estrutura do comando é a seguinte:

```
php artisan make:componente LaravelGrapesjs/{nome do grupo}/{nome do componente}
```

Observado o comando, os nomes dos grupos e dos componentes devem seguir a nomenclatura *PascalCase*, porque representa o nome da classe do controlador do componente. O nome do grupo apenas pode ser 'Blocks' ou 'Templates', porque são os únicos grupos elegíveis no editor.

Estes são os exemplos de componentes que foram criados através do comando:

- `php artisan make:component LaravelGrapesjs/Blocks/StrikeParagraph`;
- `php artisan make:component LaravelGrapesjs/Templates/Form`.

Depois de executado, o controlador e o ficheiro Blade são criados e estão prontos para conter as configurações do componente. Começando pelo controlador, a estrutura é composta por um construtor, por uma função que agrupa as configurações e por uma função que acede o ficheiro Blade e que devolve o conteúdo HTML.

Existem um conjunto de funções que devem estar no controlador, porque são chamadas durante o arranque do editor. São as seguintes:

- `getTraits()` – contém as configurações dos *traits*;
- `getTypeName()` – indica o nome do componente. O nome é sempre igual ao nome do ficheiro Blade;
- `getLinksCssJs()` – contém os *links* externos e *scripts* utilizados no componente;
- `configs()` – contém todas as configurações do componente. Esta função é chamada nas três funções acima, para devolver apenas o conteúdo que interessa.

Os *traits* são montados a partir das configurações do GrapesJS, o que necessita de um cuidado especial e uma observação na documentação do GrapesJS [58]. A estrutura dos dados no PHP deve ser o mais semelhante possível à estrutura JSON, para tornar a integração mais simples.

Estando tratado o controlador, o ficheiro Blade apenas precisa de saber as variáveis que recebe do controlador, no processo de renderização.

Todos os detalhes sobre a implementação de novos componentes podem ser observados no Anexo E – Manual de Utilizador do Editor de páginas Web - GrapesJS.

8. Testes e Resultados

Depois de os componentes estarem desenvolvidos em ambientes locais e isolados, estes foram testados na plataforma EasyWeb e, mais tarde, foram aplicados em novos projetos reais ou em atualizações de projetos reais mais antigos.

Com a implementação em projetos reais, foi possível observar os resultados trazidos ao longo do tempo, porque ofereceram mais *feedback* sobre a utilização dos componentes.

8.1. Testes

Os componentes foram inseridos no EasyWeb num projeto interno de testes de novas funcionalidades. Este consiste num típico projeto de desenvolvimento Web que apenas inclui a utilização do *backoffice*, porque o foco é utilizar os componentes dentro do CMS, sem passar pelo *website*, com exceção do componente de construção de páginas Web, cujas páginas criadas têm de ser vistas a partir do *front-end*.

Os testes realizados não abrangiam a criação de testes unitários. Pretendia-se somente colocar em funcionamento na plataforma de testes, realizar testes de adaptação e de funcionamento e, mais tarde e se estiver totalmente funcional, preparar para ser implementado em projetos reais.

Nas seguintes figuras, encontram-se representadas os componentes aplicados inicialmente no EasyWeb. A Figura 74 mostra o componente de carregamento de ficheiros, enquanto a Figura 75 contém o módulo de gestão de ficheiros, que envolve o componente anterior. A Figura 76 apresenta como o componente de gestão de tabelas está a ser usado e, por fim, a Figura 77 dá destaque ao componente de criação de páginas Web.

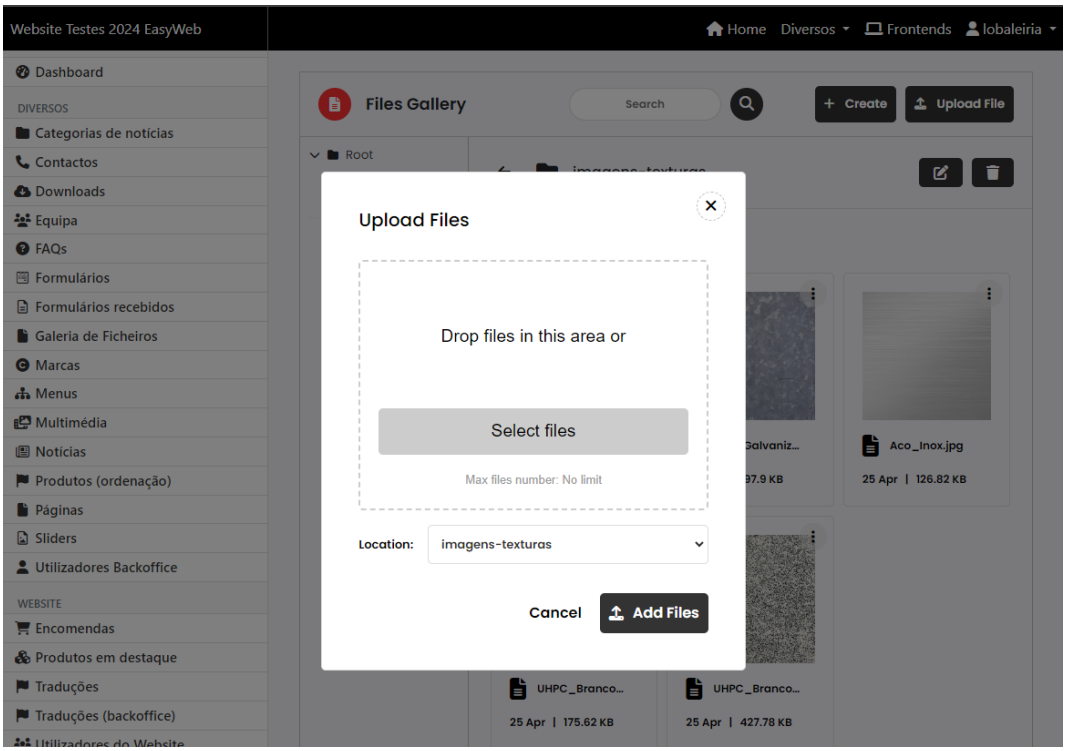


Figura 74 - Representação do componente de *upload* de ficheiros.

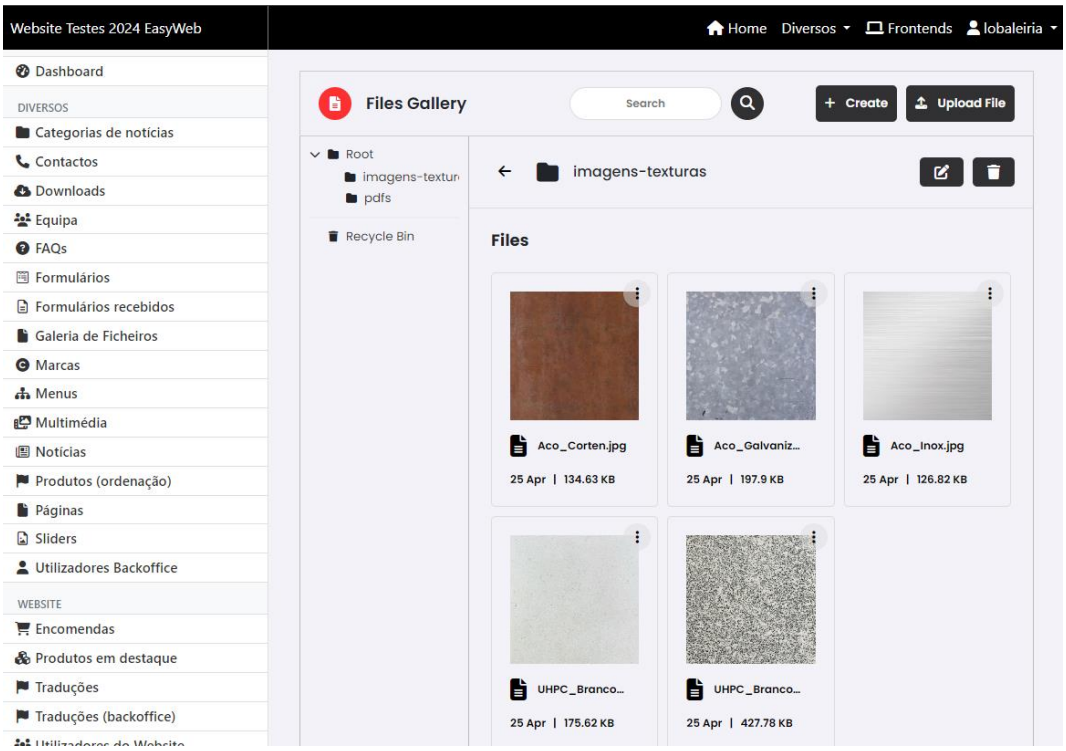


Figura 75 - Representação do módulo de gestão de ficheiros de servidores FTP.

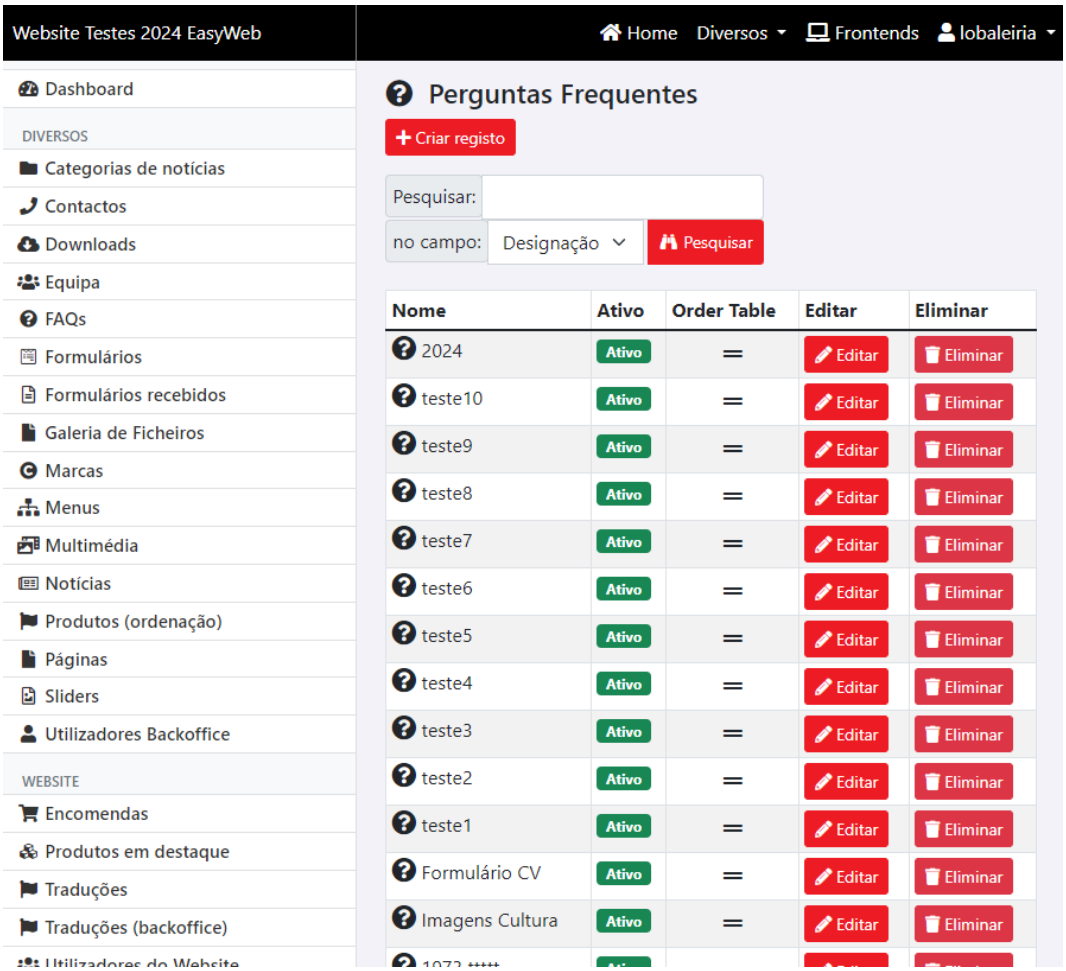


Figura 76 - Representação do componente de gestão de tabelas.

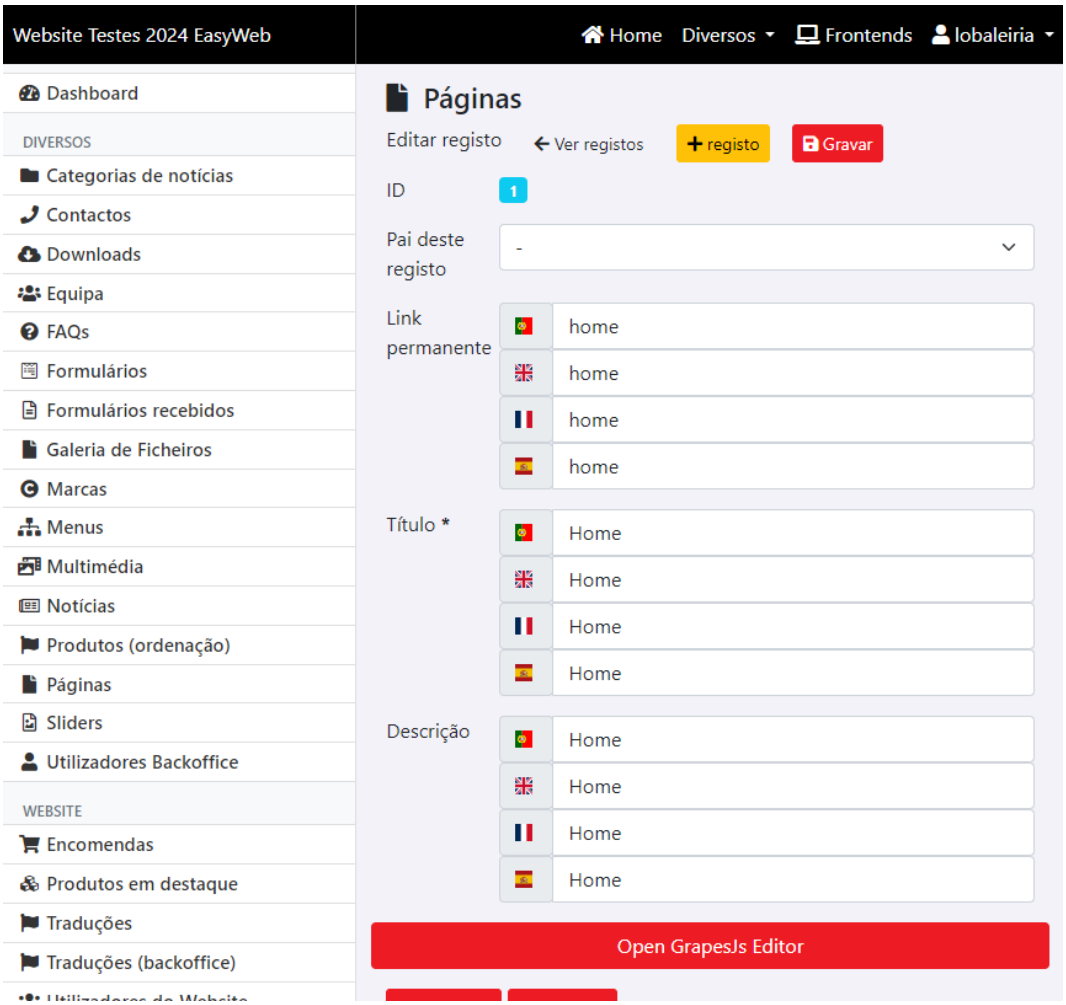


Figura 77 - Representação do componente de construção de páginas Web.

Com a integração pronta no EasyWeb para testes, encontra-se disponível a fase de adição em projetos reais, para que os clientes tenham novas funcionalidades ao usar o gestor de conteúdo e fornecem opiniões ou sugestões de melhoria para os componentes.

Dois projetos foram envolvidos para realizar os testes em ambientes reais, com o objetivo de obter resultados.

8.2.Resultados

Depois dos processos acima executados, conseguiu-se obter alguns resultados, de acordo com os utilizadores do EasyWeb. São importantes para o crescimento dos componentes e da própria plataforma.

Componente de carregamento de ficheiros

Começando pelo componente de carregamento de ficheiros: a integração com o módulo de gestão de ficheiros (ligado a um servidor FTP), trouxe resultados positivos na inserção de ficheiros através de arrasto.

Por estar integrado com ambientes que envolvem o carregamento de imagens em projetos reais, surgiu uma sugestão de pré-visualização, em miniatura, das imagens inseridas no componente, para ajudar na distinção, para além de mostrar o nome para tal.

Módulo de gestão de ficheiros

No caso do módulo de gestão de ficheiros, existem alguns pontos que foram detetados pelos clientes.

Devido à implementação de funcionalidades básicas no pequeno explorador de ficheiros do servidor FTP, sugeriu-se a adição de mais botões de ação que interagem com os ficheiros. Ações como "copiar para…", "mover para…" ou "restaurar ficheiro ou pasta" são algumas discutidas nas sugestões de melhoria.

Além disso, sugeriu-se a criação de uma área de favoritos, focado para pastas, para que os utilizadores possam ter acessos mais rápidos a pastas muito utilizadas na gestão de ficheiros, e também de uma secção com mais filtros de pesquisa, que foque também na pesquisa dentro da pasta.

Componente de gestão de tabelas

No componente de gestão de tabelas, sentiu-se, durante os testes, o limite de apresentação dos registos em linhas da tabela.

É um pormenor importante, porque, em situações onde existem dezenas de registos num módulo, o processo de arrasto das linhas para ordenação torna-se mais demorado, caso se pretende arrastar do topo da tabela para o fundo.

Ademais, a tabela teria de ser totalmente carregada, o que faz aumentar o tempo de carregamento da página e, por consequência, apostar em soluções de carregamento de mais linhas da tabela consoante a página apresenta as últimas.

Sugeriu-se, então, a implementação de paginação nas tabelas, para haver menos carregamento, e a criação de uma solução que conseguisse passar registos entre limites. Por exemplo, numa tabela que carregue vinte registos por página, a objetivo é permitir a passagem de um registo para a segunda página, quando possível.

Componente de construção de páginas Web

O componente é o único que não chegou a ser implementado em cenários reais, devido à não existência de novos projetos que não tenham as páginas criadas. Não seria possível aplicar em outros projetos mais recentes, porque necessitaria de aplicar mudanças drásticas por causa dos componentes utilizados. Este componente do EasyWeb obriga a criação de novos componentes para páginas e não consegue aproveitar os existentes.

No entanto, com a realização de testes na plataforma reservada para tal, os resultados são positivos, principalmente na integração com as páginas e com os componentes Laravel, mas alguns pormenores foram detetados e que necessitam de ser implementados em novos desenvolvimentos para melhoria.

Um desses pormenores encontra-se na base de dados, porque todos as configurações da página estão guardadas lá. Normalmente, todas as alterações aplicadas nos blocos de página ficam registadas em formato JSON na tabela das páginas. Contudo, o HTML dos componentes também fica guardado, o que é algo que ocupa mais. Esta situação acontece devido à adaptação feita entre os componentes Laravel e a biblioteca GrapesJS.

Uma solução para mitigar este constrangimento é conseguir guardar apenas a estrutura da página e registar apenas o nome do componente, como representa a Figura 78. Desta forma, menos quantidade de HTML fica registado, mais fácil é a compreensão da estrutura dos blocos na página e mais fácil é a aplicação de diferentes idiomas.

```
(...)

<div class="class-1">
    <div class="row-1">
        <div class="col-1">
            <Component_name></Component_name>
        </div>
        <div class="col-2">
            <Component_name_2></Component_name_2>
        </div>
    </div>
    <div class="row-2">
        <div class="col-1">
            <Component_name_3></Component_name_3>
        </div>
    </div>
</div>

(...)
```

Figura 78 - Uma solução que consiga diminuir o armazenamento do HTML na base de dados.

Com a solução apresentada, o servidor Web terá o papel de gerar o conteúdo HTML de cada componente ou bloco, com o idioma apropriado, e apresenta a página por completo. É um processo complexo, porque o editor também deve funcionar com esta proposta e o pacote Laravel utilizado terá de sofrer novas alterações.

A aplicação de diferentes idiomas no *website* é uma funcionalidade que muitos clientes pedem para inserir, para dar foco no mercado internacional. Portanto, deve ser visto como algo bem estudado e que seja escalável para resistir em novos desenvolvimentos. Neste momento, a implementação feita destina-se a *websites* com um único idioma.

Portanto, as sugestões para este componente passam em aplicar um botão de seleção de idioma dentro do editor, para ser mais fácil haver troca de idioma sem sair do editor. Todas as edições aplicadas em cada idioma na página ficam registadas na tabela, separando do conteúdo HTML.

Nestas sugestões de alteração, a tabela *ewpages* fica com a estrutura semelhante à da Figura 78, no atributo *gjs_page*, enquanto as configurações de CSS e dos *traits* ficam armazenas na tabela *ewpagelangs*, no atributo *gjs_traits*. A Figura 79 representa um diagrama com as tabelas utilizadas nas páginas Web de forma simplificada, não apresentando os demais atributos.

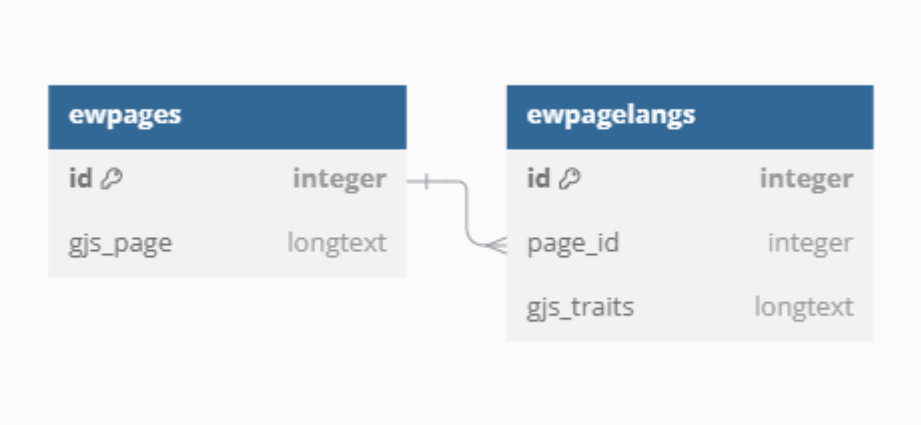


Figura 79 - A estrutura simplificada das tabelas do EasyWeb que envolvem páginas Web [63].

9. Conclusão

O projeto envolvido neste documento corresponde a um projeto de empresa e consiste na aplicação de novas funcionalidades, desenvolvidas sob a forma de componentes, numa plataforma de gestão de conteúdo para páginas Web, conhecida, genericamente, por CMS. A plataforma denomina-se como EasyWeb e provém, inicialmente, da empresa Incentea Marketing e Inovação e, mais tarde, da LOBA.

Ao longo do estudo do CMS, da procura e descrição das tecnologias envolventes, e da implementação dos componentes, a minha compreensão da parte teórica da gestão de conteúdo ficou mais completa, visto que nem todos os temas foram explorados profundamente no local de trabalho. Desta forma, deu para perceber o rigor que UI e UX aplicam nestes sistemas, sempre com o objetivo de facilitar o uso para todos.

Concluído o trabalho de desenvolvimento do projeto, pode-se afirmar que todos os objetivos definidos inicialmente, durante a introdução do projeto, foram cumpridos, com a exceção do componente de construção de páginas, cujos testes em projetos reais não foram implementados devido à dificuldade em converter os componentes existentes para componentes que o editor aceita.

As melhorias desenvolvidas para o EasyWeb foram realizadas com sucesso, mas foi implementado o essencial dos componentes, para ter uma base dos componentes pronta para novos pedidos de clientes nos próximos momentos. Um desses componentes se destaca em relação aos demais, devido à dificuldade em integrar o editor com o Laravel. De facto, no componente de construção de páginas Web, o pacote Laravel está minimamente funcional, mas necessitou de novas implementações no próprio pacote para conseguir integrar os componentes Laravel. Portanto, este projeto foi mais exigente em alguns componentes que outros, encarecendo dificuldade ao integrar o GrapesJS com os componentes Laravel.

Relativamente ao módulo de gestão de ficheiros, o envolvimento de servidores FTP sempre trouxe um levantamento de questões quanto à segurança, com base nas vulnerabilidades relatadas nos últimos tempos. Continua a ser um dos pontos que pode afetar a evolução do módulo.

Com os resultados obtidos de cada componente, espera-se que este projeto tenha um processo de melhoria contínua. Como propostas de trabalho futuro, recomenda-se melhorar o editor GrapesJS e o módulo de gestão de ficheiros, porque são os mais utilizados ao longo da utilização da plataforma EasyWeb e porque o editor necessita de integrar o suporte multi--línguas em consonância com a capacidade já existente na plataforma EasyWeb.

Referências Bibliográficas

[1] Kinsta, “What Is a Content Management System (CMS)?,” Kinsta, 18 Outubro 2023. [Online]. Available: https://kinsta.com/knowledgebase/content-management-system/. [Acedido em 8 Janeiro 2024].

[2] Contentstack, “History of content management systems and rise of headless CMS,” Contentstack, 2 Novembro 2023. [Online]. Available: https://www.contentstack.com/blog/all-about-headless/content-management-systems-history-and-headless-cms. [Acedido em 13 Janeiro 2024].

[3] Google Blogger, “Blogger.com – crie facilmente um blogue único e apelativo,” Google Blogger, 27 Junho 2023. [Online]. Available: https://www.blogger.com/about/. [Acedido em 18 Novembro 2023].

[4] A. Zantal-Wiener, “A Brief Timeline of the History of Blogging,” HubSpot, 20 Outubro 2020. [Online]. Available: https://blog.hubspot.com/marketing/history-of-blogging. [Acedido em 18 Novembro 2023].

[5] WordPress, “About,” WordPress, [Online]. Available: https://wordpress.org/about/. [Acedido em 14 Janeiro 2024].

[6] Drupal, “About - Drupal,” Drupal, 9 Janeiro 2023. [Online]. Available: https://www.drupal.org/about. [Acedido em 14 Janeiro 2024].

[7] The Joomla! Project, “Joomla! - Content Management System to build websites & apps,” The Joomla! Project, [Online]. Available: https://www.joomla.org/about-joomla.html. [Acedido em 14 Janeiro 2024].

[8] Sitecore, “What is CMS architecture,” Sitecore, [Online]. Available: https://www.sitecore.com/knowledge-center/digital-marketing-resources/what-is-cms-architecture. [Acedido em 15 Janeiro 2024].

[9] M. Marinina, “CMS State of the Art: What You Need to Know in 2019 - Chief Marketer,” Chief Marketer, 7 Janeiro 2019. [Online]. Available: https://chiefmarketer.com/cms-state-of-the-art-what-you-need-to-know-in-2019. [Acedido em 11 Dezembro 2023].

[10] DMSInfosystem, “Creating and Managing User Roles and Permissions in CMS,” Medium, 25 Maio 2023. [Online]. Available: https://medium.com/@dmsinfosystem1/creating-and-managing-user-roles-and-permissions-in-cms-1015d6056e8a. [Acedido em 22 Janeiro 2024].

[11] M. Carlyle, “OneQuext,” OneQuext, [Online]. Available: https://onequext.com/blog/what-is-ui-ux-design-and-how-can-a-cms-help-both. [Acedido em 22 janeiro 2024].

[12] H. Poirier, “How to measure the scalability of CMS software,” Agility CMS, 21 Setembro 2022. [Online]. Available: https://agilitycms.com/resources/posts/how-to-measure-the-scalability-of-cms-software. [Acedido em 25 Janeiro 2024].

[13] M. Vertal, “Scaling Your Website? Choose the Right CMS Architecture,” CrafterCMS, [Online]. Available: https://craftercms.org/blog/2020/scaling-your-website-choose-the-right-cms-architecture. [Acedido em 26 Janeiro 2024].

[14] GTCSYS, “How does a CMS handle website scalability and growth?,” GTCSYS, 31 Julho 2023. [Online]. Available: https://gtcsys.com/faq/how-does-a-cms-handle-website-scalability-and-growth. [Acedido em 25 Janeiro 2024].

[15] Equipa Lumis, “Dicas para empresa crescer com o SEO usando plataforma CMS,” Lumis, [Online]. Available: https://www.lumis.com.br/a-lumis/blog/dicas-para-empresa-crescer-com-o-seo-usando-plataforma-cms.htm#Como-crescer-com-seo-usando-plataforma-cms?-dicas-para-sua-empresa!. [Acedido em 26 Janeiro 2024].

[16] Accessi.org, “Understanding CMS Accessibility: Guidelines and Importance,” Accessi Blog - ADA, WCAG, AODA, and Web Accessibility, 16 Agosto 2023. [Online]. Available: https://www.accessi.org/blog/making-your-cms-accessible/#:~:text=Web%20Accessibility%20Standards%20for%20CMS&text=Th

e%20current%20version%20is%20WCAG,accessible%20to%20people%20with%20disabilities.. [Acedido em 26 Janeiro 2024].

[17] T. Murphy, “The top 4 content management trends in 2024,” TechTarget, 21 Dezembro 2023. [Online]. Available: https://www.techtarget.com/searchcontentmanagement/feature/The-top-5-content-management-trends. [Acedido em 25 Janeiro 2024].

[18] AIContentfy Team, “Emerging Content Management Trends: What's in Store for the Future,” AIContentfy, 6 Novembro 2023. [Online]. Available: https://aicontentfy.com/en/blog/emerging-content-management-trends-whats-in-store-for-future. [Acedido em 25 Janeiro 2024].

[19] Google Trends, “Google Trends,” Google, 15 Setembro 2024. [Online]. Available: https://trends.google.com/trends/explore?date=all&q=%2Fm%2F02vtpl,%2Fm%2F07qb81,Wix,%2Fm%2F02ntbw8,%2Fg%2F11r2rm2y37&hl=pt-PT. [Acedido em 15 Setembro 2024].

[20] Google Trends, “Google Trends,” Google, 15 Setembro 2024. [Online]. Available: https://trends.google.com/trends/explore?date=all&q=%2Fm%2F02vtpl,%2Fm%2F01641s,%2Fg%2F121mks5z,%2Fg%2F11fmgs3lc7,%2Fm%2F0w8c7m4&hl=pt-PT. [Acedido em 15 Setembro 2024].

[21] Brightspot, “Your custom, hybrid content management system,” Brightspot, [Online]. Available: https://www.brightspot.com/brightspot-cms. [Acedido em 20 janeiro 2024].

[22] Contentful, “Composable Content Platform,” Contentful, [Online]. Available: https://www.contentful.com/products/. [Acedido em 20 janeiro 2024].

[23] Prismic, “Feature overview,” Prismic, [Online]. Available: https://prismic.io/features. [Acedido em 20 janeiro 2024].

[24] Zesty.io, “Why Zesty.io headless CMS?,” Zesty.io, [Online]. Available: https://www.zesty.io/why-zesty. [Acedido em 20 janeiro 2024].

[25] Sitecore, “Why us?,” Sitecore, [Online]. Available: https://www.sitecore.com/explore/why-sitecore. [Acedido em 20 janeiro 2024].

[26] The Contentstack Team, “Headless CMS: Advantages and challenges for digital enterprises | Contentstack,” Contentstack, 29 Novembro 2023. [Online]. Available: https://www.contentstack.com/blog/all-about-headless/headless-cms-advantages-challenges-digital-enterprises. [Acedido em 23 Janeiro 2024].

[27] Mozilla Corporation, “HTML Drag and Drop API - Web APIs | MDN,” MDN Web Docs, 25 julho 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API. [Acedido em 10 agosto 2024].

[28] Atlassian, “Scrumban: a combinação de duas metodologias ágeis,” Atlassian, [Online]. Available: https://www.atlassian.com/br/agile/project-management/scrumban. [Acedido em 10 agosto 2024].

[29] Dropzone.dev, “Introduction | Dropzone,” Dropzone.dev, [Online]. Available: https://docs.dropzone.dev/. [Acedido em 12 agosto 2024].

[30] Dropzone.dev, “Configuration Options | Dropzone,” Dropzone.dev, [Online]. Available: https://docs.dropzone.dev/configuration/basics/configuration-options. [Acedido em 12 agosto 2024].

[31] PQINA, “Easy File Uploading With JavaScript | FilePond,” PQINA, 26 julho 2024. [Online]. Available: https://pqina.nl/filepond/docs/. [Acedido em 12 agosto 2024].

[32] Uppload.js, “Uppload,” Uppload, [Online]. Available: https://uppload.js.org. [Acedido em 12 agosto 2024].

[33] Pixabay, “Pixabay,” Pixabay, [Online]. Available: https://pixabay.com/. [Acedido em 12 agosto 2024].

[34] Unsplash, “Lindas imagens e fotos gratuitas | Unsplash,” Unsplash, [Online]. Available: https://unsplash.com/pt-br. [Acedido em 12 agosto 2024].

[35] React-Dropzone, “react-dropzone,” React-Dropzone, [Online]. Available: https://react-dropzone.js.org. [Acedido em 12 agosto 2024].

[36] PQINA, “Pintura Image Editor,” PQINA, [Online]. Available: https://pqina.nl/pintura. [Acedido em 12 agosto 2024].

[37] Codepen, “React Dropzone Uploader,” Codepen, [Online]. Available: https://codepen.io/kylebebak/pen/wYRNzY. [Acedido em 12 agosto 2024].

[38] Webix, “File Manager, UI Complex Widgets Webix Docs,” Webix Docs, [Online]. Available: https://docs.webix.com/desktop__filemanager.html. [Acedido em 18 fevereiro 2024].

[39] DevExtreme, “Vue Components - DevExtreme Vue,” DevExtreme, [Online]. Available: https://js.devexpress.com/Vue. [Acedido em 18 fevereiro 2024].

[40] Syncfusion, “JavaScript File Manager | HTML5 File Explorer | Syncfusion,” Syncfusion, [Online]. Available: https://www.syncfusion.com/javascript-ui-controls/js-file-manager. [Acedido em 18 fevereiro 2024].

[41] sindu12jun, “sindu12jun-TableDragger,” GitHub, 26 dezembro 2016. [Online]. Available: https://sindu12jun.github.io/table-dragger. [Acedido em 12 agosto 2024].

[42] SortableJS, “SortableJS,” GitHub, 14 janeiro 2024. [Online]. Available: https://sortablejs.github.io/Sortable. [Acedido em 12 agosto 2024].

[43] Material React Table, “Material React Table V3,” Material React Table, [Online]. Available: https://www.material-react-table.com. [Acedido em 12 agosto 2024].

[44] Material React Table, “Column Dnd Ordering Material React Table Example,” Material React Table, [Online]. Available: https://www.material-react-table.com/docs/examples/column-ordering. [Acedido em 12 agosto 2024].

[45] GrapesJS, “GrapesJS - Free and Open Source Web Template Editor Framework,” GrapesJS, [Online]. Available: https://grapesjs.com. [Acedido em 12 agosto 2024].

[46] GrapesJS, “Introduction | GrapesJS,” GrapesJS, [Online]. Available: https://grapesjs.com/docs. [Acedido em 12 agosto 2024].

[47] GitHub, “laravel-grapesjs,” GitHub, 26 maio 2023. [Online]. Available: https://github.com/mjawad096/laravel-grapesjs. [Acedido em 12 agosto 2024].

[48] GitHub, “PHPageBuilder,” GitHub, 3 julho 2024. [Online]. Available: https://github.com/HansSchouten/PHPagebuilde. [Acedido em 12 agosto 2024].

[49] GitHub, “Laravel-Pagebuilder,” GitHub, 3 julho 2024. [Online]. Available: https://github.com/HansSchouten/Laravel-Pagebuilder. [Acedido em 12 agosto 2024].

[50] PHPageBuilder, “PHP Page Builder | Add a page builder to your PHP project,” PHPageBuilder, [Online]. Available: https://www.phpagebuilder.com/. [Acedido em 12 agosto 2024].

[51] Vvveb, “Vvveb,” Vvveb, [Online]. Available: https://www.vvveb.com. [Acedido em 12 agosto 2024].

[52] Vvveb, “VvvebJs,” Vvveb, [Online]. Available: https://www.vvveb.com/vvvebjs/editor.html. [Acedido em 12 agosto 2024].

[53] GitHub, “VvvebJs,” GitHub, 8 agosto 2024. [Online]. Available: https://github.com/givanz/VvvebJs. [Acedido em 12 agosto 2024].

[54] Mozilla Corporation, “<input type="file"> - HTML: HyperText Markup Language | MDN,” MDN Web Docs, 25 julho 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file. [Acedido em 10 agosto 2024].

[55] Mozilla Corporation, “XMLHttpRequest - Web APIs | MDN,” MDN Web Docs, 03 Abril 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest. [Acedido em 10 Maio 2024].

[56] Mozilla Corporation, “Date - JavaScript | MDN,” MDN Web Docs, 25 julho 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date. [Acedido em 10 agosto 2024].

[57] P. Nguyen, “Phuoc Nguyen,” Phuoc Nguyen, 03 Abril 2020. [Online]. Available: https://phuoc.ng/collection/html-dom/drag-and-drop-table-column/. [Acedido em 15 Fevereiro 2024].

[58] GrapesJS, “Trait Manager | GrapesJS,” GrapesJS, [Online]. Available: https://grapesjs.com/docs/modules/Traits.html#add-traits-to-components. [Acedido em 29 junho 2024].

[59] Laravel, “Componentes | Laravel - The PHP Framework For Web Artisans,” Laravel, [Online]. Available: https://laravel.com/docs/11.x/blade#components. [Acedido em 29 junho 2024].

[60] GrapesJS, “Commands | GrapesJS,” GrapesJS, [Online]. Available: https://grapesjs.com/docs/modules/Commands.html#basic-configuration. [Acedido em 29 junho 2024].

[61] GrapesJS, “Components API | GrapesJS,” GrapesJS, [Online]. Available: https://grapesjs.com/docs/api/components.html#available-events. [Acedido em 29 junho 2024].

[62] GrapesJS, “Storage Manager API | GrapesJS,” GrapesJS, [Online]. Available: https://grapesjs.com/docs/api/storage_manager.html#available-events. [Acedido em 29 junho 2024].

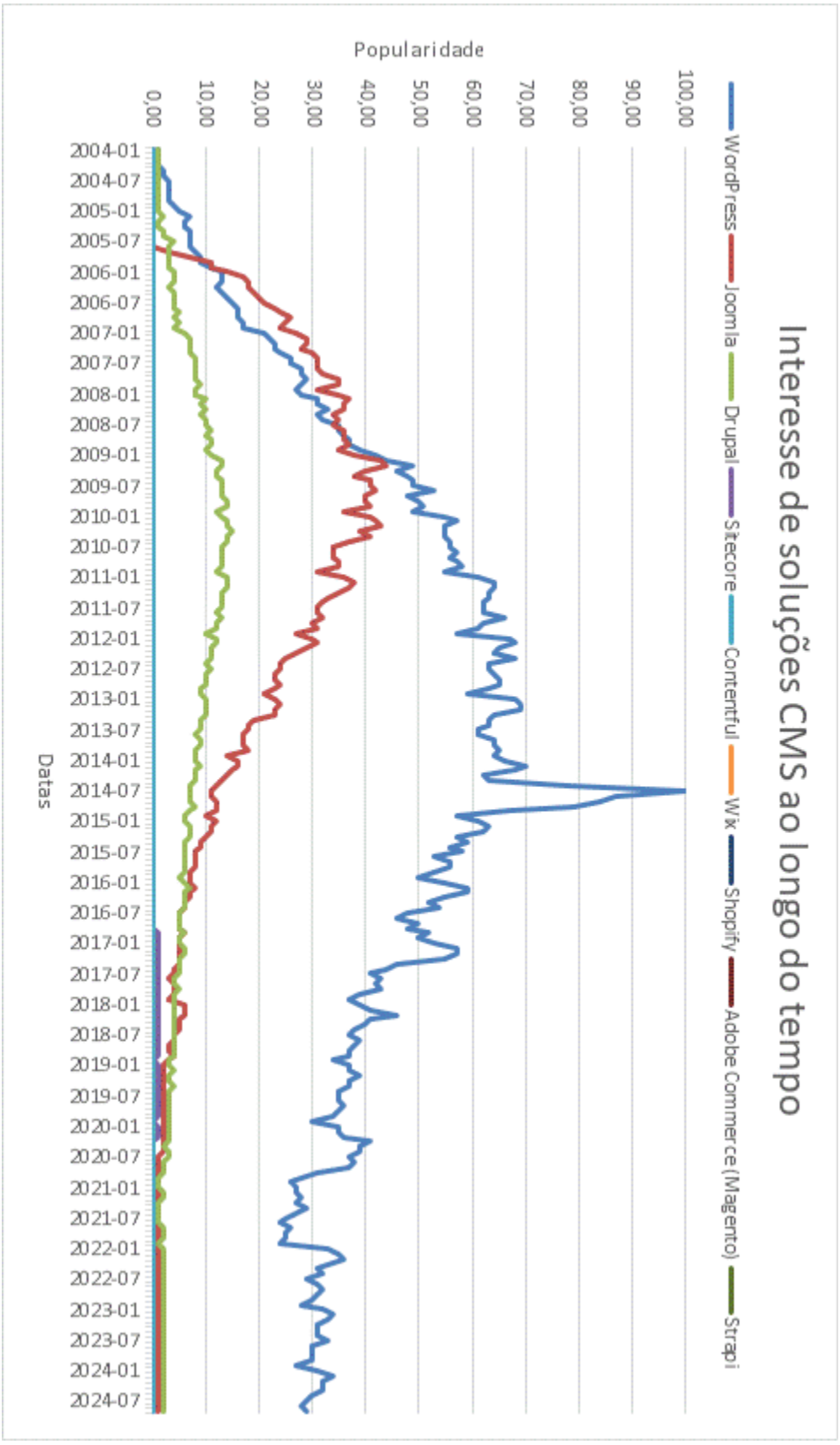
[63] dbdiagram.io, “dbdiagram.io - Database Relationship Diagrams Design Tool,” dbdiagram.io, [Online]. Available: https://dbdiagram.io/home. [Acedido em 25 agosto 2024].

[64] S. Coleman, “Everything you need to know about hybrid CMS,” Brightspot, 17 Março 2023. [Online]. Available: https://www.brightspot.com/cms-architecture/hybrid-cms/what-is-a-hybrid-cms. [Acedido em 9 fevereiro 2024].

[65] Mozilla Corporation, “DataTransfer - Web APIs | MDN,” MDN Web Docs, 15 Novembro 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/DataTransfer. [Acedido em 25 Abril 2024].

Anexos

Anexo A – Gráfico de interesse de soluções CMS ao longo dos últimos vinte anos



Anexo B - Entrevista sobre a plataforma EasyWeb

- **Título**: Estado da arte do EasyWeb - do seu nascimento até à atualidade**;**
- **Entrevistado**: Luís Oliveira;
- **Data da entrevista**: 26 de dezembro de 2023;
- **Entrevistador**: Sandro Baptista;
- **Objetivo da entrevista**: Dar a conhecer um pouco da história da plataforma e planos para futuro.

Luís Oliveira é considerado a pessoa com mais experiência na programação Web da equipa, contando com mais de 25 anos em projetos que envolvem JavaScript e PHP. Ele é, para quem o acompanha no local de trabalho, o criador da plataforma EasyWeb. Esta entrevista tem o propósito de esclarecer informações mais detalhadas sobre a plataforma, de modo a ajudar na compreensão da sua história. Esta foi realizada no âmbito da unidade curricular Projeto de Empresa, do segundo ano do mestrado de Engenharia Informática em Computação Móvel, cujo objetivo é desenvolver funcionalidades que melhorem a gestão de conteúdos e as páginas de *websites* no EasyWeb.

1. **SB**: Olá, Luís. Obrigado por teres aceitado o meu convite para esta entrevista. É sempre bom estar a falar com um dos envolventes deste projeto, pois podem contribuir com mais informação interessante.
   LO: Olá, Sandro. Obrigado pelo convite. De facto, é verdade, saber mais detalhes quando se desenvolve no projeto torna as pessoas mais inseridas, e isso ajuda na evolução desta plataforma.
2. **SB**: Vamos dar início à entrevista. Bem, o EasyWeb é a plataforma CMS que se destaca na empresa. E esta aparenta conter história... Como foi o seu nascimento?
   LO: O EasyWeb nasceu em resposta à necessidade de uma plataforma que contenha um conjunto de funcionalidades adequadas ao desenvolvimento de Websites. Ao longo do tempo, algumas dessas funcionalidades foram ficando associadas dentro de um conjunto, ao qual se juntou um *backoffice* de gestão do Website por uma questão de conveniência e facilidade de utilização. No início, não utilizava uma framework de desenvolvimento, era basicamente um conjunto de scripts com funções adequadas ao propósito.

3. **SB**: Quais foram os requisitos essenciais identificados para este CMS durante o processo de conceção e planeamento?

   LO: Principalmente robustez e estabilidade, segurança, elevado desempenho, baixo consumo de recursos, sempre a pensar na facilidade de instalação em alojamentos standard já existentes no mercado e na integração com ERP para interligação de dados para áreas reservadas / comércio eletrónico.

4. **SB**: Por que se optou por desenvolver uma nova plataforma em Laravel em vez de utilizar plataformas CMS existentes no mercado?

   LO: As primeiras versões do Easyweb ainda não utilizavam a framework Laravel. Num dado ponto, em que a framework Laravel se encontrava na versão 4, verificámos que continha um conjunto de funcionalidades que coincidiam com os requisitos base que já tinham sido definidos para o EasyWeb (robustez, estabilidade, segurança, integração, entre outros) e suplementava as funcionalidades já existentes no EasyWeb. As plataformas existentes no mercado não satisfaziam em alguns dos requisitos: Tinham falhas de segurança, eram lentas e não integravam facilmente com ERP. Integração é essencial para a implementação de áreas reservadas e comércio eletrónico B2C / B2B.

5. **SB**: EasyWeb está construído em PHP, na framework Laravel. Porquê o uso de Laravel? Quais foram os principais critérios considerados ao decidir utilizar o Laravel como base para este CMS?

   LO: O Laravel preenche os requisitos que tínhamos definido para a plataforma, adicionando funcionalidades importantes: autenticação simplificada, funções para interligação com múltiplas fontes de dados, migrações, *templates*, fácil envio de notificações por e-mail, tarefas agendadas, rotas, entre outras.

6. **SB**: Acredito que estiveste a estudar a concorrência naquela altura. Quais são as vantagens que o Laravel oferecia e oferece nesta plataforma que não seriam alcançadas com outras soluções prontas?

   LO: O Laravel, desde o início, apresentava-se como uma plataforma com elevado nível de segurança, robusta e escalável. As soluções concorrentes apresentavam requisitos de sistema mais elevados para as mesmas funcionalidades e não eram tão seguras e / ou não eram tão capazes em termos de integração com fontes de dados externas.

7. **SB**: Como enfrentaste os desafios em criar uma plataforma do zero em Laravel, considerando o tempo de desenvolvimento e os recursos necessários?

LO: Foi um processo orgânico: a primeira versão do Easyweb era um conjunto de scripts orientados a um propósito de implementar Websites de qualidade, foi um processo normal passar desse ponto para *framework* Laravel, visto a linguagem de desenvolvimento (PHP) ser a mesma e a filosofia MVC (Model / View / Controller) do Laravel ser algo familiar mesmo às primeiras versões do EasyWeb.

8. **SB**: Apenas houve uma pessoa envolvida no projeto? Se sim, houve motivos para que tenha sido assim, devido a, por exemplo, disponibilidade para trabalho normal e o desenvolvimento ou até documentação?

   LO: Ao longo do tempo, existiram várias pessoas envolvidas no projeto, as quais contribuíram com aspetos específicos da plataforma, como as funcionalidades de integração com ERP, secções do *backoffice* de gestão, etc.

9. **SB**: Depois de terminar a base do EasyWeb, sentiste, de imediato, que havia necessidade em melhorar certas funcionalidades ou processos? Percebeste que poderia ter realizado de outra forma em alguns casos?

   LO: O EasyWeb sempre teve uma base evolutiva / orgânica. Em consequência disso, ao longo do tempo foram-se identificando melhorias, as quais foram sendo implementadas por ordem de importância das mesmas.

10. **SB**: Para terminar a entrevista, neste momento, surgiu a oportunidade de realizar melhorias no EasyWeb. Estas propostas de melhoria levantaram-se neste momento ou foram aparecendo ao longo dos últimos anos?

    LO: No decorrer do tempo, foram sendo identificadas, tanto por utilizadores da plataforma como a nível interno, oportunidades de melhoria interessantes. Essas melhorias são percebidas e registadas, sendo atribuídas prioridades e dada a prioridade mais elevada aos upgrades mais relevantes para implementação.

11. **SB**: Muito bem. Bom, chegou o fim desta interessante entrevista. Luís, muito obrigado pela disponibilidade. Foi um prazer estar a falar contigo e a conhecer um pouco mais sobre a história do EasyWeb.

    LO: Ora essa. Foi um prazer estar a falar contigo e a dar a conhecer um pouco da história do EasyWeb.

Esta entrevista pretendia partilhar informação mais detalhada acerca da plataforma EasyWeb, de modo a ajudar na compreensão da sua história e na escrita da tese de mestrado.

Muitos tópicos foram abordados ao longo da conversa, desde o seu início, a escolha das tecnologias, a sua manutenção e o ponto de situação neste momento.

Conforme a informação obtida, o EasyWeb foi fundado em puro PHP, antes mesmo da chegada da versão 4 do Laravel. Por ser considerada uma tecnologia segura e por oferecer diversas funcionalidades relevantes, decidiu-se migrar todo o projeto. A integração ocorreu sem grandes problemas. Com uma "base evolutiva", é simples implementar melhorias e outras integrações com várias plataformas ERP. Devido à necessidade de aprimorar a plataforma, optou-se por iniciar, em 2023, uma nova etapa de melhorias.

Anexo C – Manual de Utilizador do Componente *Upload* de Ficheiros

DDFU.js - Drag and Drop File Upload Library

DDFU.js é uma biblioteca JavaScript que permite o utilizador adicionar um sistema de *upload* de ficheiros, em formulários para websites, de forma simples e rápida. Este pacote é um formulário adaptado que apenas recebe ficheiros e os envia para o servidor.

Introdução

DDFU.js é uma biblioteca JavaScript que permite o utilizador adicionar um sistema de upload de ficheiros, em formulários para sites, de forma simples e rápida.

Dentro de um formulário, o componente está preparado para realizar upload de ficheiros no servidor Web num processo isolado da submissão do formulário. No entanto, o carregamento é feito depois de clicar no botão da submissão, para que o utilizador veja o progresso do carregamento dos ficheiros e que o carregamento seja feito antes de enviar os dados para o servidor.

A biblioteca é composta por alguns ficheiros necessários para o seu funcionamento:

- dois ficheiros JavaScript;
- um ficheiro CSS;
- um conjunto de ícones SVG (`assets`).

Instalação

A instalação baseia-se na adição de dois ficheiros ao projeto:

- ficheiro de código JavaScript;
- ficheiro de estilos CSS.

Pré-requisitos

Para instalar a biblioteca, não é necessário importar outras bibliotecas. O desenvolvimento da biblioteca foi feito com JavaScript puro, sem a necessidade de importar outras bibliotecas.

Passos de instalação

Para instalar a biblioteca, basta adicionar o ficheiro de código e o ficheiro de estilos, no local onde haja acesso às páginas Web. O ficheiro de configurações não é para ser importado. Depois adicionar o seguinte código ao ficheiro HTML:

html

```
<html>
    <head>
        <link rel="stylesheet" href="path/to/ddfu.css"/>
    </head>
    <body>
        <script src="path/to/ddfu.js"></script>
    </body>
</html>
```

A instalação é muito simples, visto que foi construído para ser utilizado em qualquer tipo de *website*.

Configuração

A biblioteca contém um ficheiro de configurações, denominado:

- `DDFU_Configs.js`

Este ficheiro contém as configurações da biblioteca, que podem ser alteradas pelo utilizador.

As configurações disponíveis estão descritas na secção Documentação API.

Utilização

A utilização da biblioteca é muito simples, visto que foi construída para ser utilizada em qualquer tipo de *website*. A biblioteca foi construída para ser utilizada em formulários, mas pode ser utilizada em qualquer parte do website, desde que seja dentro do corpo da página.

Implementação

Para utilizar a biblioteca, basta adicionar o seguinte código ao ficheiro HTML:

html

```
<form (...)>
    <div id="(...)" class="ddfu_drop-area" ddfu-action="(...)" (ddfu-re-
quired)>
        <p>Drop files in this area or</p>
        <input type="file" name="file[]" id="any_file_id_name" ti-
tle="File" accept="(...)">
```

```
        <label for="any_file_id_name" class="ddfu_button">Select
file</label>
    </div>
    (...)
</form>
```

Também é necessário adicionar o seguinte código JavaScript:

javascript

```
const ddfu = new DDFU("id_do_elemento_ddfu_drop_area");
ddfu.init();
```

Toda esta estrutura é simples, porque apenas utiliza classes com nomes definidos, tanto no JavaScript como no CSS, para que o utilizador possa alterar o estilo do web*site* sem que a biblioteca deixe de funcionar.

Exemplos de uso

O seguinte exemplo demonstra como utilizar a biblioteca numa situação prática.

- HTML:

html

```
<form>
      <div id="ddfu_drop_area" class="ddfu_drop-area" ddfu-action="up-
load.php" ddfu-required>
            <p>Drop files in this area or</p>
            <input type="file" name="file[]" id="file" title="File" ac-
cept=".pdf" multiple>
            <label for="file" class="ddfu_button">Select file</label>
      </div>
      <button type="submit">Submit</button>
</form>
```

- JavaScript:

javascript

```
const ddfu = new DDFU("ddfu_drop_area");
ddfu.init();
```

Neste exemplo, o formulário contém apenas o componente, podendo servir em situações como a conversão de ficheiros, a criação de um sistema de partilha de ficheiros, entre outros.

Parâmetros e opções

Atributos do elemento `<div class="ddfu_drop-area">` (Tabela 7, Tabela 8):

Tabela 7 - Atributos do elemento `<div class="ddfu_drop-area">` (parte 1).

Atributo	Descrição	Obrigatório
`ddfu-action`	URL para onde os ficheiros carregados são enviados para o servidor	Sim
`ddfu-required`	Indica se é obrigatório o campo estar preenchido. **Nota**: não utilizar o atributo `required` no elemento `<input>`	Não

Tabela 8 - Atributos do elemento `<div class="ddfu_drop-area">` (parte 2).

Atributo	Tipo	Valor Padrão	Exemplo
`ddfu-action`	`string`	-	`ddfu-action="https:\\www.example.com/upload"`
`ddfu-required`	-	-	`ddfu-required`

Todo o conteúdo que esteja dentro do `<div class=”ddfu_drop-area”>` (exceto o `<input id=”file”>` e `<label class=”ddfu_button”>`, porque são obrigatórios para carregar ficheiros) é opcional, podendo ser alterado ou removido. Refere-se ao texto que aparece na zona de *drop*. O texto foi deixado em HTML para que o utilizador possa alterar o estilo do texto, se assim o desejar.

Dentro do `<div class=”ddfu_drop-area”>`, para além dos elementos de formulário, à medida que os ficheiros são associados, são criados elementos `<div class=”ddfu_file”>` que contêm o nome do ficheiro, o tamanho e um botão para remover o ficheiro da lista de ficheiros carregados.

De lembrar que o `<input>` pode ter o atributo `multiple` para que o utilizador possa carregar vários ficheiros de uma só vez, tanto com o arrasto, como também com a janela de carregamento de ficheiros. O *script* reage com ou sem o atributo `multiple`, podendo ser observado na Figura 80 e na Figura 81.

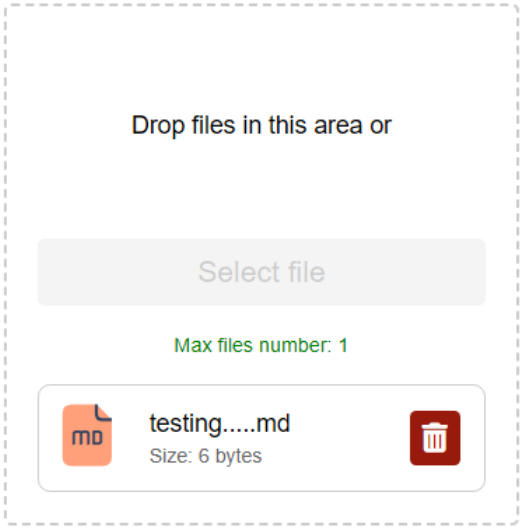


Figura 80 - *Input* dos ficheiros sem o atributo `multiple`.

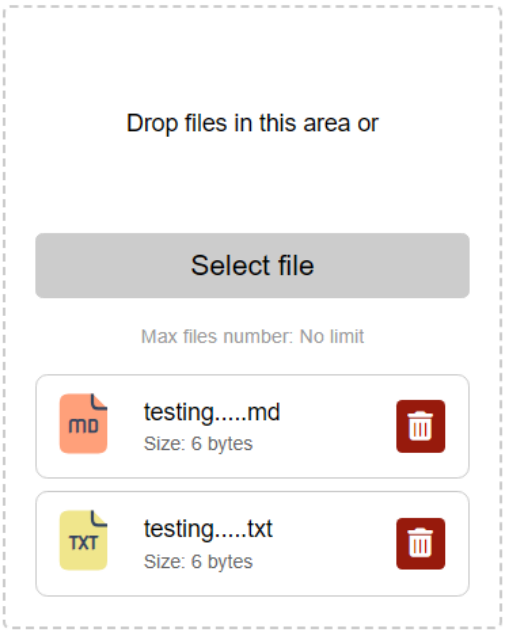


Figura 81 - *Input* dos ficheiros com o atributo `multiple`.

Observando as imagens, o que distingue se o elemento do formulário permite ou não múltiplos valores é a frase que fica por baixo do botão para carregamento manual de ficheiros. No caso de não permissão, quando o número de ficheiros carregados é igual ao limite definido, a frase muda de cor para verde e o botão fica desativado. Enquanto não estiver nenhum selecionado, a frase fica a vermelho e o botão fica ativo. Por outro lado, no caso de permissão, a frase fica sempre a cinzento, porque é indiferente ao número de ficheiros escolhidos, e o botão fica sempre ativo.

Casos de uso

Estes são alguns exemplos de utilização da biblioteca:

- Conversão de ficheiros:

```html
<section>
      <div class="section-header">
            <h1>Upload Files</h1>
            <p>Drop files or select files to upload</p>
      </div>
      <form id="form-2" action="submit_form.php" method="post" accept-
charset="utf-8" enctype="multipart/form-data">
            <fieldset class="ddfu_drop-area" id="upload-files" ddfu-ac-
tion="upload.php" ddfu-required>
                  <p>Drop files in this area or</p>
```

```
                  <input type="file" name="file[]" id="file" title="File"
accept="*/*" multiple>
                  <label for="file" class="ddfu_button">Select Files</la-
bel>
            </fieldset>

            <button type="submit">Upload</button>
      </form>
      <div class="section-footer">
            <p>Thank you for uploading files</p>
      </div>
</section>
```

- Formulários com submissão de ficheiro:

html

```
<section>
      <div class="section-header">
            <h1>Submit CV form</h1>
            <p>Fill the form and upload your CV</p>
      </div>
      <form id="form-1" action="submit_form.php" method="post" accept-
charset="utf-8" enctype="multipart/form-data">
            <fieldset>
                  <label for="name">Name *:</label>
                  <input type="text" name="name" id="name" required>
            </fieldset>
            <fieldset>
                  <label for="email">E-mail *:</label>
                  <input type="email" name="email" id="email" required>
            </fieldset>
            <fieldset class="radio-group">
                  <label for="gender">Gender *:</label>
                  <div class="radio-input">
                        <input type="radio" name="gender" id="male"
value="male" required>
                        <label for="male">Male</label>
                  </div>
                  <div class="radio-input">
                        <input type="radio" name="gender" id="female"
value="female" required>
                        <label for="female">Female</label>
                  </div>
            </fieldset>
            <fieldset>
                  <label>CV file *:</label>
                  <div class="ddfu_drop-area" id="upload-cv" ddfu-ac-
tion="upload.php" ddfu-required>
                        <p>Drop your CV file in this area or</p>
                        <input type="file" name="file[]" id="file1" ti-
tle="CV File" accept=".pdf,.doc,.docx">
                        <label for="file1" class="ddfu_button">Select CV
File</label>
```

```
                  </div>
            </fieldset>
            <button type="submit">Upload CV</button>
      </form>
      <div class="section-footer">
            <p>Thank you for submitting your CV</p>
      </div>
</section>
```

Funcionalidades

A sua estrutura é simples, em que o elemento `<div class="ddfu_drop-area">` é o parente de todo o funcionamento. Esse elemento é constituído por dois atributos, `ddfu-action` e `ddfu-required`, essenciais para o funcionamento do componente.

Dentro, existem o tal elemento `<input type="file">` e o seu `<label>`, que serve como botão para carregamento manual de ficheiros, porque a intenção é esconder o elemento principal. Como os elementos `<label>` estão associados ao elemento `<input>` através do atributo `for`, o evento clique no botão ativa o evento clique no elemento `<input>`. `<label>` tem a classe `ddfu_button`.

Este pormenor facilita o processo manual, principalmente para os utilizadores que não estão habituados a arrastar ficheiros para o navegador ou quando o arrasto não é permitido, que é o caso dos dispositivos móveis.

De seguida, um detalhe do elemento é mostrado. Como referido anteriormente, o número máximo de ficheiros é referido, podendo mudar de cor consoante o que está na lista de ficheiros. Este elemento tem a classe `ddfu_counter`.

Por baixo do estado de carregamento de ficheiros, existe um elemento `<div>` que serve para mostrar os ficheiros selecionados. Este elemento é escondido quando não existem ficheiros selecionados, e é mostrado quando existem. Tem a classe `ddfu_gallery` e é parente de todos os elementos que mostram os ficheiros selecionados.

Cada elemento `<div>`, que mostra um ficheiro selecionado, tem a classe `ddfu_file-preview` e é parente de todos os elementos que mostram as informações do ficheiro selecionado.

Dentro de `<div class="ddfu_file-preview">`, são apresentados uma imagem (um ficheiro SVG, para ser mais preciso) que representa a extensão do ficheiro, o nome, o tamanho e um botão para remover o ficheiro selecionado. Ao ser carregado, a lista de ficheiros

associada ao elemento `<input>` é atualizada e o elemento, que representa o ficheiro, é removido.

Os alertas são apresentados no topo do elemento `<div class="ddfu_drop-area">`, com a classe `ddfu_alert`. Estes alertas são removidos automaticamente após 5 segundos. Possuem cores diferentes, de acordo com o tipo de alerta.

Existem 4 diferentes cores registadas, onde podem ser observadas na Tabela 9.

Tabela 9 - Diferentes cores usadas para cada tipo de alerta, e respetivas representações.

Cor HEX	Tipo	Representação
#d89999	Erro	
#d8cc99	Aviso	
#a8d899	Sucesso	
#99a4d8	Informação	

Toda a área do elemento `<div class="ddfu_drop-area">` aceita o *drop*, exceto nos elementos filhos. Quando se arrasta um ficheiro para essa área, o contorno fica a vermelho, para indicar que é possível largar o ficheiro. Se o ficheiro for aceite, automaticamente a lista de ficheiros é atualizada e cria-se um elemento `<div class="ddfu_file-preview">` para mostrar o ficheiro selecionado. Se o ficheiro não for aceite, é apresentado um alerta a indicar que o ficheiro não é aceite. O erro irá depender da extensão, da existência de um ficheiro com o mesmo nome e do número máximo de ficheiros aceites.

O componente desenvolvido está ligado a uma rota de carregamento de ficheiros. O processo é iniciado depois de clicar no botão de submissão do formulário, em que, no início, os ficheiros são carregados e, de seguida, executa-se o processo normal de submissão de formulário. Cada registo da lista contém uma barra de progresso, que mostra o progresso do carregamento dos ficheiros. Quando o carregamento de um ficheiro é concluído, o registo é removido da lista.

Documentação API

Configurações disponíveis no ficheiro `DDFU_Configs.js`:

- **Classes CSS**: usado para alterar o valor das classes da biblioteca.
  - **Não recomendado editar, porque os estilos CSS nativos do componente ficam obsoletos.**

javascript

```
static #cssClasses = {
      dropArea: "ddfu_drop-area",
      gallery: "ddfu_gallery",
      filePreview: "ddfu_file-preview",
      counter: "ddfu_counter",
      // ...
};
```

- **Mensagens de alerta**: usado para alterar as mensagens de alerta da biblioteca. As mensagens podem ser alteradas, de acordo com o idioma do *website*. As cores também podem ser alteradas, mas é recomendado manter as cores padrão.

javascript

```
static #alertMessages = {
      component_not_found: "HTML element not found with the given ID",
      class_not_found: `Selector ${DDFU_Configs.cssClassesUsed.dropArea}
not found`,
      required_input: "File input is required",
      not_empty_input: "There is already a file in the input",
      zero_files_input: "Zero files in target input",
      // ...
};

static #alertMessagesColors = {
      error: "#d89999",
      warning: "#d8cc99",
      success: "#a8d899",
      information: "#99a4d8",
};
```

- **Cores nos ícones para extensão**: usado para alterar as cores dos ícones que representam as extensões dos ficheiros. As cores estão em formato hexadecimal, mas podem ser alteradas para RGB, HSL, etc., desde que esteja preparado para CSS. Também se pode adicionar mais extensões.

javascript

```
static #listExtensionsColors = {
      pdf: "#FFB6C1",
```

```
      doc: "#B0E0E6",
      txt: "#F0E68C",
      // ...
      other: "#C0C0C0",
};
```

- **Expressões utilizadas no componente**: usado para listar as expressões utilizadas no componente.

javascript

```
static #componentTexts = {
      progress: "Progress: ",
};
```

- **Localização dos ícones SVG**: usado para alterar o caminho dos ícones SVG. Os ícones estão guardados na pasta `assets`.

javascript

```
static #assetsPaths = {
      trashCan: {
            src: "../assets/Trash_Can.png",
            alt: "Trash can",
      },
      closeIcon: {
            src: "../assets/Cross.png",
            alt: "Close icon",
      },
};
```

Configurações (Tabela 10):

Tabela 10 - Configurações das localizações dos ícones SVG.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
src	Caminho do ícone	`string`	-	`src: "../assets/Trash_Can.png"`
alt	Texto alternativo do ícone	`string`	-	`alt: "Trash can"`

- **Configuração dos atributos do componente**: usado para listar os atributos para configuração do componente. São utilizados internamente, para apresentar erro ou aviso de atributos. Também estão descritos na secção Parâmetros e opções.
  - **Não recomendado editar, porque pode colocar o funcionamento da biblioteca em risco.**

javascript

```
static #componentAttributes = [
      {
            name: "ddfu-required",
            description: "Attribute to define if component is required
or not on form submission",
            required: false,
            error_message: "",
      },
      {
            name: "ddfu-action",
            description: "Attribute to define form action URL to upload
files",
            required: true,
            error_message:
DDFU_Configs.alertMessages.action_not_defined,
      },
];
```

Configurações (Tabela 11):

Tabela 11 - Configurações dos atributos do componente.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
name	Nome do atributo	string	-	name: "ddfu-required"
description	Descrição do atributo	String	-	description: "Attribute to define if component is required or not on form submission"
required	Indica se o atributo é obrigatório	boolean	-	required: false
error_message	Mensagem de erro, caso o atributo não seja definido	string	-	error_message: ""

- **Configurações gerais da biblioteca**: usado para configurar algumas propriedades da biblioteca, como ativar ou desativar alertas, ativar ou desativar ficheiros rejeitados, ativar ou desativar o modo de depuração, entre outros.

javascript

```
static #settings = {
      activateRejectedFiles: false,
      activateAlerts: true,
```

```
    alertDuration: 3,
    debuggerMode: true,
};
```

Configurações (Tabela 12):

Tabela 12 – Configurações gerais da biblioteca.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
activateRejectedFiles	Ativar ou desativar ficheiros rejeitados	boolean	false	activateRejectedFiles: true
activateAlerts	Ativar ou desativar alertas	boolean	true	activateAlerts: false
alertDuration	Duração dos alertas em segundos	number	3	alertDuration: 5
debuggerMode	Ativar ou desativar o modo de depuração	boolean	true	debuggerMode: false

Testes

A biblioteca foi testada em diferentes navegadores, como o Google Chrome, Mozilla Firefox, Microsoft Edge e Safari. A biblioteca também foi testada em diferentes dispositivos, como computadores, tablets e smartphones.

Contribuição

A biblioteca, neste momento, não aceita contribuições de terceiros. No entanto, o autor do projeto poderá aceitar sugestões ou correções de erros.

Para contribuir, contactar o autor do projeto.

Autores e reconhecimentos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Referências

Devido a restrições a dependências de terceiros, os ícones SVG estão guardados na pasta `assets`.

Histórico de lançamentos

Neste momento, o projeto encontra-se na versão 1.0.0.

Próximos passos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Neste momento, não existem planos para futuras atualizações.

Perguntas frequentes

Sem perguntas frequentes.

Suporte

Este projeto foi desenvolvido para a tese da unidade curricular de Projeto de Empresa, do curso de Mestrado em Engenharia Informática - Computação Móvel, do Instituto Politécnico de Leiria, no ano letivo de 2023/2024.

A intenção é aplicar este *plugin* numa plataforma de CMS, mas poderá ser utilizado em qualquer projeto.

Made with ❤️ by **Sandro Baptista**.

Anexo D – Manual de Utilizador do Componente de Gestão de Tabelas

DDTRS.js - Drag and Drop Table Rows Sort

DDTRS.js é uma biblioteca JavaScript que permite a reordenação de linhas de uma tabela HTML através de arrasto.

Introdução

A biblioteca DDTRS.js foi desenvolvida para facilitar a reordenação de linhas de uma tabela HTML através de arrasto. A biblioteca é leve e fácil de usar, e não requer a instalação de nenhuma dependência. Cada linha do `<tbody>` da tabela contém uma célula que permite o arrasto. Depois de largar a linha na posição desejada, a tabela é atualizada com a nova ordem das linhas.

A biblioteca é composta por alguns ficheiros necessários para o seu funcionamento:

- dois ficheiros JavaScript;
- um ficheiro CSS;
- um conjunto de ícones SVG (`assets`).

Instalação

A instalação baseia-se na adição de dois ficheiros ao projeto:

- ficheiro de código JavaScript;
- ficheiro de estilos CSS.

Pré-requisitos

Para instalar a biblioteca, não é necessário importar outras bibliotecas. O desenvolvimento da biblioteca foi feito com JavaScript puro, sem a necessidade de importar outras bibliotecas.

Passos de instalação

Para instalar a biblioteca, basta adicionar o ficheiro de código e o ficheiro de estilos, no local onde haja acesso às páginas Web. O ficheiro de configurações não é para ser importado. Depois adicionar o seguinte código ao ficheiro HTML:

html

```
<html>
    <head>
        <link rel="stylesheet" href="path/to/ddtrs.css"/>
    </head>
    <body>
        <script src="path/to/ddtrs.js"></script>
    </body>
</html>
```

A instalação é muito simples, visto que foi construído para ser utilizado em qualquer tipo de *website*.

Configuração

A biblioteca contém um ficheiro de configurações, denominado:

- `DDTRS_Configs.js`

Este ficheiro contém as configurações da biblioteca, que podem ser alteradas pelo utilizador.

As configurações disponíveis estão descritas na secção Documentação API.

Utilização

A utilização da biblioteca é muito simples, visto que foi construída para ser utilizada em qualquer tipo de website. A biblioteca foi construída para ser utilizada em formulários, mas pode ser utilizada em qualquer parte do website, desde que seja dentro do corpo da página.

Implementação

Para utilizar a biblioteca, basta adicionar o seguinte código ao ficheiro HTML:

html

```
<table class="(...) ddtrs_table">
      <thead>
            <tr>
                  (...)
                  <th>Order Drag</th>
                  (...)
            </tr>
      </thead>
      <tbody>
            (...)
            <tr data-id="(...)">
                  (...)
                  <td class="ddtr_drag">
```

```
                        <span class="icon-light">
                              <i class="fa-solid fa-grip-lines"></i>
                        </span>
                  </td>
                  (...)
            </tr>
            (...)
      </tbody>
</table>
```

Também é necessário adicionar o seguinte código JavaScript:

javascript

```
    const ddtrs = new DDTRS();
    ddtrs.setAlertsBoxLocation(document.querySelector('#id_do_ele-
mento_notificacoes'));
    ddtrs.init();
```

Toda esta estrutura é simples, porque apenas utiliza classes com nomes definidos, tanto no JavaScript como no CSS, para que o utilizador possa alterar o estilo do *website* sem que a biblioteca deixe de funcionar.

Para que haja comunicação entre os dados da tabela e a base de dados, é necessário adicionar o atributo `data-id` à linha da tabela. Este atributo deve conter o ID do registo da base de dados que está a ser representado na linha da tabela. O ID é necessário para que o *script* possa identificar o registo que está a ser arrastado, e assim poder atualizar a base de dados com a nova ordem das linhas.

Todo o conteúdo que esteja dentro do `<td class="ddtr_drag">` é opcional, podendo ser alterado ou removido. Refere-se ao ícone que aparece na célula que permite o arrasto da linha. O ícone foi deixado em HTML para que o utilizador possa alterar o seu estilo, se assim o desejar. O ícone utilizado é `fa-grip-lines` da biblioteca Font Awesome. Mesmo que se utilize ícones externos, não está associado ao código JavaScript e é algo que os programadores podem alterar, não sendo uma importação necessária para o funcionamento da biblioteca.

Também se encontra adaptada para o uso de `<tfoot>`. No entanto, não é obrigatório.

No JavaScript, apenas é necessário executar uma vez a função `init()` para que a biblioteca funcione corretamente em todas as tabelas com a classe `ddtrs_table` existentes na página Web.

Exemplos de uso

O seguinte exemplo demonstra como utilizar a biblioteca numa situação prática.

- HTML:

```html
<table id="t2" class="ddtrs_table">
      <thead>
            <tr>
                  <th>ID</th>
                  <th>Header 1</th>
                  <th>Header 2</th>
                  <th>Header 3</th>
                  <th>Order Drag</th>
            </tr>
      </thead>
      <tbody>
            <tr data-id="1">
                  <td>1</td>
                  <td>Data 1</td>
                  <td>Data 2</td>
                  <td class="header3">Data 3</td>
                  <td class="ddtrs_drag">
                        <span class="icon-light">
                              <i class="fa-solid fa-grip-lines"></i>
                        </span>
                  </td>
            </tr>
            <tr data-id="2">
                  <td>2</td>
                  <td>Data 1</td>
                  <td>Data 2</td>
                  <td class="header3">Data 3</td>
                  <td class="ddtrs_drag">
                        <span class="icon-light">
                              <i class="fa-solid fa-grip-lines"></i>
                        </span>
                  </td>
            </tr>
      </tbody>
</table>
<div class="notifications-center"></div>
```

- JavaScript:

```javascript
      const ddtrs = new DDTRS();
      ddtrs.setAlertsBoxLocation(document.querySelector('.notifications-
center'));
      ddtrs.init();
```

Neste exemplo, o utilizador tem uma tabela com duas linhas, onde cada linha contém um identificador, três colunas de dados e uma coluna para arrastar a linha. O utilizador pode arrastar as linhas para reordenar a tabela. O utilizador também tem um elemento HTML com a classe `notifications-center` para mostrar notificações.

Parâmetros e opções

Atributos para as linhas `<tr>` do `<tbody>` (Tabela 13, Tabela 14):

Tabela 13 - Atributos para as linhas `<tr>` do `<tbody>` (parte 1).

Atributo	Descrição	Obrigatório
`data-id`	Identificador da linha, usado para definir valor de ordenação	Sim

Tabela 14 - Atributos para as linhas `<tr>` do `<tbody>` (parte 2).

Atributo	Tipo	Valor padrão	Exemplos
`data-id`	`string`	`''`	`data-id="1"` `data-id="abcd123"`

Casos de uso

Um caso de uso está representado no exemplo de utilização da biblioteca. O utilizador tem uma tabela com várias linhas, onde cada linha contém um identificador, três colunas de dados e uma coluna para arrastar a linha. O utilizador pode arrastar as linhas para reordenar a tabela. O utilizador também tem um elemento HTML para mostrar notificações. O funcionamento é sempre igual, independentemente do conteúdo da tabela.

Funcionalidades

Ao carregar a tabela com as funcionalidades desta biblioteca, consegue-se observar a coluna, que permite o arrasto, na posição que o programador desejou. Para que seja possível alterar a posição, basta apenas pressionar um botão do rato (qualquer botão do rato permite iniciar o arrasto) na área da célula, arrastar no sentido vertical e largar o botão no local onde se pretende colocar.

Imediatamente após o início do arrasto, o ponteiro do rato move-se para o centro da célula, mantendo a posição X inicial. Este ponto é importante, visto que é o ponteiro do rato que irá delimitar a zona de arrasto das linhas. Ou seja, quando o ponteiro interseta o limite

do `<tbody>`, automaticamente a linha é "barrada", mas o ponteiro pode ultrapassar sem problemas. Se o botão se mantiver pressionado e o ponteiro voltar à área da tabela, a linha ainda está a ser arrastada; se largar fora da área, a linha fica na posição mais próxima do limite (no topo ou no fundo). O ponteiro, ao longo do arrasto em `<tbody>`, pode-se deslocar em toda a área da linha, e, se largar o botão fora da área da célula para arrasto, a troca de posição funciona com sucesso, porque o que interessa apenas é a posição em Y do ponteiro.

A posição Y é essencial, porque a estrutura da tabela assim a pede. Não faz sentido permitir o arrasto horizontalmente, visto que a tabela é uma coluna com múltiplas linhas. Portanto, o arrasto funciona no eixo do Y.

Quando a linha está a ser arrastada e a aproximar-se do limite, esta tem a tendência em ultrapassá-lo até metade da sua altura. Isto acontece, porque, como há dependência das coordenadas do ponteiro e como o ponteiro está no centro da linha verticalmente, a troca de linha ocorre quando o ponteiro percorre mais de metade da área da linha com a posição desejada, tanto quando vai para cima como para baixo. Durante o processo, a que fica "por debaixo" da linha a arrastar passa para a posição onde a outra estava. No caso em que se pretende passar da última posição para a primeira, a linha vai passando por todas as posições, realizando trocas com as existentes.

O processo de troca de posições depende de uma tabela clone criada pelo script. A tabela original nunca é afetada durante o arrasto e apenas sofre atualização depois de largar a linha na posição desejada. Como se cria uma tabela temporária, existe a possibilidade em escolher as colunas a apresentar, porque serve somente para o arrasto.

Após o arrasto da linha na tabela, vários alertas são gerados por cima da tabela, para ter uma noção do processo da atualização no navegador e na base de dados. Nos elementos que representam os alertas, existem 4 diferentes cores registadas, onde podem ser observadas na Tabela 15.

Tabela 15 - Diferentes cores usadas para cada tipo de alerta, e respetivas representações.

Cor HEX	Tipo	Representação
#d89999	Erro	
#d8cc99	Aviso	

#a8d899	Sucesso	
#99a4d8	Informação	

Documentação API

Configurações disponíveis no ficheiro `DDTRS_Configs.js`:

- **Classes CSS**: usado para alterar o valor das classes da biblioteca.
  - **Não recomendado editar, porque os estilos CSS nativos do componente ficam obsoletos.**

javascript

```
      static #selectorsUsed = {
            table: ".ddtrs_table",
            dragging: ".ddtrs_dragging",
            placeholder: ".ddtrs_placeholder",
            cloneTable: ".ddtrs_clone-table",
            // ...
      };
```

- **Seletores CSS excluídos do clone**: usado para indicar os elementos HTML que não devem ser clonados. Estes elementos são excluídos do clone da tabela. Este pormenor está associado à possibilidade em mostrar as colunas essenciais durante o arrasto das linhas.

javascript

```
      static #excludedSelectors = [
            "table-actions",
            "show-details-row",
            "more-details-row",
            // ...
      ];
```

- **Mensagens de alerta**: usado para alterar as mensagens de alerta da biblioteca. As mensagens podem ser alteradas, de acordo com o idioma do *website*. As cores também podem ser alteradas, mas é recomendado manter as cores padrão.

javascript

```
      static #alertMessages = {
            error: "Error when updating table",
            error_network: "Network response was not ok",
            error_component_not_found: `HTML element not found with the
${DDTRS_Configs.selectorsUsed.table} class`,
```

```
            information: "Updating table...",
            // ...
      };

      static #alertMessagesColors = {
            error: "#d89999",
            warning: "#d8cc99",
            success: "#a8d899",
            information: "#99a4d8",
            // ...
      };
```

- **Localização dos ícones SVG**: usado para alterar o caminho dos ícones SVG. Os ícones estão guardados na pasta `assets`.

javascript

```
      static #assetsPaths = {
            dragIcon: {
                  src: "../assets/Drag_Reorder.png",
                  alt: "Trash can",
            },
            closeIcon: {
                  src: "../assets/Cross.png",
                  alt: "Close icon",
            },
      };
```

Configurações (Tabela 16):

Tabela 16 - Configurações de localização dos ícones SVG.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
src	Caminho do ícone	string	""	"../assets/Drag_Re-order.png"
alt	Texto alternativo do ícone	string	""	"Trash can"

- **Configurações gerais da biblioteca**: usado para configurar algumas propriedades da biblioteca.

javascript

```
      static #scriptSettings = {
            debuggerMode: true,
            alertsConfig: {
                  boxLocation: null,
                  duration: 3, // seconds
            },
            network_config: {
```

```
                    route: `${window.location.href}/some-route-to-update-
table-order`,
                    mimeType: "application/json",
                    csrfToken:
                          document.querySelector("meta[name='csrf-to-
ken']").content || "",
                    method: "POST",
                    activate: false,
                },
        };
```

Configurações (Tabela 17):

Tabela 17 - Configurações gerais da biblioteca.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
debuggerMode	Ativa o modo de depuração	boo-lean	false	true
alertsConfig	Configurações de alertas	ob-ject	{}	Ver as configurações abaixo
network_con-fig	Configurações de rede	ob-ject	{}	Ver as configurações abaixo

Configurações de alertsConfig (Tabela 18):

Tabela 18 - Configurações de alertsConfig.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
boxLocation	Localização do elemento de no-tificações	HTMLE-lement	null	document.querySelec-tor('.notifications-center')
duration	Duração do alerta	number	3	5

Configurações de network_config (Tabela 19):

Tabela 19 - Configurações de network_config.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
route	Rota para atualizar a ordem da tabela	string	""	"https://www.exam-ple.com/some-route-to-update-table-order"

mimeType	Tipo de conteúdo	string	"applica-tion/json"	"application/xml"
csrfToken	Token CSRF	string	""	document.querySelec-tor("meta[name='csrf-token']").content
method	Método HTTP	string	"POST"	"GET"
activate	Ativa a atu-alização da tabela	boo-lean	false	true

Testes

A biblioteca foi testada em diferentes navegadores, como o Google Chrome, Mozilla Firefox, Microsoft Edge e Safari. A biblioteca também foi testada em diferentes dispositivos, como computadores, tablets e smartphones.

Contribuição

A biblioteca, neste momento, não aceita contribuições de terceiros. No entanto, o autor do projeto poderá aceitar sugestões ou correções de erros.

Para contribuir, contactar o autor do projeto.

Autores e reconhecimentos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Referências

Devido a restrições a dependências de terceiros, os ícones SVG estão guardados na pasta `assets`.

Histórico de lançamentos

Neste momento, o projeto encontra-se na versão 1.0.0.

Próximos passos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Neste momento, não existem planos para futuras atualizações.

Perguntas frequentes

Sem perguntas frequentes.

Suporte

Este projeto foi desenvolvido para a tese da unidade curricular de Projeto de Empresa, do curso de Mestrado em Engenharia Informática - Computação Móvel, do Instituto Politécnico de Leiria, no ano letivo de 2023/2024.

A intenção é aplicar este *plugin* numa plataforma de CMS, mas poderá ser utilizado em qualquer projeto.

Made with ❤️ by **Sandro Baptista**.

Anexo E – Manual de Utilizador do Editor de páginas Web - GrapesJS

Laravel-GrapesJS - Manual de utilizador

Este manual de utilizador pretende explicar como utilizar o Laravel-GrapesJS, um pacote Laravel que permite a utilização do GrapesJS, um editor de páginas Web *open-source*. Além disso, explica como criar componentes personalizados para o GrapesJS, utilizando as ferramentas Laravel.

Introdução

O Laravel-GrapesJS é um pacote Laravel que permite a utilização do GrapesJS, um editor de páginas Web *open-source*. Este pacote foi desenvolvido para facilitar a integração do GrapesJS com o Laravel, permitindo a criação de páginas Web de forma rápida e fácil.

O objetivo deste manual é explicar como utilizar o Laravel-GrapesJS, desde a utilização do pacote até à criação de componentes personalizados para o GrapesJS. Este manual está dividido em várias secções, cada uma dedicada a um aspeto específico do Laravel-GrapesJS.

Instalação

O pacote já se encontra instalado no EasyWeb.

Pré-requisitos

Sem pré-requisitos.

Passos de instalação

Sem passos de instalação.

Configuração

Sem configuração.

Utilização

O acesso ao editor é feito a partir do registo de uma página no EasyWeb.

O editor está estruturado na seguinte forma, conforme está na Figura 82:

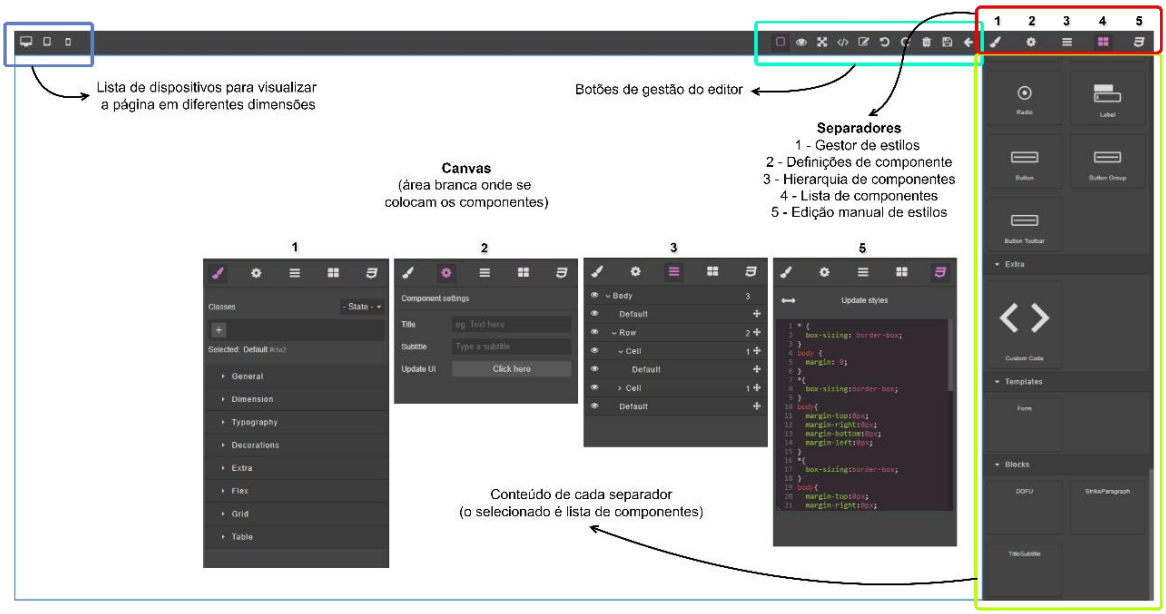


Figura 82 – Composição do editor GrapesJS.

No topo do editor, temos a barra de ferramentas, onde se pode encontrar várias opções relacionadas com a gestão das edições feitas. No lado esquerdo, está a área de trabalho, onde se pode arrastar e soltar os componentes para a página. No lado direito, está a área de configuração, onde se pode configurar os componentes selecionados. A lista dos componentes está nessa secção e pode-se arrastar e soltar para a área de trabalho.

Implementação de componentes personalizados

Para criar um componente ou um *template*, é necessário realizar diversos passos:

1. Executar comandos PHP Artisan para criar o componente ou *template*. `php artisan make:component LaravelGrapesjs/Blocks/{component-name}` para criar um componente. `php artisan make:component LaravelGrapesjs/Templates/{template-name}` para criar um *template*.

   Assim, um ficheiro Blade fica criado na pasta `resources/views/components/laravel-grapesjs/blocks/{component-name}.blade.php` ou em `resources/views/components/laravel-grapesjs/templates/{template-name}.blade.php`.

   Também é criado um ficheiro PHP controlador na pasta `app/View/Components/LaravelGrapesjs/Blocks/{ComponentName}.php` ou em `app/View/Components/LaravelGrapesjs/Templates/{TemplateName}.php`.

2. Aplicar alterações no ficheiro PHP controlador. O ficheiro PHP controlador deve ter a seguinte estrutura:

php

```
class ComponentName extends Component
{
  (...) // variáveis que podem ser utilizadas em traits, por exemplo

  public function __construct(array $data = [])
  {
        (...) // atribuição de valores às variáveis criadas
  }

  (...) // secção de funções onde se acede a valores default das va-
riáveis

  public function getTraits()
  {
        return $this->configs()['traits'];
  }

  public function getTypeName()
  {
        return $this->configs()['type'];
  }

  public function getLinksCssJs()
  {
        return $this->configs()['links_css_js'];
  }

  public function render(): View|Closure|string
  {
        (...) // tratamento das variáveis antes de serem enviadas
para a view

        return view('components.laravel-grapesjs.blocks.component-
name')
              ->(...);
  }

  public function configs()
  {
        return [
              'type' => 'component-name',
              'traits' => [...], // configurar os traits
              'default_values' => [...], // valores default das vari-
áveis
              'links_css_js' => [
                    'css' => [
                          [
                                'href' => (...),
```

```
                                'rel' => (...),
                            ], // função que chama a biblioteca - esti-
    los
                            (...) // mais estilos
                        ],
                        'js' => [
                            [
                                'src' => (...),
                                'type' => (...),
                            ], // função que chama a biblioteca
                            (...) // mais funções
                        ],
                ],
            ];
      }
    }
```

3. Aplicar alterações no ficheiro Blade. O ficheiro Blade deve ter a seguinte estrutura:

php

```
@templateIcon('path-to-icon')

(...) // HTML do componente + uso de variáveis vindas do controlador
```

Exemplos de uso

O seguinte exemplo demonstra como criar um componente personalizado para o GrapesJS.

- Comando PHP Artisan para criar o componente: `php artisan make:component LaravelGrapesjs/Blocks/DDFU;`
- Ficheiro PHP controlador: `app/View/Components/LaravelGrapesjs/Blocks/DDFU.php;`
- Ficheiro Blade: `resources/views/components/laravel-grapesjs/blocks/ddfu.blade.php;`
- Configuração do ficheiro PHP controlador:

php

```
 class DDFU extends Component
 {
  public $label;
  public $name;
  public $accept;
  public $multiple;
```

```
    public $required;

    /**
    * Create a new component instance.
    */
    public function __construct(array $data = [])
    {
          $this->label = $data['label'] ?? $this->getDefaultLabel();
          $this->name = $data['name'] ?? $this->getDefaultName();
          $this->accept = $data['accept'] ?? $this->getDe-
faultAccept();
          $this->multiple = $data['multiple'] ?? $this->getDefaultMul-
tiple();
          $this->required = $data['required'] ?? $this->getDe-
faultRequired();
    }

    private function getDefaultLabel()
    {
          return $this->configs()['default_values']['label'];
    }

    private function getDefaultName()
    {
          return $this->configs()['default_values']['name'];
    }

    private function getDefaultAccept()
    {
          return $this->configs()['default_values']['accept'];
    }

    private function getDefaultMultiple()
    {
          return $this->configs()['default_values']['multiple'];
    }

    private function getDefaultRequired()
    {
          return $this->configs()['default_values']['required'];
    }

    public function getTraits()
    {
          return $this->configs()['traits'];
    }

    public function getTypeName()
    {
          return $this->configs()['type'];
    }

    public function getLinksCssJs()
    {
```

```
            return $this->configs()['links_css_js'];
      }

      /**
      * Get the view / contents that represent the component.
      */
      public function render(): View|Closure|string
      {
            $this->label = empty($this->label) ? $this->getDefaultLa-
bel() : $this->label;
            $this->name = empty($this->name) ? $this->getDefaultName() :
$this->name;
            $this->accept = empty($this->accept) ? $this->getDefaultAc-
cept() : $this->accept;
            $this->multiple = empty($this->multiple) ? $this->getDefau-
ltMultiple() : $this->multiple;
            $this->required = empty($this->required) ? $this->getDefau-
ltRequired() : $this->required;

            return view('components.laravel-grapesjs.blocks.d-d-f-u')
                  ->with('label', $this->label)
                  ->with('name', $this->name)
                  ->with('accept', $this->accept)
                  ->with('multiple', $this->multiple)
                  ->with('required', $this->required);
      }

      public function configs()
      {
            return [
                  'type' => 'd-d-f-u',
                  'traits' => [
                        [
                              'label' => 'Label',
                              'type' => 'text',
                              'name' => 'label',
                              'category' => [
                                    'id' => 'basic',
                                    'label' => 'Basic',
                                    'open' => false,
                              ],
                        ],
                        [
                              'label' => 'Name',
                              'type' => 'text',
                              'name' => 'name',
                              'category' => [
                                    'id' => 'basic',
                                    'label' => 'Basic',
                                    'open' => false,
                              ],
                        ],
                        [
                              'label' => 'Accept',
```

```
                              'type' => 'text',
                              'name' => 'accept',
                              'category' => [
                                    'id' => 'basic',
                                    'label' => 'Basic',
                                    'open' => false,
                              ],
                        ],
                        [
                              'label' => 'Multiple',
                              'type' => 'checkbox',
                              'name' => 'multiple',
                              'category' => [
                                    'id' => 'basic',
                                    'label' => 'Basic',
                                    'open' => false,
                              ],
                        ],
                        [
                              'label' => 'Required',
                              'type' => 'checkbox',
                              'name' => 'required',
                              'category' => [
                                    'id' => 'basic',
                                    'label' => 'Basic',
                                    'open' => false,
                              ],
                        ],
                  ],
                  'default_values' => [
                        'label' => 'File',
                        'name' => 'file',
                        'accept' => 'image/*',
                        'multiple' => false,
                        'required' => false,
                  ],
                  'links_css_js' => [
                        'css' => [
                              [
                                    'href' => '/plugins/js-drag-drop-
files-upload/css/drag-drop-upload-files.css',
                                    'rel' => 'stylesheet'
                              ], // função que chama a biblioteca - esti-
los
                        ],
                        'js' => [
                              [
                                    'src' => '/plugins/js-drag-drop-fi-
les-upload/js/DDFU.js',
                                    'type' => 'module'
                              ], // função que chama a biblioteca
                              [
                                    'src' => '/js/index2.js',
                                    'type' => 'module'
```

```
                            ] // função que inicializa a biblioteca
                        ],
                    ],
                ];
    }
}
```

- Configuração do ficheiro Blade:

php

```
@templateIcon('images/logo.jpg')

<label>{{ $label }}</label>
<div class="ddfu_drop-area" id="upload-cv" ddfu-action="{{
route('uploadFile') }}" ddfu-required>
  <p>Drop your CV file in this area or</p>
  <input type="file" name="{{ $name }}[]" id="file1" title="CV File"
accept="{{ $accept }}"
       @if ($required) required @endif @if ($multiple) multiple
@endif>
  <label for="file1" class="ddfu_button">Select CV File</label>
</div>
```

Parâmetros e opções

Controlador do componente:

- `getTraits()`: retorna os *traits* do componente;
- `getTypeName()`: retorna o nome do componente;
- `getLinksCssJs()`: retorna os links CSS e JavaScript do componente;
- `configs()`: retorna a configuração do componente.

Configuração do `configs()` (Tabela 20):

Tabela 20 - Configuração do `configs()`.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
`type`	Nome do componente	`string`	`null`	`'d-d-f-u'`
`traits`	*Traits* do componente	`array`	`[]`	Ver as configurações abaixo
`default_values`	Valores *default* das variáveis	`array`	`[]`	Ver as configurações abaixo

links_css_js	Links CSS e JavaScript do componente	array	[]	Ver as configurações abaixo

Configuração do configs() traits: cada registo está guardado no manual do GrapesJS [58].

Configuração do configs() default_values: cada registo é uma variável do componente. Cada uma tem um valor por omissão. Exemplo:

php

```
'default_values' => [
      'label' => 'File',
      'name' => 'file',
      'accept' => 'image/*',
      'multiple' => false,
      'required' => false,
],
```

Configuração do configs() links_css_js (Tabela 21):

Tabela 21 - Configuração do configs() links_css_js.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
css	Links CSS do componente	array	[]	Ver as configurações abaixo
js	Links JavaScript do componente	array	[]	Ver as configurações abaixo

Configuração do configs() links_css_js css (Tabela 22):

Tabela 22 – Configuração do configs() links_css_js css.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
href	URL do ficheiro CSS	string	null	'https://www.example.com/file.css'
rel	Relação do ficheiro CSS	string	null	'stylesheet' preload prefetch

Configuração do configs() links_css_js js (Tabela 23):

Tabela 23 – Configuração do `configs()` `links_css_js` `js`.

Propriedade	Descrição	Tipo	Valor Padrão	Exemplo
`src`	URL do ficheiro JavaScript	`string`	`null`	`'https://www.example.com/file.js'`
`type`	Tipo do ficheiro JavaScript	`string`	`null`	`'module' text/javascript`

Casos de uso

Sem casos de uso.

Funcionalidades

O editor GrapesJS permite a criação de páginas web de forma rápida e fácil. O editor tem várias funcionalidades, como a possibilidade de adicionar componentes, editar o conteúdo, alterar o estilo, entre outras.

Para utilizar o editor, basta arrastar e soltar os componentes para a área de construção à esquerda. Os componentes podem ser personalizados, de acordo com as necessidades do utilizador, na aba "Component Settings".

Para mais informações sobre o editor GrapesJS, consultar a documentação oficial [46]. Com a documentação, é possível aplicar várias configurações, tanto em funcionalidades como na aparência do editor.

Cada componente tem as seguintes funcionalidades:

- *Traits*: são propriedades que podem ser configuradas no componente, durante a construção da página. Estas propriedades podem ser configuradas no componente, para que o utilizador possa alterar o estilo ou o comportamento do componente.
- *Ficheiros CSS e JavaScript*: são ficheiros que são carregados no componente. Estes ficheiros podem ser estilos ou funções que são necessárias para o funcionamento do componente.

Documentação API

Funcionamento do editor: GrapesJS [46].

Documentação do pacote Laravel-GrapesJS: Laravel-GrapesJS [47].

Criação de componentes personalizados: ver a secção Implementação de componentes personalizados.

Testes

A biblioteca foi testada em diferentes navegadores, como o Google Chrome, Mozilla Firefox, Microsoft Edge e Safari. A biblioteca também foi testada em diferentes dispositivos, como computadores, tablets e smartphones.

Contribuição

A biblioteca, neste momento, não aceita contribuições de terceiros. No entanto, o autor do projeto poderá aceitar sugestões ou correções de erros.

Para contribuir, contactar o autor do projeto.

Autores e reconhecimentos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Referências

GrapesJS [45] - o editor de páginas Web *open-source*.

Laravel-GrapesJS [47] - o pacote Laravel que permite a utilização do GrapesJS.

Histórico de lançamentos

Neste momento, o projeto Laravel-GrapesJS encontra-se na versão 3.4.1 (maio de 2023) no GitHub [47]. Em GrapesJS, a versão é 0.21.13 (agosto de 2024) [46].

Sobre as melhorias, a versão encontra-se em 1.0.0.

Próximos passos

No caso de serem encontrados erros ou se for necessário adicionar novas funcionalidades, este projeto poderá ser atualizado.

Sobre a documentação, esta pode também ser atualizada, para que seja mais fácil de entender o funcionamento do projeto.

Em caso de dúvidas, sugestões ou erros, contactar o autor do projeto.

Próximas atualizações:

- [ ] Aplicação de idiomas para os componentes e para a página.

Perguntas frequentes

Sem perguntas frequentes.

Suporte

Este projeto foi desenvolvido para a tese da unidade curricular de Projeto de Empresa, do curso de Mestrado em Engenharia Informática - Computação Móvel, do Instituto Politécnico de Leiria, no ano letivo de 2023/2024.

A intenção é aplicar este *plugin* numa plataforma de CMS, mas poderá ser utilizado em qualquer projeto.

Made with ❤️ by **Sandro Baptista**.