Instituto Superior de Tecnologias Avançadas

Licenciatura em Informática

CUSTOM CONTENT MANAGEMENT SYSTEM

Daniel José de Matos Dias Nº.1377

Coordenador – Professor Dr. Pedro Brandão

Orientadora: Professora Dra. Dulce Mourato

Lisboa, Ano letivo 2022/2023

Instituto Superior de Tecnologias Avançadas

Licenciatura em Informática

CUSTOM CONTENT MANAGEMENT SYSTEM

Daniel José de Matos Dias Nº.1377

Coordenador – Professor Dr. Pedro Brandão

Orientadora: Professora Dra. Dulce Mourato

Lisboa, Ano Letivo 2022/2023

RESUMO

Os Sistemas de Gestão de Conteúdo ou CMS têm uma vasta aplicação, tanto a nível pessoal como empresarial, uma vez que simplificam a criação e gestão de *websites.* Através do desenvolvimento de interfaces simples e intuitivas, os *websites* começaram a ser também geridos por utilizadores menos experientes. Desta utilização, surgiram novas dificuldades e lacunas no desenvolvimento dos CMS.

Neste projeto, as dificuldades enfrentadas pelos utilizadores foram identificadas e foi feita uma análise da evolução dos CMS, desde os primeiros sistemas desenvolvidos aos sistemas mais atuais. Deste modo, foi possível identificar as principais alterações aos sistemas CMS, a partir das quais esta evolução ocorreu.

O objetivo deste projeto consistiu na identificação destas lacunas, que foram reportadas pelos clientes, e o posterior desenvolvimento de um novo sistema CMS, visando colmatar as dificuldades apresentadas nos sistemas previamente criados.

PALAVRAS-CHAVE: CMS, Gestão, Interfaces, *Website*.

ABSTRACT

Content Management Systems or CMSs have a wide range of applications, from personal to corporate websites, mainly because they simplify website creation and management. Through the development of simple and intuitive interfaces, websites began to be managed by less experienced users as well. As a result, new difficulties and gaps in CMS development emerged.

In this project, the difficulties faced by users were identified and the evolution of CMSs was analyzed, by studying the differences between the first systems developed and the most current ones. Consequently, the main changes in CMS systems, from which this evolution occurred, were identified.

The goal of this project was to identify these gaps, which were reported by customers, and to develop a new CMS system, within the scope of overcoming the difficulties reported in previously created systems.

KEYWORDS: CMS, Management, Interfaces, *Website*.

ABREVIATURAS

WYSIWYG – What You See Is What You Get

CMS - Content management system

API - Application Programming Interface

COPE - Create Once, Publish Everywhere.

CRUD - Create, Read, Update and Delete.

ÍNDICE

1. INTRODUÇÃO 10

2. REVISÃO DE LITERATURA 12

2.1 Principais características que um CMS necessita de ter 19

2.2 Arquitetura do CMS 20

2.3 Principais diferenças entre um Traditional CMS e um Headless CMS 22

2.4 Problemas de segurança nos diferentes tipos de CMS 22

2.5 Diferença entre Sistema CMS opensource e closedsource 23

3. MATERIAIS E MÉTODOS DE DESENVOLVIMENTO 24

4. RESULTADOS E DISCUSSÃO DE RESULTADOS 27

4.1 Principais problemas de segurança mais comuns num CMS 27

4.2 Como melhorar a segurança de qualquer CMS 28

4.3 Autenticação 29

4.4 Tecnologias utilizadas 30

4.5 Lista de requisitos 30

4.6 Dashboard 31

4.7 Tipos de dados permitidos 32

4.8 Tipos de campos suportados pelo o CMS 34

4.9 Adaptação do frontend ao tipo de dados. 35

5. MÓDULOS DO PROTÓTIPO 39

5.1 Tipos e funcionalidades dos módulos do CMS 39

CONCLUSÃO 46

BIBLIOGRAFIA 48

ÍNDICE DE FIGURAS

Figura 1 – Traditional CMS vs Headless CMS.....21

Figura 2 – Fluxo de acesso ao Sistema CMS e suas funcionalidades 25

Figura 3 – Página de Login do CMS.....30

Figura 4 – Página de Login do CMS.....31

Figura 5 – Dashboard do Custom CMS.....31

Figura 6 – Lista de permissões do Custom CMS.....34

Figura 7 – Editor de ficheiros JSON 34

Figura 8 – Transformação de tabelas SQL em campos de texto.....35

Figura 9 – Componente Datepicker.....37

Figura 10 – Componente textarea simples 37

Figura 11 – Componente texarea transformado em editor com formatações.....38

Figura 12 – Transformação campo da tabela em componente 38

Figura 13 – Explorador de imagens 39

Figura 14 – Módulo JSON do Custom CMS.....41

Figura 15 – Estrutura da tabela SQL.....41

Figura 16 – Painel de edição da coluna selecionada. 42

Figura 17 – Módulo de envio de email 43

Figura 18 – Editor de texto com formatações simples e completas 43

Figura 19 – Módulo criador de conteudos.....44

Figura 20 – Funcionalidade do editor de ficheiros 45

Figura 21 – Editor de ficheiros.....45

Figura 22 – Editor de ficheiros de texto e de código fonte 46

Figura 23 – Painel de logs 47

ÍNDICE DE TABELAS

Tabela 1 – Diferenças entre Tradicional CMS e Headless CMS…….....23

Tabela 2 – Lista de Permissões do Custom CMS ...……..33

Tabela 3 – Tipos de campos suportados no CMS .....…….....36

1. INTRODUÇÃO

No mundo digital, cada vez mais existe a noção de que cada *website* é único e com a evolução das tecnologias mais customizadas às necessidades do cliente e do negócio, tem surgido uma necessidade de construir um sistema que permita a instalação num servidor, que consiga ler automaticamente o esquema de um *datasource* e construa um *frontend* preparado para executar as operações ***Create, Read, Update and Delete*** (CRUD).

O projeto e protótipo apresentados surgiram da necessidade de se desenvolver internamente na empresa, um sistema que fosse dinâmico e que se adaptasse facilmente a um conjunto de dados, independentemente de serem ficheiros JSON/JS ou tabelas SQL. Este sistema é um Custom Content Management System (CMS), no qual é feita a parametrização da estrutura dos tipos de dados (ficheiros JSON ou tabelas SQL) e a mesma é interpretada pelo *website*, que fornece todos os objetos e funcionalidades para permitir a edição de dados, o envio de *emails* e de ficheiros para o servidor, bem como a criação de conteúdos (Oracle, 2022).

Com o objetivo de diminuir o tempo e os custos do desenvolvimento, este sistema foi desenvolvido para fornecer ao cliente um conjunto de ferramentas, que permitissem atualizar o seu *website* sem requerer uma formação ou treino específico. Atualmente, existe um elevado número de CMS disponíveis, tanto *opensource* como *closedsource* (Yermolenko & Golchevskiy, 2021), mas a sua utilização pode ser complexa e desafiante. Por esta razão, foi desenvolvido este protótipo com o propósito de simplificar a gestão e atualização do *website*, que é feita de forma autónoma pelo cliente.

O objetivo desta investigação foi identificar quais as falhas e dificuldades que os utilizadores comuns enfrentam ao utilizar alguns dos CMS mais populares (Baptista, 2014), com o intuito de desenvolver um protótipo baseado nas tecnologias mais modernas e com uma interface intuitiva e simples, de modo a promover uma melhor utilização do sistema e domínio das suas funcionalidades por parte do utilizador. A investigação teve ainda como objetivos específicos os seguintes: a) pesquisar as falhas e problemas dos principais CMS existentes atualmente; b) desenvolver um protótipo para colmatar estes problemas; c) criar um conjunto de funcionalidades que sejam comuns e úteis para os utilizadores que atualizam conteúdos num *website*.

Este protótipo foi desenvolvido com as mais recentes tecnologias, como javascript e CSS3, estas tecnologias permitem criar uma interface gráfico *user-friendly* e *responsive* (Multi ecrãs).

Sendo um projeto inteiramente alojado na Cloud, é importantíssimo consegue aceder em qualquer dispositivo, seja um computador de trabalho, um tablet ou até o telemóvel pessoal, assim os ecrãs do CMS devem conseguir adaptar-se a qualquer ecrã. O utilizador irá conseguir ter acesso ao sistema precisando apenas de um dispositivo e ‘internet’ disponível, esta autonomia faz com que um website esteja constantemente a ser atualizado, muito útil para lojas ‘online’ onde é importante gerir stock e ter o produto atualizados.

2. REVISÃO DE LITERATURA

Esta secção é dedicada à revisão de literatura atualizada dos sistemas CMS, tendo como objetivo a apresentação das principais características destes sistemas e a identificação de lacunas no desenvolvimento dos mesmos. De acordo com o problema identificado, desenvolveu-se a seguinte questão de investigação, quais as principais funcionalidades que um sistema CMS deve conter para ser mais fácil e intuitivo atualizar qualquer *website*?.

Para lhe dar resposta, foi efetuada comparação e análise de vários sistemas CMS que englobam os mais utilizados (Baptista, 2014). Os sistemas CMS pesquisados foram comparados de acordo com os seguintes critérios:

1. Estrutura do sistema;
2. Componente tecnológica;
3. Adição de extensões;
4. Modelo de gestão do *website*.

Um dos primeiros sistemas CMS desenvolvidos foi o Vignette, que foi lançado em 1995, com o objetivo de simplificar a publicação de conteúdo no *website* e a sua gestão. Este sistema é baseado nas linguagens JavaEE e Microsoft .NET, enquanto que a Base de Dados (BD) é suportada por IBM DB2. No Vignette, a gestão é feita através das suas duas aplicações – a de manutenção de conteúdo, um elemento *back-office* que permite a criação, edição e localização de conteúdo pelos utilizadores; e o portal que permite a publicação de conteúdo no *website.* Para além destas aplicações, o Vignette retém a possibilidade de serem adicionadas extensões. Neste sistema, a gestão é realizada em torno do conteúdo do *website* (Open Text Corporation, 2023; Saraiva, 2013).

O sistema TYPO3 teve a sua primeira versão lançada em 2000, e é maioritariamente usado para a criação de *websites* de grandes empresas com suporte internacional. Este sistema é baseado na linguagem PHP, o suporte de base de dados pode ser fornecido por MySQL, PostGreSQL, Oracle ou Microsoft SQL Server, enquanto que a aplicação do servidor *web* pode ser Apache ou Microsoft IIS. O TYPO3 é constituído pelo cerne do sistema e a API de extensão. A partir desta segunda componente, podem ser adicionados vários tipos de funcionalidades e extensões que serão conectados ao cerne. Já este, tem como funções o controlo de permissões de autenticação e verificações durante a instalação, a construção do interface de apresentação, bem como a

conectividade e iniciação de bases de dados. Neste sistema, a gestão é feita em torno do conteúdo do *website* e a mesma é feita através de uma página de administração *back-end* (Do Carmo, 2007; TYPO3 Association, 2023).

O sistema Drupal foi lançado em 2001 e fornece uma estrutura back-end que permite não só criar e organizar conteúdo, manipular a aparência e automatizar tarefas administrativas, como também definir permissões e papéis para usuários e colaboradores. Este CMS tem como principal objetivo criar conteúdo, pelo que uma única instalação pode servir como um *website* simples, um multi- ou single-blog, um fórum ou até um community *website* para usuários gerarem conteúdo. Drupal é apropriado tanto para indivíduos, negócios e organizações, independentemente do tamanho destas, uma vez que se trata de um sistema muito flexível que oferece um vasto número de features, incluindo conteúdo de modelação avançado, uma boa variedade de temas e uma grande biblioteca de módulos de extensão.

O sistema Drupal foi escrito em PHP e pode correr em qualquer servidor que corra PHP - como o Microsoft IIS e o Apache - e gere o conteúdo segundo uma abordagem content-centric, que significa que o conteúdo é o foco em detrimento do aspecto de onde este vai ser colocado. A gestão do Drupal é feita através de 3 camadas bem definidas que idealmente interagem entre elas por meio de APIs e interfaces bem delineadas, garantindo a coesão e a baixa dependência entre os componentes do sistema. Nomeadamente:

(1) apresentação - lida com a interface do usuário, a aparência e o layout do *website* - composta por temas e arquivos de template (.tpl.php);

(2) lógica de negócios - manipula a lógica, regras e funcionalidades do *website* - e contém os módulos personalizados que interagem, processam e transformam os dados recolhidos para que possam ser usados na primeira camada;

(3) armazenamento de dados - gere o acesso, a persistência e a recuperação de dados - é composto pelo banco de dados (MySQL ou PostgreSQL) responsável pela interação dos dados com a camada anterior.

Esta abordagem content-centric permite uma clara separação de responsabilidades, o que facilita a manutenção, a reutilização de código e o desenvolvimento escalável de um *website*. Além disto, Drupal tem uma grande e ativa comunidade que contribui para o seu próprio desenvolvimento (Buytaert, 2018*;* Saraiva, 2013).

O sistema DotNetNuke foi inicialmente lançado em 2002, com o intuito de desenvolver de um modo dinâmico *websites* e aplicações, gerir conteúdos e utilizadores. Baseado na linguagem Visual Basic .NET (VB.NET), este sistema funciona com o auxílio de Microsoft Internet Information Services (IIS) e do Microsoft SQL Server (Express 2005 or Desktop Engine 2000 for free deployment), que fornece o suporte da base de dados. Desde o seu primeiro lançamento, o DotNetNuke encontra-se atualmente na sua quarta edição (versão 4.0), que requer a versão 2.0 da .NET Framework.

A gestão deste sistema é realizada através do *front-end*, sendo que existem três camadas de gestão – uma (camada de acesso de dados) para a gestão de dados e de acesso aos mesmos, a segunda (lógica de negócio) contém a lógica operacional do sistema e, por fim, a terceira camada (apresentação) contém opções de interação entre utilizadores e diversos mecanismos de apresentação do *website*.

Em termos de extensões, este sistema retém a possibilidade de adição de módulos, para obtenção de mais funcionalidades, ou *skins*, para customização da apresentação do *website.* Por fim, no DotNetNuke, a gestão do *website* é feita ao nível das páginas *web* e a configuração do mesmo é personalizável (Do Carmo, 2007; Saraiva, 2013; (Corp, 2019).

O sistema Joomla foi lançado em 2005, e tem uma aplicabilidade diversa, desde *websites* de empresas e de e-commerce a fóruns de comunidade. Este sistema foi construído em PHP e o suporte de base de dados é fornecido por MySQL. A aplicação do servidor é variável, desde que seja suportada por PHP e MySQL. Neste sistema, a gestão é feita através de uma interface de administração *back-end*, e a mesma é constituída por 3 camadas – a primeira funciona ao nível da estrutura do sistema e consiste nas bibliotecas e *plugins*, a segunda consiste nas aplicações que vêm acopladas ao Joomla – JApplications. A terceira camada ocorre ao nível das extensões, onde são executados os módulos, componentes e a lógica.

No Joomla, existem três tipos de extensões que podem ser acopladas ao sistema – componentes, módulos e *plugins*. Os componentes funcionam como mini-aplicações, embora sejam também considerados um conjunto de módulos geridos por lógica operacional. Embora menos complexos, os módulos têm maior flexibilidade para serem reutilizados em diferentes componentes, pelo que é comum vermos os dois tipos de extensões a funcionarem em conjunto de modo a proporcionar uma variedade de conteúdo na interface. Além destes, ainda temos os *plugins* que do mesmo modo interagem com os anteriores através de código que é executado em cenários

pré-definidos dentro do CMS. Por fim, a gestão deste sistema é feita ao nível da página, sendo possível personalizar a estrutura do *website* (Open Source Matters, Inc., 2019; Do Carmo, 2007; Saraiva, 2013).

O sistema Magento foi inicialmente lançado em 2008 e tem uma aplicabilidade comercial, sendo que é usado para a criação de *websites* de *e-commerce* e lojas *online*. Algumas das funcionalidades principais destes *websites* incluem a gestão da loja *online*, a exposição de produtos, a disponibilização de diversos métodos de pagamento e envio de encomendas. Este CMS é baseado em Laminas, que tem por base a linguagem PHP, enquanto que o suporte de base de dados é fornecido por MySQL. No sistema Magento, a gestão é feita através de uma interface administrativo *back-end*, onde são fornecidas opções de configuração de diversos componentes do *website* (e.g., manutenção de encomendas, gestão de clientes e do catálogo), bem como de instalação e configuração de extensões, promovendo a inclusão de *third-party services* de modo a obter um *website* com mais funcionalidades (MacGregor, 2015; Adobe, 2013; Da Silva et al., 2017).

O sistema WordPress consiste num *website* de publicação e gestão de conteúdo, encontra-se em funcionamento desde o seu lançamento em 2009, e atualmente é o sistema mais utilizado (*Q-Success, 2023*). A linguagem base deste sistema é PHP e o suporte de dados é fornecido exclusivamente por MySQL. Apesar disto, este CMS combina a linguagem referida com HTML e JavaScript nas suas funções essenciais e utiliza o LAMP (Linux, Apache, MySQL, PHP). O sistema Wordpress apresenta uma estrutura modular e o seu núcleo gere as suas funcionalidades básicas, fornecendo também uma panóplia de temas e *plugins* que podem ser obtidos não só no repositório oficial do *website* WordPress.org, mas também através de *third-party sources*. Esta vasta gama de temas pode ser incorporada possibilitando a alteração da aparência do *website* e os *plugins* estendem a sua funcionalidade. Para além disto, o WordPress apresenta uma interface *front-end* que permite ao utilizador visualizar o conteúdo e uma página de administração *back-end*, onde é feita a gestão do conteúdo e de outras funcionalidades. Este CMS é o mais amigável ao utilizador pelo simples facto de que a interface é fácil de usar e muito intuitiva, permitindo assim que a uma curva de aprendizagem seja imediata e rápida. Reforçando esta noção, o WordPress permite que os utilizadores façam a gestão de tudo através da interface de admin sem qualquer intervenção técnica, desde criar/editar páginas, gerir ficheiros multimédia e até instalação/atualização de *plugins*. O facto de suportar um sistema de permissões e atribuição de

papéis a utilizadores permite a colaboração de vários utilizadores no *website* de forma controlada e organizada. (Ferreira, 2012; Baptista, 2014; Q-Success, 2023).

O Shopify é uma plataforma de comércio *closedsource* que foi fundada em 2004 e que oferece ferramentas de modo a simplificar a gestão de uma loja online de pequenos comerciantes (Shatrov, 2018). Este sistema representa 5,7% dos CMS usados, tornando-o o segundo maior no mercado, depois do WordPress (*Q-Success, 2023*). O Shopify é uma plataforma de *e-commerce* hospedada, o que significa que ele fornece um sistema completo para criar e gerenciar lojas *online*. Ele possui uma arquitetura centralizada, onde os usuários constroem e gerenciam as suas lojas na infraestrutura do Shopify (Shatrov, 2018). Este sistema utiliza o Ruby on Rails como estrutura subjacente, mas também usa HTML, CSS e JavaScript no desenvolvimento *frontend*. A infraestrutura do Shopify lida com o *hosting*, a segurança e o desempenho, permitindo que os utilizadores se concentrem simplesmente na personalização da loja e na gestão de produtos.

Em termos de funcionalidades, o sistema disponibiliza uma App Store com uma vasta gama de extensões que permitem aprimorar o funcionamento das lojas Shopify (Shatrov, 2018). Para além disso, este sistema fornece uma interface administrativa que permite gerir todos os aspectos de uma loja *online*. Os utilizadores podem criar listas de produtos, configurar *gateways* de pagamento, gerir pedidos e envios, personalizar temas e acompanhar análises, facilitando a gestão por parte de utilizadores não técnicos das suas lojas *online* (Shatrov, 2018).

Os Sistemas de Gestão de Conteúdo têm evoluído significativamente ao longo dos anos, embora ainda existam algumas lacunas de investigação neste domínio. Uma das lacunas mais evidentes é ao nível da segurança, na medida que o sistema pode ser vulnerável a tentativas de *hacking*, *malware* ou violação de dados (Baptista, 2014). Dois sistemas que servem de exemplo desta falha são o Joomla, o Wordpress e o Magento (Ferreira, 2012; Baptista, 2014; Pavlović et al., 2019). É crucial a investigação neste campo estar atualizada já que assegura a proteção dos dados de utilizador, e sendo que a integridade do *website* deve ser a prioridade aquando da criação e manutenção do sistema. Uma maneira de solucionar este problema seria optar por métodos de autenticação mais fiáveis como o OAuth ou o Forms authentication (Pavlović et al., 2019). Por outro lado, a instalação de *plugins* de terceiros também contribuiria para melhorar a proteção dos utilizadores (Baptista, 2014).

Na mesma ordem de relevância temos também as falhas ao nível da experiência e utilização por parte do utilizador. A maioria dos CMSs já providenciam interfaces *user-friendly*, tornando

expectável que este campo requeira competitividade. A facilidade de utilização do sistema é um ponto vital aquando da decisão de qual usar (Saraiva, 2013; Baptista, 2014). Neste aspeto, o Drupal e o Joomla são os que se destacam pela negativa, devido a existir uma acentuada curva de aprendizagem em ambos. No primeiro sistema porque existe necessidade de um conhecimento de programação *a priori* e de exigir a leitura de vários documentos técnicos, enquanto que no segundo se trata de falta de documentação útil (Baptista, 2014).

Parte da pesquisa realizada durante a elaboração deste sistema CMS teve em consideração as necessidades dos utilizadores e a importância de tornar os processos de criação e edição o mais simples possível. Deste modo, foi possível otimizar a utilização do CMS, tornando-o acessível a todo o tipo de utilizadores, desde os mais experientes aos que nunca tiveram contacto com este tipo de interface. Outra ideia a salientar é o facto de, aquando da formatação do texto na introdução de conteúdo, certos sistemas não conseguem oferecer o resultado desejado. Esta lacuna é facilmente resolvida com a instalação de módulos de formatação de texto, como módulos WYSIWYG (Baptista, 2014).

Outras lacunas a ter em consideração são as relacionadas com a acessibilidade e a linguagem disponível. Assegurar a capacidade de utilizadores com incapacidades usarem o sistema é um aspeto crucial tanto do *design* do *website*, como da gestão do conteúdo, portanto, é essencial assegurar o cumprimento de todos os requisitos de acessibilidade. Para transformar esta ideia numa realidade, foi necessário cumprir os standards da WCAG (*Web Content Accessibility Guidelines*), e, sendo assim, tornar o conteúdo e interface percetíveis - visíveis para, pelo menos, um dos sentidos; manter os componentes da interface e a navegação funcionais - interface não pode requerer uma interação que o utilizador não pode fornecer; tornar o conteúdo robusto - tecnologias de assistência têm de conseguir depender da interpretação do conteúdo; e, tanto o conteúdo, como a interface têm de ser compreensíveis - operações não podem requerer mais do que o utilizador consegue fazer (Refsnes Data, 2018). Além disso, CMSs deveriam suportar vários tipos de conteúdo e atender aos requerimentos em várias línguas. Para construir um sistema mais inclusivo e globalmente acessível, seria importante a integração de capacidades multilinguísticas e fazer a gestão de traduções (Refsnes Data, 2018).

Para além destes problemas, é necessário ter em atenção que é frequente a integração com sistemas externos como CRMs (*Customer Relationship Management Software*), plataformas *e-commerce* ou *third-party services*. De modo a tornar isto possível, é preciso aumentar as

capacidades de integração, assegurar a consistência dos dados e simplificar o processo de configuração (Halabi et al., 2017 em Chatterjee et al., 2022). Por outro lado, o sistema pode não ter o desempenho esperado e a capacidade de manter o seu funcionamento em circunstâncias diferentes para as quais foi inicialmente concebido. Nesse caso, seria interessante otimizar o desempenho e melhorar os mecanismos de *caching*, assim como utilizar arquiteturas escaláveis e aumentar as técnicas de equilíbrio de carregamento (*load*) (Misra, 2000).

As lacunas mencionadas destacam potenciais áreas de investigação com a finalidade de melhorar este género de sistemas, oferecendo a oportunidade de aprimorar a segurança, utilização, acessibilidade, desempenho e capacidades de integração, proporcionando aos utilizadores soluções de gestão de conteúdo mais robustas e eficientes. No futuro, a integração de tecnologias de inteligência artificial (IA) nos CMSs pode criar oportunidades de análise automatizada de conteúdo, gerar conteúdo ou até conceber personalizações e recomendações inteligentes (Paruchuri, 2018). Assim, investigar como a IA pode aprimorar os processos de gestão de conteúdo e as experiências dos utilizadores seria uma área de pesquisa a considerar.

Em suma, estes sistemas ou aplicações, que não só permitem editar/publicar conteúdo e gerir a manutenção de um *website* a partir de uma página central, mas também armazenam conteúdo numa base de dados, foram criados com a intenção de permitir aos utilizadores a publicação de ficheiros multimédia e documentos de texto de forma facilitada e autónoma. Deste modo, a redução de intervenções técnicas tornou todo o procedimento mais automatizado e consistente, que por sua vez permite acelerar o processo de atualização e cingir um maior número de utilizadores.

Para além dos benefícios mencionados, os CMS possibilitam a manipulação de conteúdo sem o receio de alterar o *design* e todo o *website* é atualizado com uma única alteração de dados, tornando tudo relativamente mais rápido. O facto de o administrador poder atribuir níveis de acesso aos utilizadores e controlar as alterações feitas, arquivando as versões anteriores, acrescenta aos CMSs uma noção de segurança por meio de aprovação de conteúdo antes de publicação. Estas noções adquiridas ao longo desta revisão de leitura contribuíram para realçar a utilidade e adaptabilidade destes sistemas às necessidades dos utilizadores e das empresas.

É importante realçar que em todos os sistemas existem problemas e obstáculos que impedem atingir o grau de excelência. Apesar de todos os métodos de facilitação, existem inúmeras

lacunas onde o programa pode fracassar e o facto de ser um ‘modelo pré-concebido’ induz a existência de características restritivas e a imposição de um certo nível de aprendizagem.

A conclusão mais imediata está diretamente relacionada com o método de análise que levou à estruturação do sistema em desenvolvimento neste Projeto Académico. Esta noção centra-se na ideia de que antes de se escolher um CMS, é necessário avaliar não só quanto conhecimento se tem e quanto é necessário no que diz respeito à elaboração e expansão, mas também quão intuitiva é a administração, quantos níveis de grupos de utilizadores e quanta codificação personalizada se espera. Posto isto, no decorrer deste trabalho, foi concebido um sistema com um *template* semelhante aos CMSs apresentados neste *state of the art*, embora as particularidades o tornem único. Dentro do aspecto qualitativo, será conduzida uma análise aos clientes de forma a conceber um sistema que responda às necessidades reportadas pelos utilizadores na atualização do *backoffice*, enquanto que ao nível quantitativo, é imperativa uma correspondência entre o número de funcionalidades implementadas e as necessidades dos clientes.

Um CMS é tipicamente dividido em dois componentes, a Content Management Aplication (CMA), uma interface gráfica que permite aos utilizadores modificar e criar conteúdos gráfico sem ter conhecimento de HMTL, e a Content Delivery application (CDA), que é a plataforma responsável por gerir toda a informação guardada, transcrevendo o código armazenado numa *datasource*, seja base de dados ou outro tipo de dados, e apresentando no *website* (Corp, 2019).

a. Principais características que um CMS necessita de ter

- **Segurança:** Deverá ter uma camada de segurança, que deve autenticar utilizadores a quem tenha sido fornecido acesso, devido à importância de ser desenvolvida uma camada de proteção para evitar ciberataques e impedir o acesso à informação dos clientes (Enginess, 2023).

- ***User-friendly*:** A interface gráfica deverá ser o mais simples e intuitiva possível, de modo a que os utilizadores consigam trabalhar com o CMS sem existir a necessidade de obterem conhecimentos técnicos (Enginess, 2023).

- **Escalabilidade:** O sistema CMS deverá permitir o crescimento do negócio e adaptar-se ao mesmo, de modo a fornecer o mesmo tipo de serviços de forma rápida e simples. Se o código desenvolvido e os servidores não conseguirem acompanhar este crescimento, o sistema pode gerar lentidão e perda de utilizadores (Enginess, 2023).

b. Arquitetura do CMS

No mercado dos CMS, existem duas principais abordagens de arquitetura de um CMS, a mais conhecida é o **CMS Tradicional (**Traditional CMS**)**, e a segunda denomina-se **Headless CMS** (Oracle, 2022) como representado na Figura1.

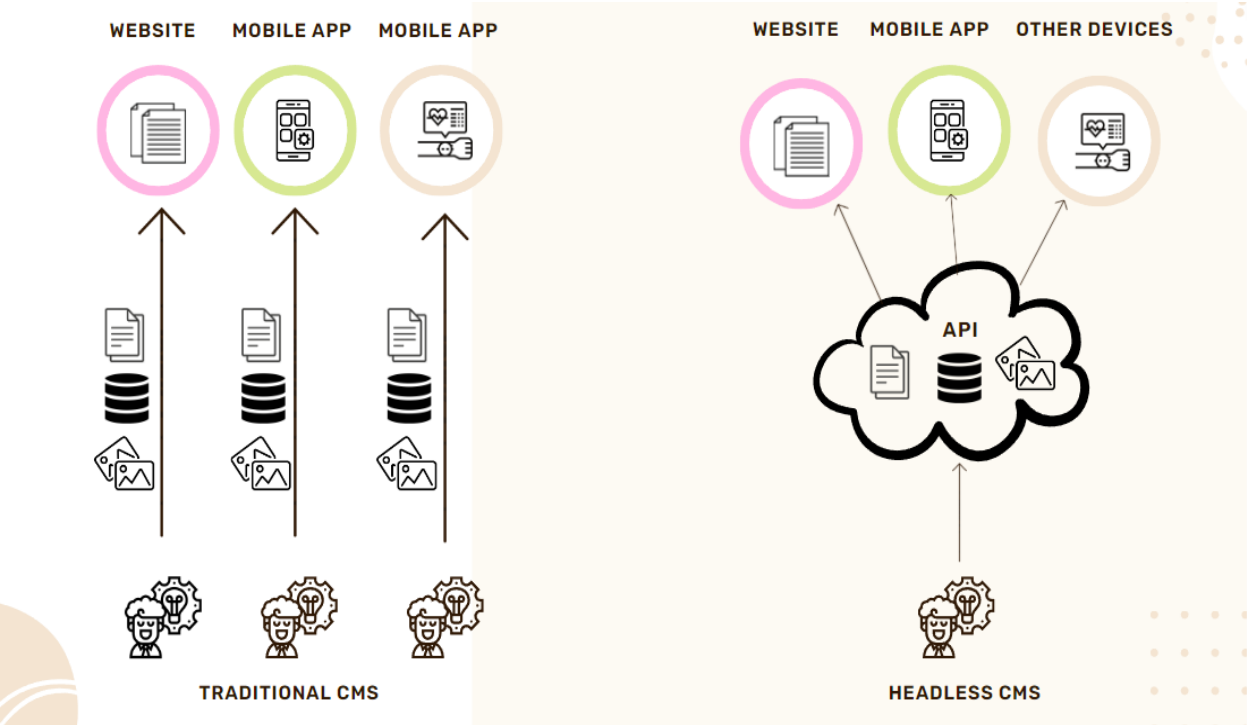


Figura1 – Traditional CMS vs Headless CMS. (Dias, 2023, p. 2)

O CMS Tradicional é um sistema integrado que combina a gestão de conteúdos (CMA), juntamente com a entrega do conteúdo (CDA) numa única plataforma. Sistemas como o WordPress e Drupal utilizam este tipo de CMS, no qual o conteúdo e a entrega do mesmo estão geralmente ligados, usando uma interface gráfica que ajuda os utilizadores a conseguirem criar

conteúdos e publicá-los para que fiquem disponíveis no *website*, que por outras palavras significa que o *layout* está preparado para receber este tipo de conteúdo (Buytaert, 2018; WordPress, 2023).

Características principais:

- O CMS é responsável pela criação e entrega de conteúdos;
- A base de dados onde é guardada a informação é central;
- Ideal para desenvolver *websites* simples, uma vez que os sistemas CMS possuem uma lista grande de *templates* que já são preparados para serem usados;

- Não é necessário conhecimento técnico avançado, com facilidade em fácil criar conteúdos e *plugins* de integração como, por exemplo, usar o sistema de pagamento PayPal numa loja (Oracle, 2022).

O Headless CMS é um sistema que desacopla a parte de gerenciamento de conteúdos (CMA) da entrega de conteúdos (CDA), o conteúdo é criado no *backend* usando uma base de dados, como *datasource* ou tipo de dados, e os conteúdos são disponibilizados através de uma API que pode ser consumida por qualquer interface, seja um *website*, uma aplicação *mobile*, ou qualquer dispositivo (Oracle, 2022)..

Características principais:

- O CMS foca-se apenas na criação e edição de conteúdos, estando totalmente desligado da camada de apresentação, permitindo criar uma maior flexibilidade e escolha de tecnologia;
- Unifica os conteúdos que são centralizados num único ponto central, e ao publicar, publica em diversos dispositivos e interfaces, usando o principio de COPE (Create Once, Publish Everywhere.);
- Melhora a reutilização de conteúdos e a escalabilidade, ou seja, cada aplicação consegue fazer a gestão individual de como apresentar os seus conteúdos, permitindo o sistema CMS crescer em escala juntamente com o negócio;
- É possível publicar os conteúdos sem ter impacto ou causar indisponibilidade nas aplicações;

- Ideal para negócios que precisem de ter sistemas com multiplataforma (*Website*, aplicações Mobile, wearables, etc..) (Oracle, 2022)..

c. Principais diferenças entre um Traditional CMS e um Headless CMS

Tabela 1 – Diferenças entre Traditional CMS e Headless CMS (Dias, 2023, p. 10)

	Traditional CMS	**Headless CMS**
Conteúdos disponível Multiplataformas	X	✓
Escalabilidade	X	✓
Eficiência e performance do código	X	✓
Fácil de Instalar	✓	✓
Controlo total sobre todas as funcionalidades	✓	X
Conteúdos disponibilizados por API	X	✓

d. Problemas de segurança nos diferentes tipos de CMS

Um dos maiores e principais problemas ao nível de segurança, acontece nos Traditional CMS, que são sistemas *opensource*, portanto, qualquer utilizador tem acesso ao código-fonte, e com alguma facilidade consegue detetar falhas e explorar essas vulnerabilidades para retirar informação crucial e privada (Baptista, 2014).

É importante estar sempre atento às atualizações do sistema, uma vez que, um dos riscos de atualizar um CMS com uma versão muito recente é que algumas funcionalidades podem deixar de funcionar.

Os sistemas headless CMS são mais seguros neste aspeto. Como o sistema está separado da camada de apresentação, é mais difícil encontrar pontos de vulnerabilidade. A utilização de uma camada API-first, é uma boa prática de segurança que impede muitos dos ataques a que estes sistemas estão sujeitos (Oracle, 2022).

e. Diferença entre Sistema CMS opensource e closedsource

Sistemas como o WordPress, que é usado em 63.2% de todos os *websites* que contêm um CMS (*Q-Success, 2023*), e que é um sistema tão popular e *opensource*, aumenta muito a probabilidade de se encontrar falhas de segurança que se pode explorar e retirar informação sensível. Por outro lado, um sistema *opensource* geralmente é grátis e permite melhorar e adaptar as necessidades específicas do cliente, portanto, existe geralmente uma comunidade grande que pode colaborar para melhorar o código e melhorar as funcionalidades. Em contrapartida, além dos temas de segurança, o código *opensource* não tem uma equipa de produto preparada para ajudar qualquer questão que os utilizadores precisem.

Sistemas como o Shopify são designados *closedsources*, o que significa que o código-fonte não está publicado, e apenas é conhecido pela equipa interna da empresa. São sistemas mais focados no produto final, mais estáveis e com uma equipa de suporte que consegue ajudar a desbloquear alguns problemas. Em termos de segurança, é um pouco mais robusto, e menos propenso a ataques externos, como são sistemas mais fechados e menos populares, exige um esforço maior para conseguir encontrar falhas. Sendo este tipicamente um serviço pago, é constantemente monitorizado e sempre que existem novas versões ou melhoramentos, é atualizado automaticamente

3. MATERIAIS E MÉTODOS DE DESENVOLVIMENTO

O projeto foi inspirado pela necessidade desenvolver um sistema CMS adaptável a qualquer *Website,* independente da área de negócios, este CMS engloba um conjunto de funcionalidades que são comuns a diversos clientes.

O sistema foi desenvolvido para ser fácil de utilizar, criando um design moderno, simples e responsive adaptando facilmente a qualquer ecrã, desde telemóveis, tablets e monitores, os utilizadores têm acesso a todos os conteúdos e tem permissões de escrita assim como criação de ficheiros, envio de emails e criação de conteúdos, como se comprovou com a criação de um *script* detalhado , em formato de fluxograma para que fosse possível ilustrar o funcionamento do sistema, representado na Figura 1.

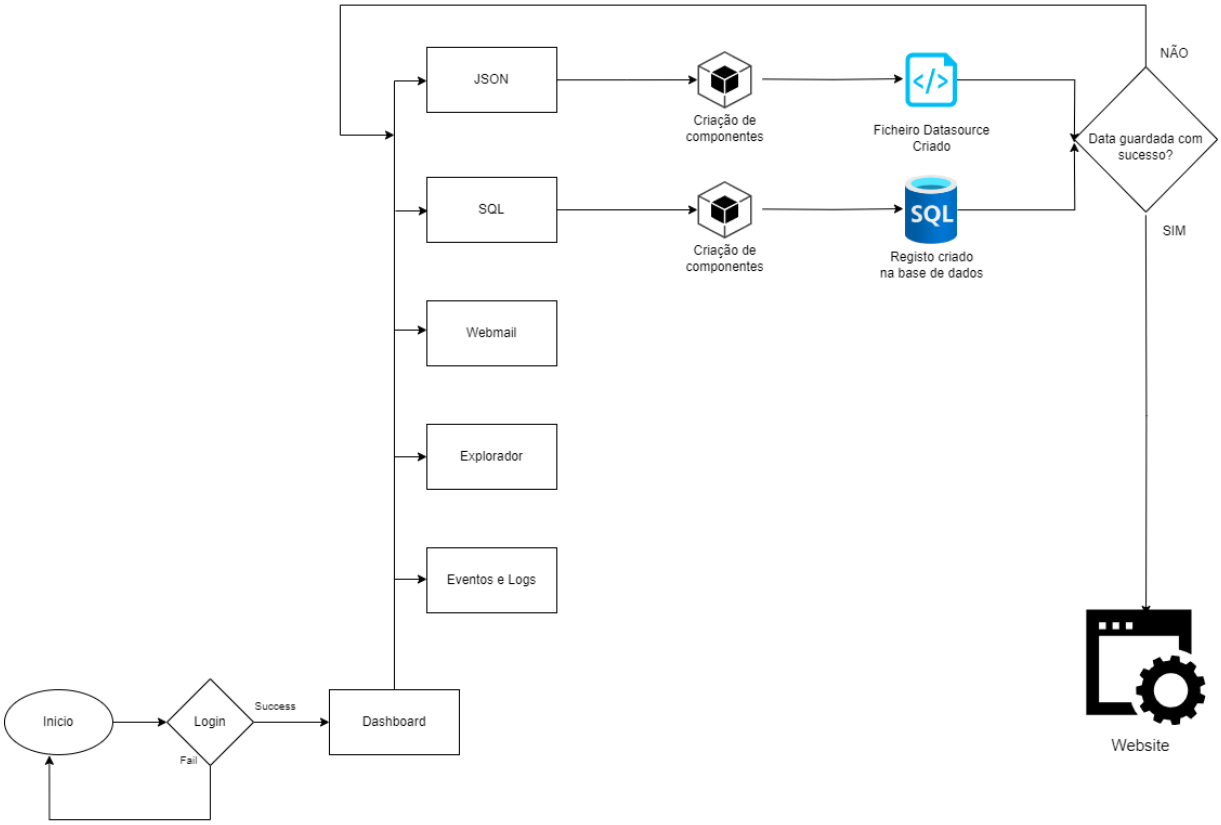


Figura 2 – Fluxo de acesso ao sistema CMS e suas funcionalidades. (Dias, 2023, p.1)

Aquando da organização deste projeto foi criado um Cronograma de investigação com os passos e respetiva distribuição temporal desde o inicio do projeto até à entrega da versão final.

Neste projeto foi utilizado uma metodologia predominante qualitativa com base num estudo de caso, inspirado pelo trabalho de Boukar et al. (2012):

> "Content management (CM) is one of the instruments that an enterprise needs to own to implement a knowledge management project. It is a system of methods and techniques to collect manage and publish content in a company (Boukar et al., 2012, p. 50).

Inicialmente foi realizada uma análise aos clientes de forma a perceber quais as dificuldades que mais reportam ao atualizar um CMS, nomeadamente em que funcionalidades surgem mais constrangimentos e questionado quais os pontos que estariam em falta de modo a tornar o sistema o mais eficiente possível. As questões realizadas abordaram temas como:

1) o tempo consumido no desenvolvimento,

2) existência de dificuldades na utilização destes sistemas,

3) aspetos a ser alterados para facilitar ao utilizador o seu uso e

4) existência de uma preferência por atualização automática do *website*.

Por questões de privacidade não serão feitas transcrições na integra, no entanto, as observações feitas encontram-se documentadas neste capítulo. Segundo o público alvo selecionado, existia uma necessidade de um sistema que reduzisse o tempo de desenvolvimento de um sistema *BackOffice* CMS e automatizasse significativamente o processo. Foi ainda realizada uma análise referente ao número de funcionalidades implementadas no CMS que vai ao encontro das necessidades dos clientes.

De modo a fundamentar o modo de implementação do *BackOffice*, foi feita uma revisão sistemática da literatura para identificar, selecionar e investigar os conteúdos, mais relevantes relacionados com os temas desta investigação. Posteriormente, foi feita uma análise à evolução dos CMS, com um especial foco nas diferenças entre os sistemas mais antigos e os mais recentes (Yermolenko & Golchevskiy, 2021) com a intenção de evitar erros cometidos no passado e aperfeiçoar ideias bem sucedidas.

Após a análise e recolha dos dados dos utilizadores inferiu-se que a o *Frontend* tinha de albergar a lógica de construção dos campos dinâmicos utilizando tecnologias HTML/CSS/JS e recorrendo a bibliotecas Javascript como Moustache, permitindo a criação de templates HTML e jquery como library de interação e manipulação de HTML. O *Backend* foi construído em

ASP.NET MVC, com APIControllers e base de dados SQL e foi usado como um *gateway* que permitia ler o *schema* da base de dados e passar a informação para o *Frontend*.

Todo o projeto foi realizado conforme as normativas éticas e sem causar qualquer violação de direitos dos utilizadores, mantendo sempre a segurança dos dados e dos próprios clientes, razão pela qual não existe nenhuma transcrição literal dos testemunhos obtidos. O sistema obteve autorização total para ser testado e os resultados documentados neste projeto tinham total aprovação.

4. RESULTADOS E DISCUSSÃO DE RESULTADOS

O principal objetivo do desenvolvimento e criação deste projeto é permitir o utilizador editar, adicionar remover e atualizar os conteúdos dinâmicos do seu *website* sem ter conhecimentos técnicos e experiência noutros sistemas CMS. Com o crescimento e evolução da internet cada vez mais surge a necessidade de termos sistemas que sejam capazes de conseguir gerir os conteúdos assim como a informação da base de dados desses *websites*. Com o crescimento e evolução das linguagens de programação, a complexidade dos *websites* foi aumentando, assim como a necessidade de conseguir atualizar os seus conteúdos de uma maneira fácil e rápida, sem a necessidade dos colaboradores terem formações para conseguirem executar o sistema (Oracle, 2022).

a. Principais problemas de segurança mais comuns num CMS

Quando se implementa um CMS, a preocupação principal que se tem é sobre segurança e proteção contra ciberataques e, é possível e importante desenvolver mecanismos que ajudem a bloquear possíveis ataques.

As principais técnicas utilizadas são:

- **Injeção de código**: Ao conseguir colocar código através de falhas de segurança como, por exemplo, SQL injection (*W3Schools, 2023*), o atacante consegue através de submissão de formulários enviar comandos de SQL que permite retirar informação de uma base de dados e até mesmo apagar os dados.
- **Cross-Site Request:** Ataques através de cliques, em que são enviados comandos para o servidor para retirar, danificar ou remover dados e usa a própria aplicação para aceder ao servidor pensando que é uma chamada legítima.

- **File Inclusion Exploitation:** CMS que não tem validação de envio de ficheiros maliciosos, permite enviar ficheiros com código, que ao executar esses ficheiros pode dar acesso a dados sensíveis comprometendo o sistema (Varty, 2023).

b. Como melhorar a segurança de qualquer CMS

Com a evolução e crescimento da comunidade da internet, os ciberataques a empresas também têm aumentando nos últimos anos, com o objetivo de retirar informação sensível e comprometer os sistemas. Posto isto, cada vez mais é importante criar um conjunto de mecanismos de segurança que protejam as aplicações e os seus dados.

Para aumentar a segurança, podemos implementar as seguintes alterações:

- **Certificado HTTPS:** É recomendável instalar um certificado de HTTPS principalmente na máquina em produção, pois, não só vai melhorar a posição do *website* no índex do Google, como vai criar uma camada de segurança e evitar ataques;

- **Vários níveis de autenticação**: Um sistema CMS deverá ter vários tipos de nível de autenticação para corresponder às diferentes funções que os utilizadores possam ter;

- **Monitorização e auditoria:** É importante construir mecanismos de monitorização em tempo real, onde é possível ver os erros gerados e toda a atividade dos utilizadores. Deste modo, um acompanhamento real ajuda na resolução rápida de possíveis *bugs*;

- **Atualização:** Os CMS principalmente *opensource* precisam de ser atualizados com alguma regularidade, a comunidade que gere estes sistemas é grande e disponibiliza com regularidade novas versões (Baptista, 2014; Pavlović et al, 2019).

Uma das grandes dificuldades sentidas no desenvolvimento deste CMS foram, como seria possível criar um sistema que conseguisse dinamicamente ler qualquer conteúdo de uma base de dados, seja uma tabela simples com três campos a uma tabela com vários tipos de campos. Para conseguir inserir qualquer data dinâmica fora desenvolvidas duas Stored Procedures (SP´s) que insere cada coluna e o valor inserido, faz uma validação e verifica se o tipo de dados inserido é conforme as características do mesmo. Foi construída outra Stored Procedure que lê cada tabela a data e a informação de cada campo, esta SP permite ler as informações e construir o *frontend* conforme a estrutura da tabela.

c. Autenticação

Como método de autenticação do protótipo, utilizamos Forms authentication da framework .NET, uma vez que permite autenticar diferentes tipos de utilizadores que podem ter acesso permitido ou negado aos recursos (HaiyingYu, 2022).

O primeiro que foi desenvolvido foi o de login (Figura 3), é fundamental criar um login para os utilizadores do CMS, todas as paginas de CMS têm que ser protegidas por um login para evitar ataques e utilizadores não autorizados a terem acesso à informação sensível e aos dados do sistema.

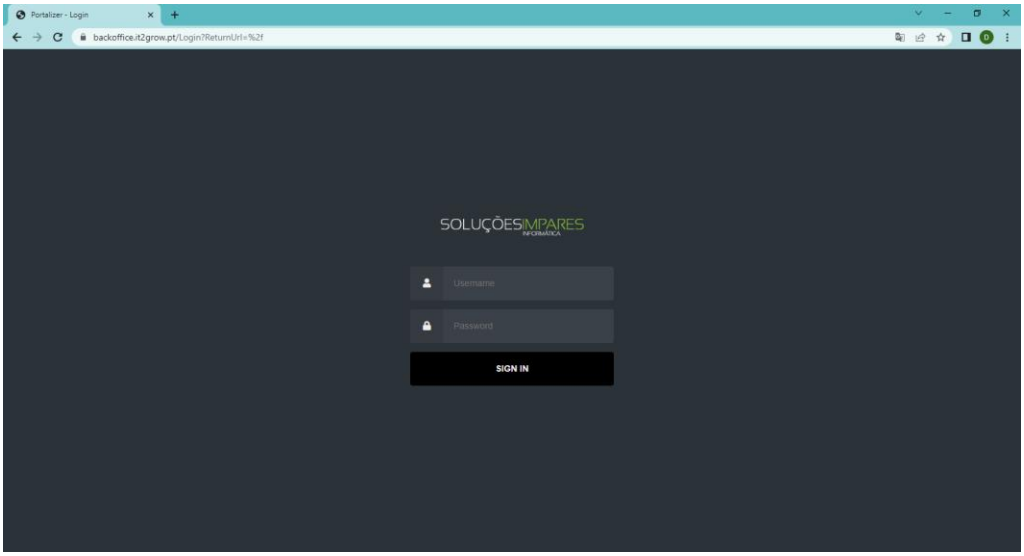


Figura 3 – Página de Login do CMS. (Dias, 2023, p. 2)

Foi decidido usar Forms authentication, por ser seguro, de fácil implementação e compatível com diferentes ambientes, tendo em conta que o *website* de uma empresa pequena e média utiliza planos de alojamento partilhados, o acesso ao IIS (Internet Information Services) ou painel de configurações é por vezes limitado. Este CMS não tem a necessidade de criar utilizadores dinâmicos, os utilizadores são criados na fase de instalação (Figura4), os users de acesso ao CMS são colocados no Web.config do projecto.

```
<credentials passwordFormat="Clear">
        <user name="admin" password="adminportal123"/>
        <user name="webmaster" password="portalizer123"/>
        <user name="editor" password="editor123"/>
```

Figura4 - Web.config do projecto. (Dias, 2023, p. 2)

d. Tecnologias utilizadas

O protótipo deste custom CMS foi desenvolvido em tecnologia ASP.net MVC, usa uma base de dados SQL e a troca de informação entre protótipo e base de dados é feita através de Stored Procedures dinâmicas.

Foi ainda utilizado algumas frameworks Javascript, para ajudar a criar a estrutura básica do esqueleto do template do protótipo (Bootstrap e Modernize) e outras para ajudar na implementação de funcionalidades e validações dos campos (Jquery, InputMask, Parsley, Fontawsome, datatables, chart.js)

e. Lista de requisitos

Antes do desenvolvimento de qualquer projeto, o mais importante é fazer o levantamento dos requisitos, pelo que, esta analise é o processo de refinamento e modelagem de construção do sistema.

f. Dashboard

Foi criado um dashboard do CMS (Figura5), que fornece aos utilizadores algumas informações sobre o seu sistema.

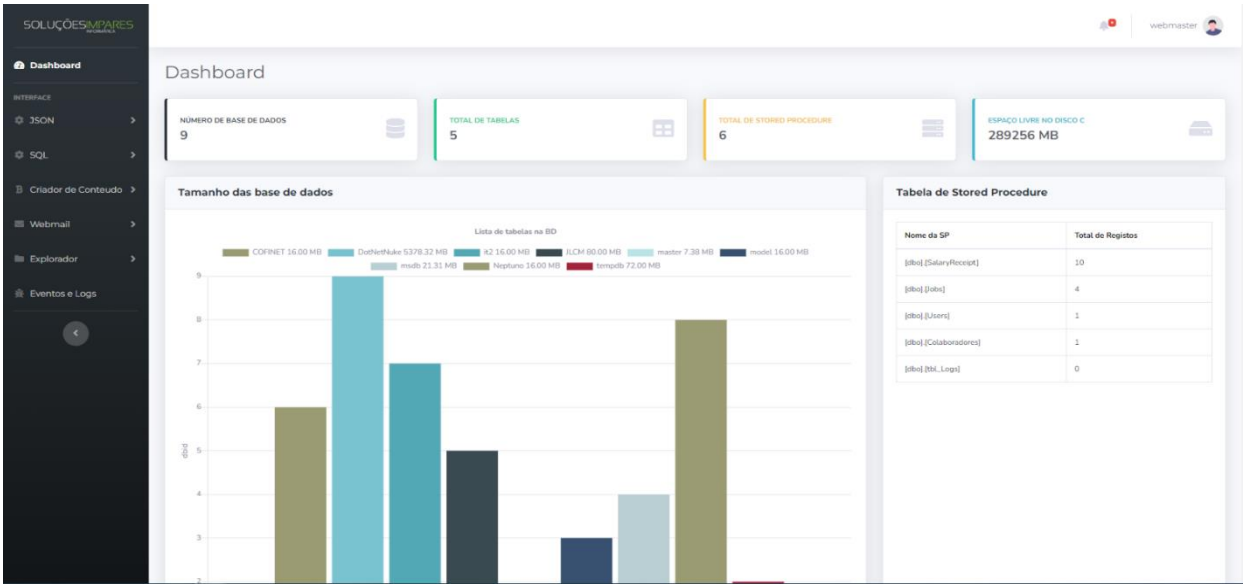


Figura5 – Dashboard do Custom CMS. (Dias, 2023, p. 2)

Neste dashboard, o sistema vai ler todas as bases de dados disponíveis no servidor, e vai apresentar o nome e o tamanho de cada base de dados, lê ainda as stored procedure, tabelas criadas, o total de registo de cada tabela, e o tamanho de disco disponível no servidor. Esta verificação inicial ajuda o utilizador a ter uma visão geral do seu ecossistema de dados, ajuda ainda o utilizador a perceber que o sistema CMS está ligado a um sistema de base de dados SQL válido e têm permissões de leitura.

No canto superior direito, temos um ícone de sinal de alerta, ao carregar aparece uma lista com diversas permissões (Tabela 3; Figura4), o sistema ao iniciar faz uma serie de testes de permissões, estas permissões vão depender do bom funcionamento do sistema, sem permissões básicas de escrita ou leitura o sistema não irá conseguir trabalhar.

Tabela 2 – Lista de permissões (Dias, 2023, p. 11)

Sistema valida as seguintes permissões
Ligação à base de dados
Execução de querys
Criação de tabelas
Criação de Views
Criação de Stored Procedure
Criação de Funções
Inserir dados na base de dados
Actualizar dados na base de dados
Apagar dados na base de dados
Executar stored procedure
Alterar SP/Tabelas

Figura6 – Lista de permissões do Custom CMS (Dias, 2023, p.3)

g. Tipos de dados permitidos

Neste CMS, é possível atualizar dados através de dois tipos de *datasources*, ficheiros JSON, e tabelas criadas no SQL Server.

No *datasource* JSON, o próprio ficheiro guarda a *data* e o *schema* do tipo de dados que vai ser apresentado, o CMS lê o *schema* de cada ficheiro e o *frontend* consegue interpretar e disponibilizar os objetos, permitindo ao utilizador apenas alterar conteúdos e gravar de novo o ficheiro (Figura7).

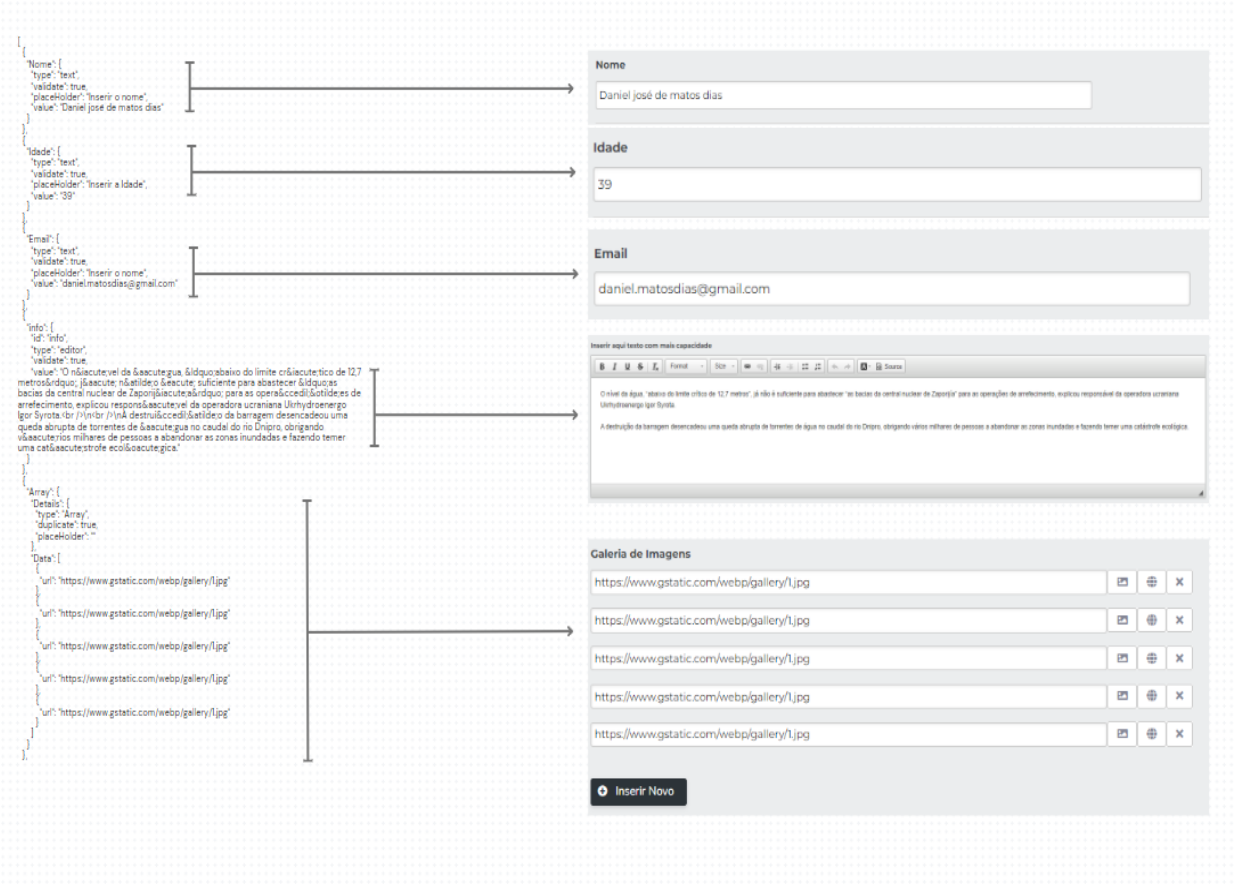


Figura 7 – Editor de ficheiros JSON. (Dias, 2023, p. 3)

No SQL, o sistema lê todas as tabelas, analisa cada tabela uma a uma, retira informações como o tipo de campo, se é texto, número ou data, valida ainda o tamanho máximo dos caracteres de cada campo, e informa as validações do *frontend*, aplica as características de cada campo às regras de validação.

Na Figura6, podemos ver a tabela de Jobs, o sistema vai ler o primeiro campo (idWork), vai analisar o campo e verificar se é chave primaria, se é um campo identity ou se têm algum default binding, caso seja não aparece no CMS porque são campos que são alimentados automaticamente.

No segundo campo *Titulo*, o tipo de campo é um nvarchar(max), este campo tem um limite muito grande de caracteres, o *frontend* retira a validação máxima de caracteres e alterar para uma textarea com scroolbar, para ser possível escrever mais texto, sendo um campo que aceita **Null**, também não é colocada nenhuma validação que obriga a guardar algum valor. No campo da

localidade, o tipo de campo é um nchar(100), aqui o sistema vai colocar como máximo de caracteres o valor de 100, e vai colocar permissão de escrita de números e letras.

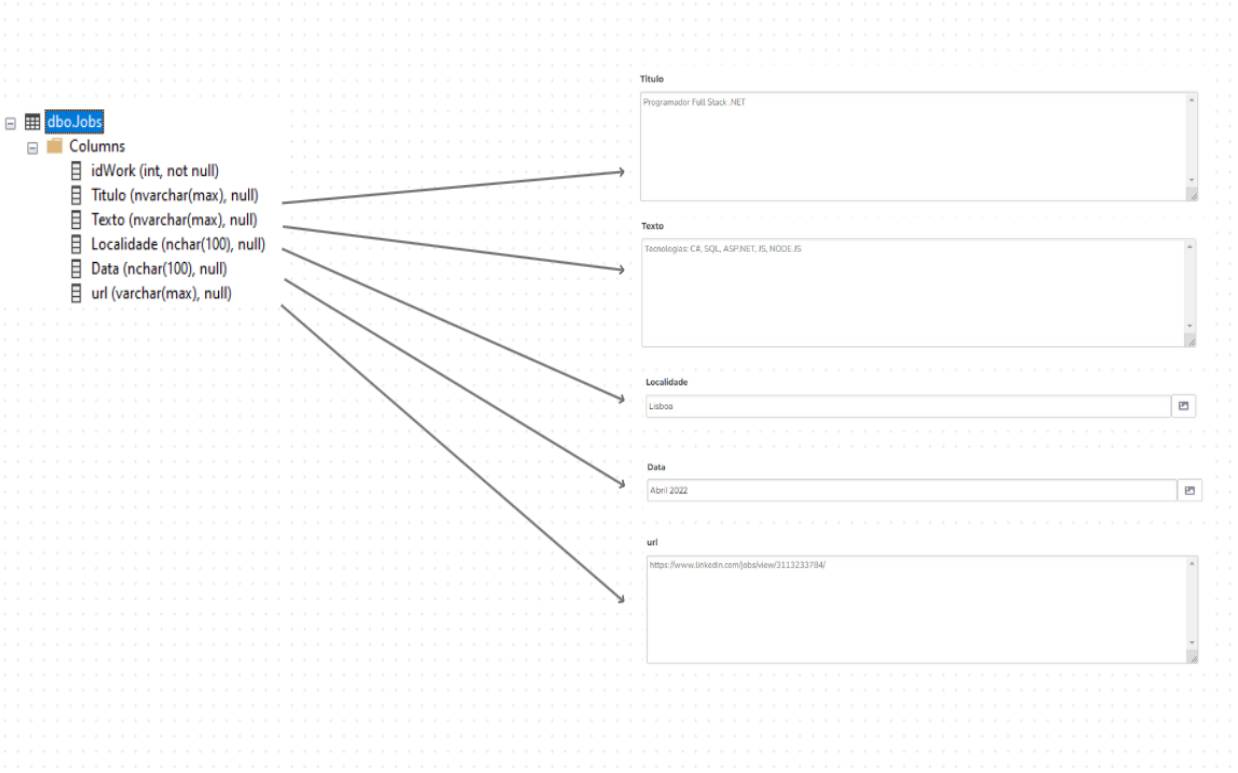


Figura 8 – Transformação de tabelas SQL em campos de texto. (Dias, 2023, p. 4)

h. Tipos de campos suportados pelo o CMS

O sistema CRM analisa cada campo da tabela e transforma os *input fields*, consoante o seu tipo de dados, insere validações automaticamente e adiciona as regras para apenas escrever o que é permitido.

Na Tabela 4 (Dias, 2023, p. 3) encontramos os tipos de dados suportados pelo sistema CMS e os tipos de campos mais usados nas tabelas SQL Server, de acordo com o SQL Online Courses | *LearnSQL.com* (n.d.).

Tabela 3 – Tipos de campos suportados no CMS. (Dias, 2023, p. 12)

Tipo	Tamanho máximo	Valores válidos
char	8000 caracteres	Permite qualquer caracter
varchar	8000 caracteres	Permite qualquer caracter
nvarchar	4000 caracteres	Permite qualquer caracter
text	2Gb maximo de texto	Permite qualquer caracter
ntext	2Gb maximo de texto	Permite qualquer caracter
varchar(MAX)	1,073,741,824 caracteres	Permite qualquer caracter
decimal	-10^38 até 10^38	Apenas números
numeric	-10^38 até 10^38	Apenas números
int	-2,147,483,648 até 2,147,483,647	Apenas números
bigint	-9,223,372,036,854,775,808 até 9,223,372,036,854,775,807	Apenas números
float	−1.79E+308 até 1.79E+308	Apenas números
Bit	0 e 1	Verdadeiro ou Falso
date	01/01/1973 ate 31/12/9999	Data (DD-MM-YYYY)
datetime	01/01/1973 00:00 ate 31/12/9999 24:59	Data e Hora (DD-MM-YYYY HH:MM:SS)
money	−922,337, 203, 685,477.5808 até +922,337, 203, 685,477.5807	Apenas números
smallmoney	−214,478.3648 até +214478.3647	Apenas números

i. Adaptação do frontend ao tipo de dados.

Nos tipos de campos datetime, o sistema adiciona um plugin de um datapicker (Figura9). O plugin do datapicker já é suportado e vem embutido nos browsers mais recentes, é construído um layout de um calendário que ajuda o utilizador a escolher as datas de maneira mais simples e sem problemas de inserir uma data errada.

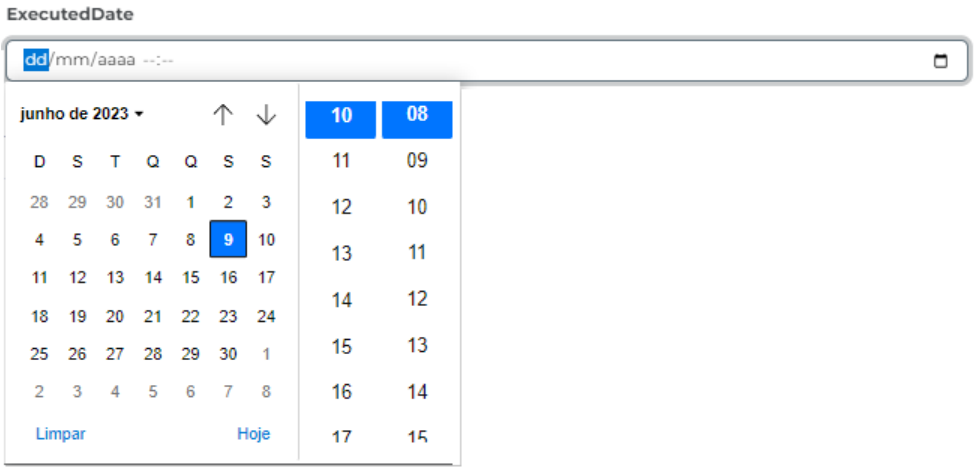


Figura 9 – Componente Datepicker. (Dias, 2023, p. 4)

Para textos que não tenham limite de caracteres como (nchar(MAX)), o frontend transforma numa textarea com scroolbar vertical.

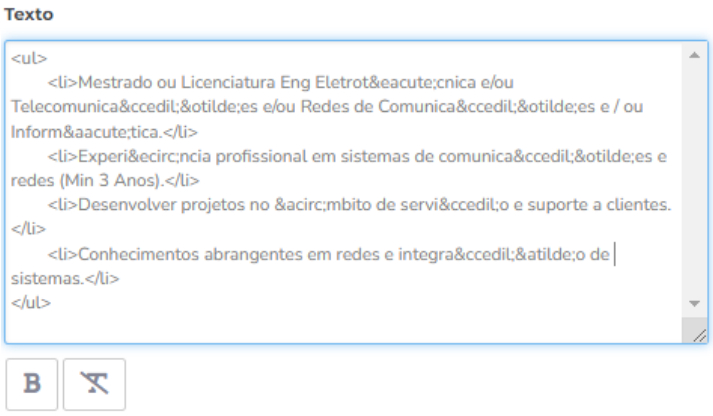


Figura 10 – Componente textarea simples. (Dias, 2023, p. 4)

Nestas textareas, por baixo existem dois botões, estes botões ao serem carregados irão transformar a textarea, num editor de texto com mais formatações. O ckeditor é um editor de texto plugin (read-to-use), ideal para conteúdos que necessitam formatações como listas, bold, tamanho da fonte, e tipo de fonte (Figura9).

O segundo botão, transforma o plugin ckeditor de volta à textarea normal, se utilizar algumas formatações o texto ficará com *tags* HTML, estas *tags* são depois interpretadas pelo *fronted* do *website* que é gerido pelo o CMS.

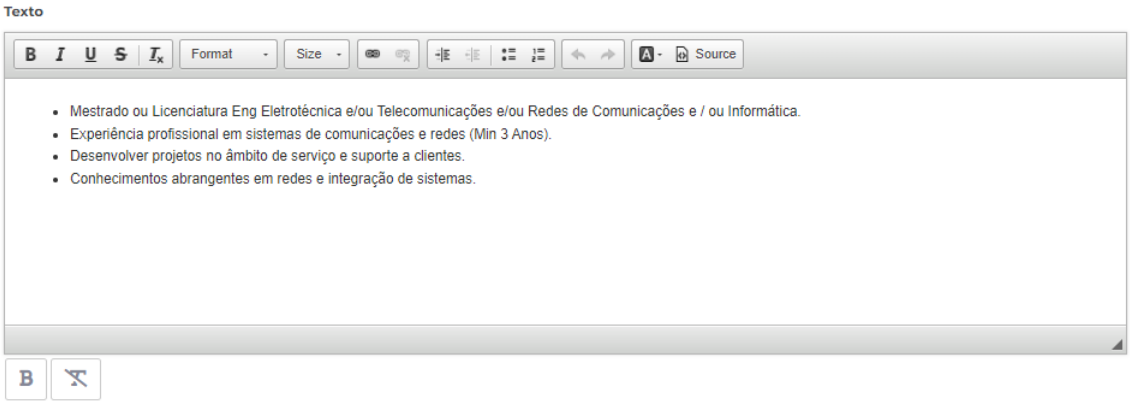


Figura 11 – Componente texarea transformado em editor com formatações. (Dias, 2023, p. 5)

Nas caixas de texto que não têm comprimento de nvarchar(max), as caixas de texto não têm a necessidade de ser uma textarea com scrollbar. No exemplo da figura10, o frontend do CMS irá montar uma caixa de texto, e como o tipo de campo é um nchar(200) que aceita nulls, o campo não é obrigatório preencher e fica limitado ao máximo de 200 caracteres (maxlength=200), aceita números e letras.

Figura 12 – Transformação campo da tabela em componente. (Dias, 2023, p. 5)

Estas caixas de textos têm um botão com um símbolo de imagem, ao carregar no botão irá aparecer um explorador de ficheiros, que irá permitir escolher ficheiros, neste caso queremos escolher o endereço para o ficheiro PDF onde está o CV do colaborador.

Este explorador de ficheiros, como se pode observar na Figura11, auxilia o utilizador sempre que necessita de escolher um ficheiro do sistema de ficheiros, permite a navegação dentro de várias pastas.

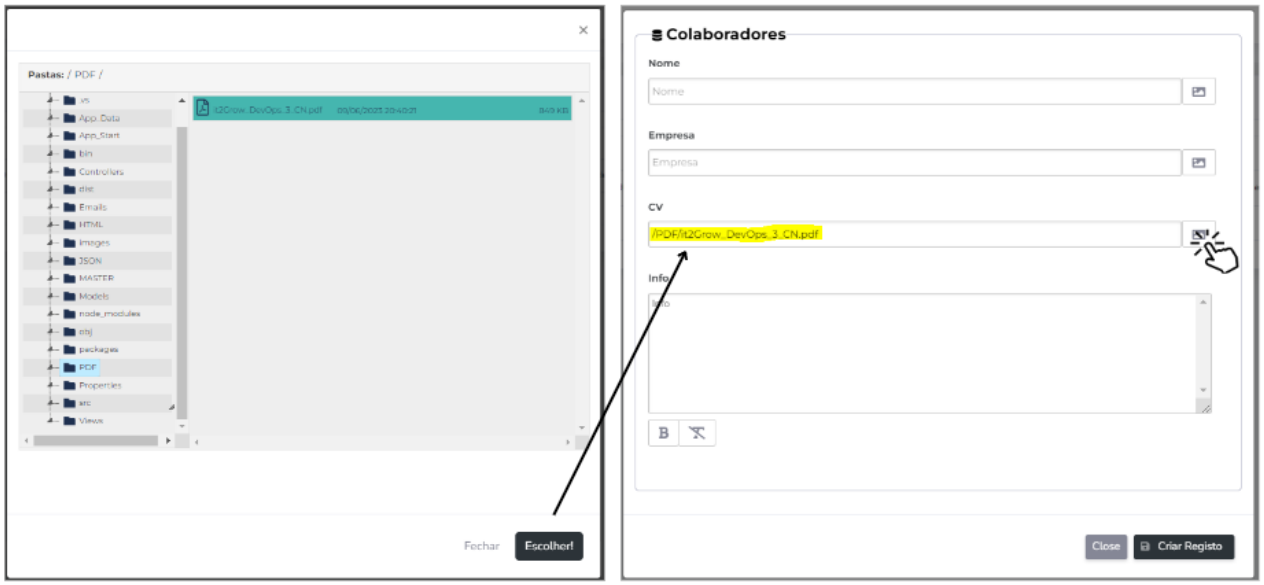


Figura13 – Explorador de imagens. (Dias, 2023, p. 5)

5. MÓDULOS DO PROTÓTIPO

a. Tipos e funcionalidades dos módulos do CMS

No desenvolvimento deste protótipo foram desenvolvidos seis módulos principais com o objetivo de fornecer ao cliente o maior numero de funcionalidades para conseguir atualizar e ter autonomia sobre o seu *website*, mas também tornar a atualização de conteúdos fácil e de simples manutenção, de modo a que o cliente não necessite de formações para conseguir dominar o sistema.

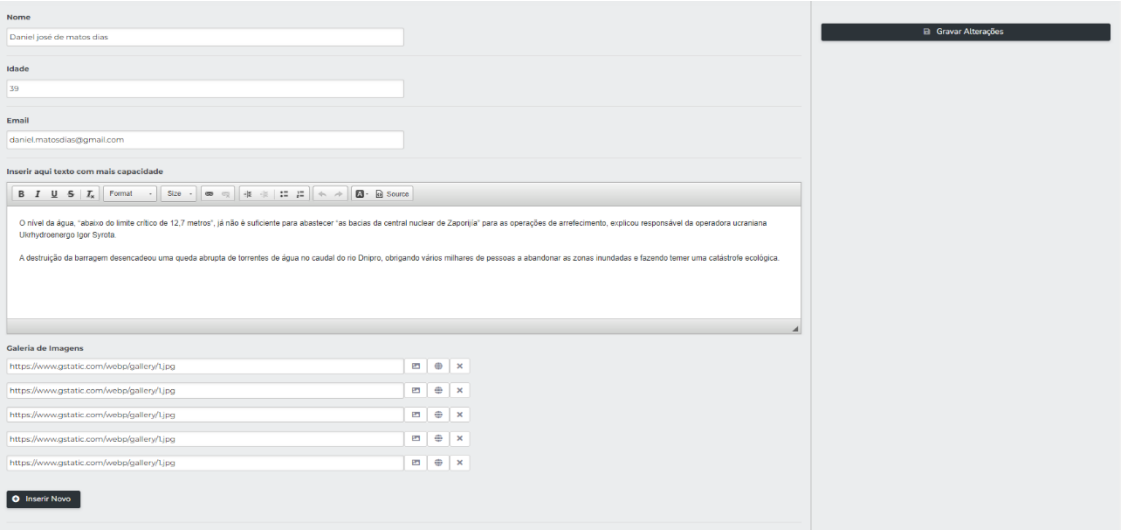
Módulo JSON: Permite atualizar ficheiros de configuração JSON, estes ficheiros contém a data a ser apresentada no *website*, assim como o esquema de cada tipo de dados. A informação pode ser montada com a estrutura que definirmos e o sistema vai analisar cada campo e construir os objetos para atualizar a informação. O *website* só precisa de ler estes ficheiros e mostrar a informação, sendo que idealmente estes ficheiros são uteis para conteúdos dinâmicos como galeria de imagens (Figura14).

Tipos de dados suportados (Dias, 2023, p. 11):

```
"Nome": { -> Nome do campo – Variável
        "type": "text", -> Tipo de dados – Fixo - (text | number | editor)
        "valid ate": true, -> Se têm validação obrigatória – Fixo - (true | false)
        "placeHolder": "Inserir o nome", -> descrição do campo a atualizar – Variável
        "value": "Daniel josé de matos dias" -> Valor da data a guardar no campo – Variável
    }

"Array": {– Fixo
        "Details": {
           "type": "Array", -> Tipo de dados – Fixo - (Array)
           "duplicate": true, -> Se é possível duplicar – Fixo - (true | false)
           "placeHolder": "Galeria de Imagens" -> descrição do campo a atualizar – Variável
        },
        "Data": [
           {
                        "url": "https://www.gstatic.com/webp/gallery/1.jpg" -> Data a ser atualizar.
           },
           }
```

Figura14 – Módulo JSON do Custom CMS. (Dias, 2023, p. 6)



Módulo SQL: Este módulo é o responsável pela analise do sistema de base de dados, lê as tabelas e campos e obtém o esquema do tipo de dados de cada um, transformando o frontend com objectos que permite ao utilizador criar, inserir, editar e apagar dados (Figura13).

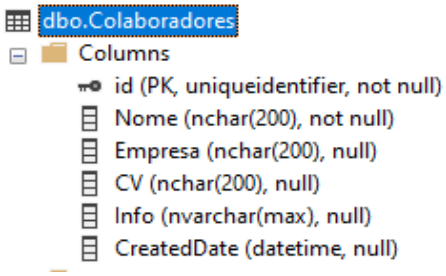


Figura 15 – Estrutura da tabela SQL (Dias, 2023, p.6).

Na Figura 14 podemos visualizar que o sistema vai ler o primeiro campo e vai analisar o tipo de dados que tem, como é uma chave primeira uniqueidentifier e com default Value (newid()) o sistema ignora este campo e não coloca no frontend para ser atualizável, ou seja, é atualizado automaticamente.

O segundo campo **Nome**, é um (nchar(200), not null) com tamanho de 200 e not null, o sistema vai gerar um campo de texto que permite números e letras, com o máximo de 200 caracteres e é obrigatório ser preenchido.

O terceiro e quarto campo **Empresa e CV**, é um (nchar(200), null) têm tamanho máximo de 200 caracteres e aceita nulls, logo não vai ser obrigatório ter que inserir informação.

O quinto campo **Info** é um (nvarchar(MAX)) aqui sendo um o sistema sabe que pode ser inserida muita informação, então coloca este como um textarea com scroolbar.

O ultimo e sexto campo **CreatedDate** é um datetime, o sistema sabe que só é valido datas fixas, transforma como um campo com um datepicker que só permite inserir datas.

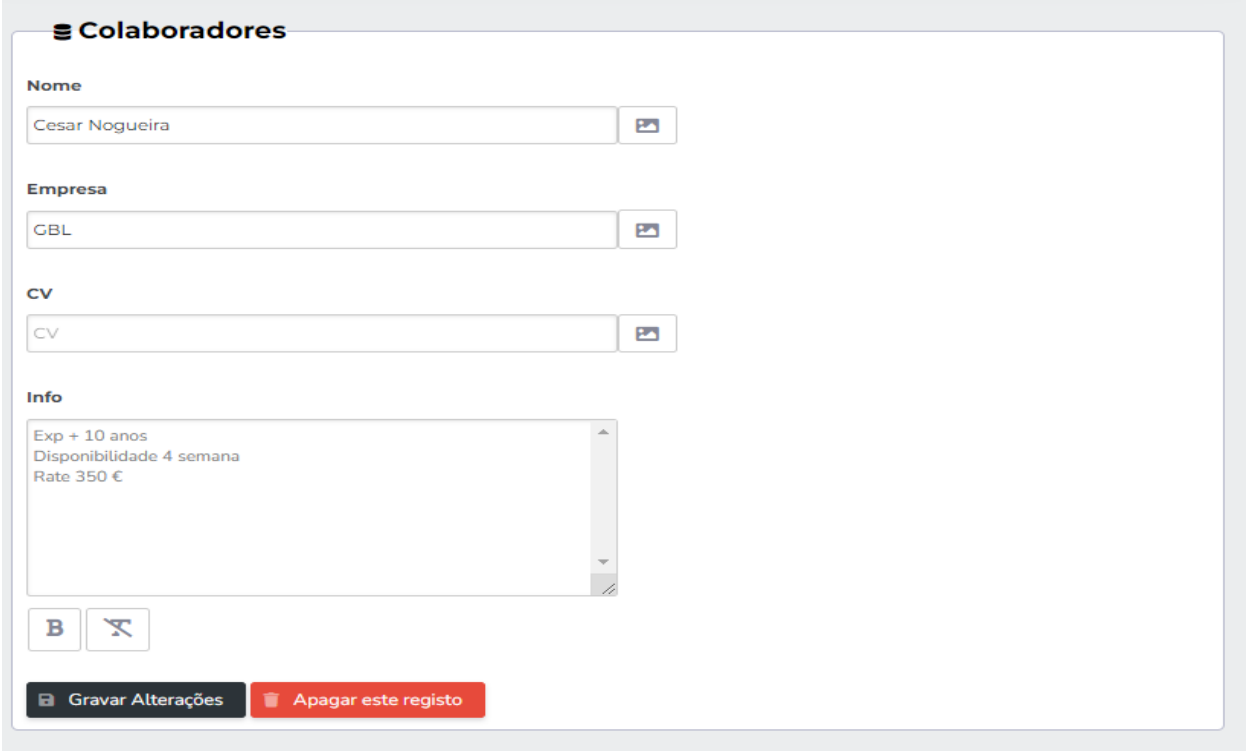


Figura16 – Painel de edição da coluna selecionada. (Dias, 2023, p.6)

Webmail: Módulo que permite enviar emails através das contas disponíveis no servidor, no qual é possível criar templates para tipos de emails padrão e enviar para os clientes, que é útil para o envio de emails de apresentação da empresa, de boas vindas, entre outros. Cada email enviado é guardado no sistema interno de logs do CMS (Figura15).

É possível alterar os *templates* dos emails de duas maneiras, a primeira é através do Módulo Criador de Conteúdos, basta abrir a pasta onde estão gravados os ficheiros HTML, editar e gravar com o mesmo nome. A segunda maneira é através do modulo, Explorador de Ficheiros, que permite editar o código fonte html de qualquer ficheiro.

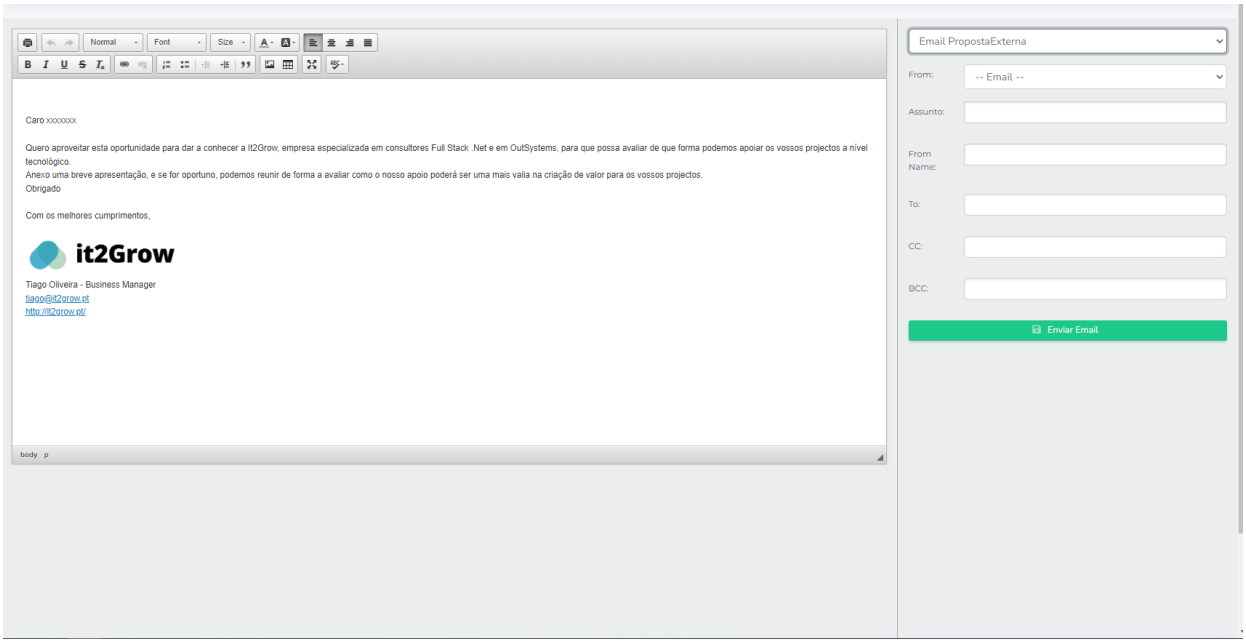


Figura 17 – Módulo de envio de email. (Dias, 2023, p. 7)

Criador de Conteúdos: Módulo responsável pela criação de conteúdos e ficheiros HTML, estes ficheiros podem ser lidos pelo *website* e inseridos dinamicamente, pode ser usado, por exemplo, para ter uma secção de notícias ou até um blog com vários artigos.

Este editor (ckeditor) permite inserir um conjunto de formatações pré-definidas, permite ainda transformar num editor de HTML, para utilizadores que tenham mais experiencia em construção de HTML, têm um modo básico em que apenas mostra algumas formatações simples e um modo completo (Figura16).

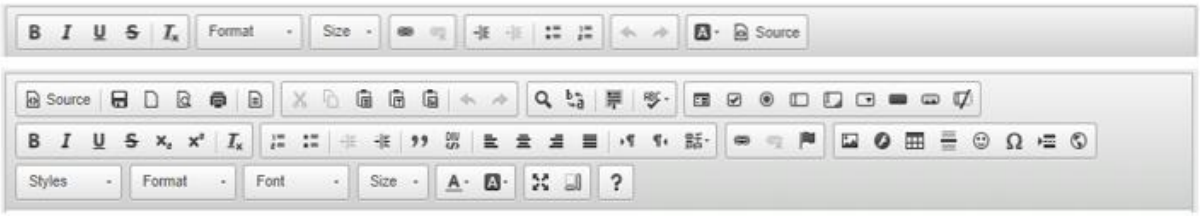


Figura18 – Editor de texto com formatações simples e completas. (Dias, 2023, p. 7)

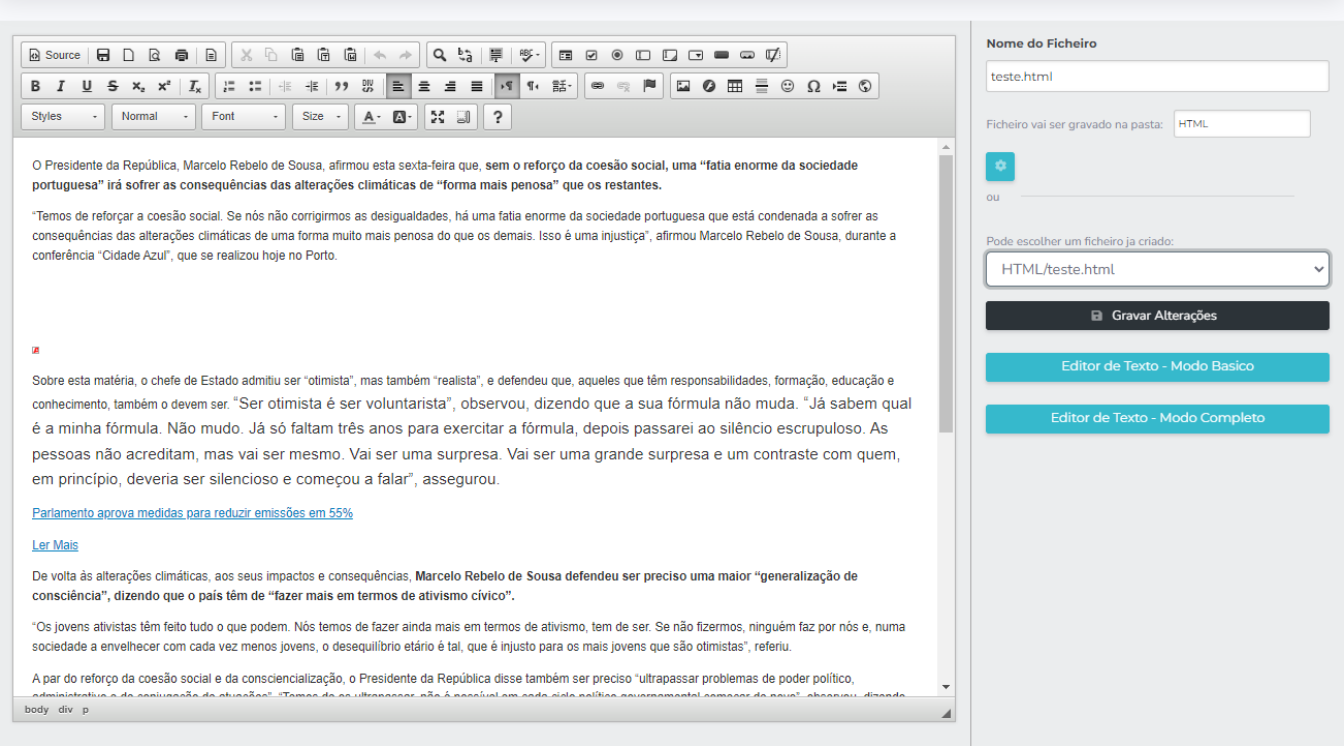


Figura19 – Módulo criador de conteudos. (Dias, 2023, p. 7)

Explorador de Ficheiro: Este módulo foi desenvolvido com a finalidade de criar um sistema de ficheiros que consegue dar ao cliente a capacidade de criar, apagar, editar e fazer upload de ficheiros e pastas, umas das dificuldade e problemas ao atualizar um *website*, é fazer a gestão dos ficheiros que são enviados para o servidor, por vezes os servidores ficam cheios de imagens ocupando muito espaço e criando lentidão nos recursos.

Este explorador permite, navegar, editar, criar e apagar pastas assim como permite fazer upload de múltiplos ficheiros de maneira rápida e fácil. Para além disto, ainda possibilita fazer operações ao nível dos ficheiros como download, abrir, copiar, cortar, colar e apagar, como é possível ver na Figura18 e Figura19.

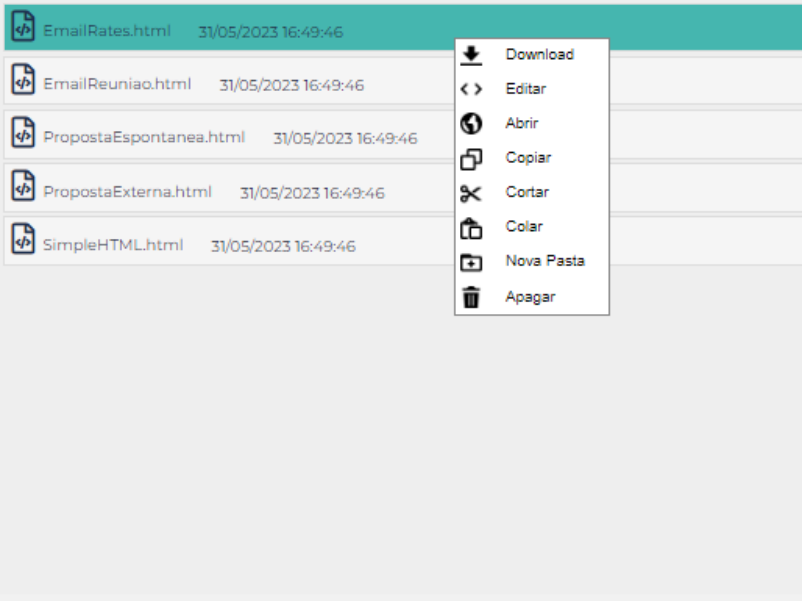


Figura20 – Funcionalidade do editor de ficheiros. (Dias, 2023, p. 8)

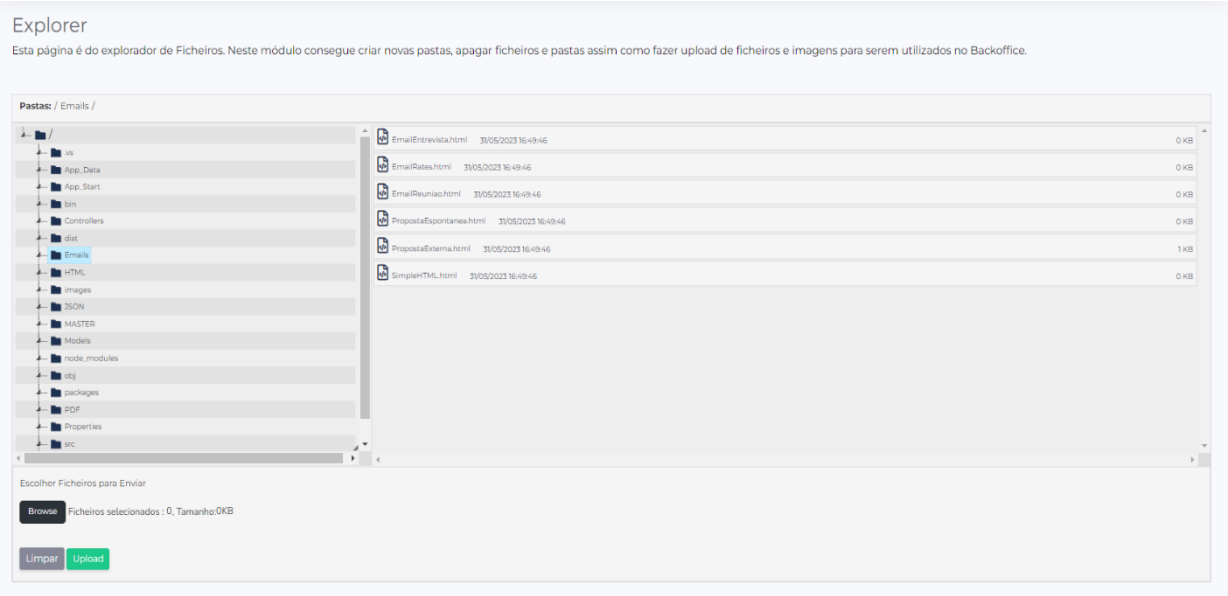


Figura 21 – Editor de ficheiros. (Dias, 2023, p. 8)

Foi desenvolvido ainda um editor de código fonte (Figura20), para utilizadores mais experientes que tenham a necessidade de alterar os *templates* criados, é possível editar ficheiros de texto, html ou código fonte.

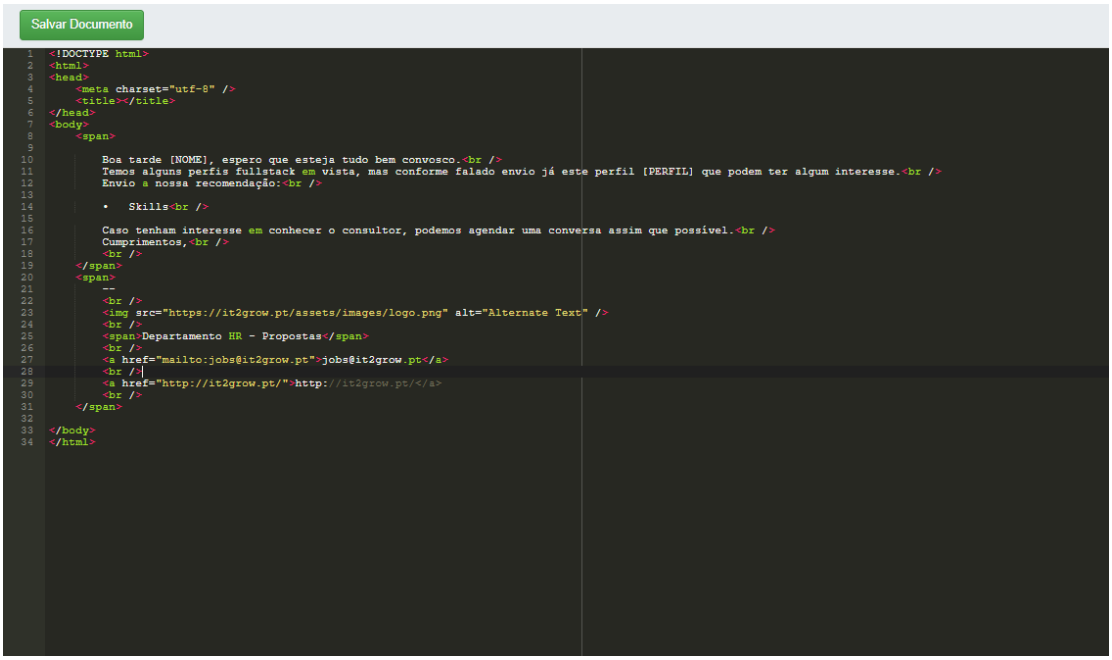
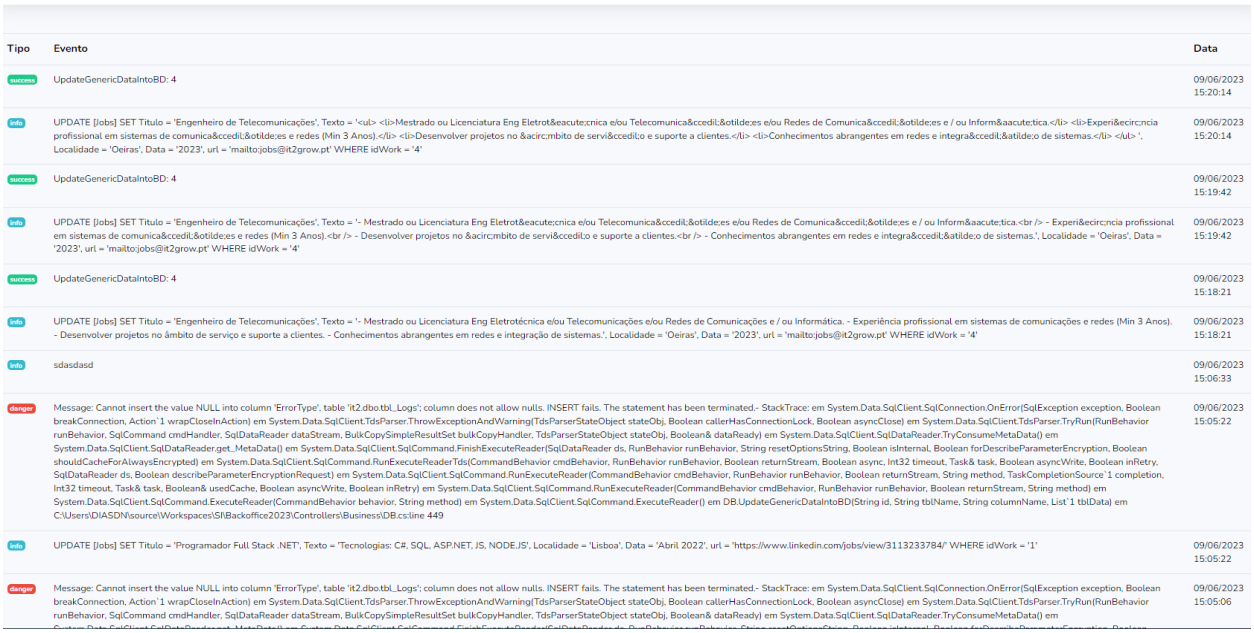


Figura 22 – Editor de ficheiros de texto e de código fonte. (Dias, 2023, p. 8)

Sistema de Logs: O sistema de logs é uma funcionalidade importante e deve ser obrigatória em qualquer CMS, ou sistema. Permite-nos monitorizar o sistema, fornecem uma visão geral do estado de saúde da aplicação, pode ser usado para identificar problemas e perceber se o sistema está a ser bem utilizado (Figura22).



Tipo	Evento	Data
success	UpdateGenericDataIntoBD: 4	09/06/2023 15:20:14
info	UPDATE [Jobs] SET Titulo = 'Engenheiro de Telecomunicações', Texto = ' - Mestrado ou Licenciatura Eng Eletrotécnica e/ou Telecomunicações e/ou Redes de Comunicações e / ou Informática. - Experiência profissional em sistemas de comunicações e redes (Min 3 Anos). - Desenvolver projetos no âmbito de serviço e suporte a clientes. - Conhecimentos abrangentes em redes e integração de sistemas. ', Localidade = 'Oeiras', Data = '2023', url = 'mailto:jobs@it2grow.pt' WHERE idWork = '4'	09/06/2023 15:20:14
success	UpdateGenericDataIntoBD: 4	09/06/2023 15:19:42
info	UPDATE [Jobs] SET Titulo = 'Engenheiro de Telecomunicações', Texto = '- Mestrado ou Licenciatura Eng Eletrotécnica e/ou Telecomunicações e/ou Redes de Comunicações e / ou Informática. - Experiência profissional em sistemas de comunicações e redes (Min 3 Anos). - Desenvolver projetos no âmbito de serviço e suporte a clientes. - Conhecimentos abrangentes em redes e integração de sistemas.', Localidade = 'Oeiras', Data = '2023', url = 'mailto:jobs@it2grow.pt' WHERE idWork = '4'	09/06/2023 15:19:42
success	UpdateGenericDataIntoBD: 4	09/06/2023 15:18:21
info	UPDATE [Jobs] SET Titulo = 'Engenheiro de Telecomunicações', Texto = '- Mestrado ou Licenciatura Eng Eletrotécnica e/ou Telecomunicações e/ou Redes de Comunicações e / ou Informática. - Experiência profissional em sistemas de comunicações e redes (Min 3 Anos). - Desenvolver projetos no âmbito de serviço e suporte a clientes. - Conhecimentos abrangentes em redes e integração de sistemas.', Localidade = 'Oeiras', Data = '2023', url = 'mailto:jobs@it2grow.pt' WHERE idWork = '4'	09/06/2023 15:18:21
info	sdasdasd	09/06/2023 15:06:33
danger	Message: Cannot insert the value NULL into column 'ErrorType', table 'it2.dbo.tbl_Logs'; column does not allow nulls. INSERT fails. The statement has been terminated.- StackTrace: em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady) em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() em System.Data.SqlClient.SqlDataReader.get_MetaData() em System.Data.SqlClient.SqlCommand.FinishExecuteReader(SqlDataReader ds, RunBehavior runBehavior, String resetOptionsString, Boolean isInternal, Boolean forDescribeParameterEncryption, Boolean shouldCacheForAlwaysEncrypted) em System.Data.SqlClient.SqlCommand.RunExecuteReaderTds(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, Boolean async, Int32 timeout, Task& task, Boolean asyncWrite, Boolean inRetry, SqlDataReader ds, Boolean describeParameterEncryptionRequest) em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method, TaskCompletionSource`1 completion, Int32 timeout, Task& task, Boolean& usedCache, Boolean asyncWrite, Boolean inRetry) em System.Data.SqlClient.SqlCommand.RunExecuteReader(CommandBehavior cmdBehavior, RunBehavior runBehavior, Boolean returnStream, String method) em System.Data.SqlClient.SqlCommand.ExecuteReader(CommandBehavior behavior, String method) em System.Data.SqlClient.SqlCommand.ExecuteReader() em DB.UpdateGenericDataIntoBD(String id, String tblName, String columnName, List`1 tblData) em C:\Users\DIASDN\source\Workspaces\Sfi\Backoffice2023\Controllers\Business\DB.cs:line 449	09/06/2023 15:05:22
info	UPDATE [Jobs] SET Titulo = 'Programador Full Stack .NET', Texto = 'Tecnologias: C#, SQL, ASP.NET, JS, NODE.JS', Localidade = 'Lisboa', Data = 'Abril 2022', url = 'https://www.linkedin.com/jobs/view/3113233784/' WHERE idWork = '1'	09/06/2023 15:05:22
danger	Message: Cannot insert the value NULL into column 'ErrorType', table 'it2.dbo.tbl_Logs'; column does not allow nulls. INSERT fails. The statement has been terminated.- StackTrace: em System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) em System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) em System.Data.SqlClient.TdsParser.TryRun(RunBehavior runBehavior, SqlCommand cmdHandler, SqlDataReader dataStream, BulkCopySimpleResultSet bulkCopyHandler, TdsParserStateObject stateObj, Boolean& dataReady) em System.Data.SqlClient.SqlDataReader.TryConsumeMetaData() em	09/06/2023 15:05:06

Figura23 – Painel de logs (Dias, 2023, p. 9)

Em conclusão os exemplos demonstrados permitem ao utilizadores conseguirem ter uma gestão dinâmica e intuitiva do seu website, este CMS foi desenhado para ser uma ferramenta util de trabalho, permitindo atualizar qualquer conteudo sem a necessidade de apoio ou formações para conseguirem utilizar.

Ao definir que tipo de campo ou secção o cliente pode atualizar, permite a atualização de qualquer bloco de HTML sem conhecimentos técnicos e sem correr o risco de ‘desformatar’ o website, este CMS fornece ainda um conjunto de ferramentas, como webmail, criação de conteúdos e páginas.

CONCLUSÃO

O mundo digital está em constante evolução e todo o conteúdo é exclusivo e personalizado conforme as necessidades específicas dos proprietários, pelo que para manter os sistemas de gestão de conteúdo atualizados e que sejam flexíveis e adaptáveis tornou-se uma tarefa fundamental. O sistema CMS em questão tem como objetivo principal automatizar e simplificar o processo de desenvolvimento de um BackOffice, permitindo que os clientes atualizem facilmente o conteúdo de seus *websites*, sem a complexidade dos CMS tradicionais.

Ao pesquisar as falhas e os problemas dos principais CMS existentes, foi possível identificar áreas de melhoria. Com base nessas descobertas, foi desenvolvido um protótipo de CMS que resolve essas falhas. O sistema resultante oferece um conjunto de funcionalidades comuns e úteis para os utilizadores que atualizam conteúdo num *website*, proporcionando controlo sobre a gestão dos respetivos *websites*, reduzindo significativamente o tempo necessário para aprender a utilizar o sistema.

Uma das principais conclusões sobre este sistema CMS centra-se na redução do tempo de desenvolvimento para as equipas de programação já que é um sistema dinâmico e genérico que agiliza o processo de criação de um BackOffice personalizada. Por outro lado o sistema foi projetado para ser facilmente instalado em servidores e utilizado por qualquer pessoa, mesmo sem conhecimentos técnicos avançados, daí ter uma aparência intuitiva e acessível. Com a existência da possibilidade de atualização de conteúdo autónomo, os custos associados à contratação de recursos externos para tarefas de manutenção e atualização são reduzidos ou mesmo nulos. Outro aspeto relevante, é a resposta indispensável às vulnerabilidades criadas por ameaças externas de *hacking*, *malware* e/ou violação de dados.

Em suma, este sistema CMS oferece uma solução empreendedora automatizada que atende ao objetivo de proporcionar uma experiência amigável e eficiente para o cliente, sem perder no processo o nível de segurança e de proteção de dados que um sistema de gestão de conteúdo implica.

BIBLIOGRAFIA

Adobe (Ed.). (2023). *Magento, now Adobe Commerce | eCommerce Software*. (n.d.). Retrieved May 12, 2023, from https://business.adobe.com/products/magento/magento-commerce.html

Automattic, Inc. (Ed.). (2019). *Blog Tool, Publishing Platform, and CMS - WordPress*. WordPress. https://wordpress.org/

Baptista, M. (2014). *Estudo Comparativo de CMSs - Estudo de Caso: uma IPSS*. (Dissertação de mestrado, Universidade da Madeira, Madeira, Portugal)

Buytaert, D. (Ed.). (2018, April 5). *Drupal - Open Source CMS*. Drupal.org. https://www.drupal.org/

Chatterjee, S., Mikalef, P., Khorana, S., & Kizgin, H. (2022). Assessing the Implementation of AI Integrated CRM System for B2C Relationship Management: Integrating Contingency Theory and Dynamic Capability View Theory. *Information Systems Frontiers*. https://doi.org/10.1007/s10796-022-10261-w

Corp, D. (Ed.). (2019). *Content Management System - .NET CMS Software from DNN*. (n.d.). DNN Software. Retrieved May 12, 2023, from https://www.dnnsoftware.com/

Da Silva, S. T., Da Silva, F. A., Maracci, F. V., & Pazoti, M. A. (2017). O Uso de Web Services para Integração de Sistemas ERP com Plataforms E-Commerce Magento. *Colloquium Exactarum*, *9*(1), 79–94. https://doi.org/10.5747/ce.2017.v09.n1.e186

Dias, D. (2023) Diário de Investigação - Observação de Campo. (Não Impresso). Pp. 1-11.

Do Carmo, J. (2007). *Web Content Management Systems - Experiences and Evaluations with the WebComfort Framework*. (Dissertação de mestrado, Instituto Superior Técnico, Lisboa, Portugal)

Enginess. (2023, January 3). *15 Must-Have Enterprise CMS Features*. Enginess. https://enginess.io/insights/must-have-enterprise-cms-features

Ferreira, F. (2012). *Proposta de Portal Empresarial para pequena Empresa prestadora de Serviços de Contabilidade.* (Dissertação de mestrado, ISCTE Business School, Lisboa, Portugal)

HaiyingYu. (2022, January 25). *Use ASP.NET forms-based authentication - ASP.NET*. Microsoft Learn. https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/aspnet/development/forms-based-authentication

Halabi, O., El-Seoud, S. A., Alja'am, J. M., Alpona, H., Al-Hemadi, M., & Al-Hassan, D. (2017). Design of Immersive Virtual Reality System to Improve Communication Skills in Individuals with Autism. *International Journal of Emerging Technologies in Learning (Ijet)*, *12*(05), 50. https://doi.org/10.3991/ijet.v12i05.6766

MacGregor, A. (2015). *Magento PHP Developer's Guide*. Packt Publishing Ltd.

Mahamat, M. (2012b). Content Management System (CMS) Evaluation and Analysis. *Journal of Technical Science and Technologies*, *1*(1), 50.

Misra, M. (2000). Architectural issues in building a high volume e-commerce company. *IETE Technical Review*, *17*(4), 235-240.

Moran, M. (2023). CMS Market Share: What Is the Most Popular CMS? *Colorlib*. https://colorlib.com/wp/cms-market-share/

Open Source Matters, Inc. (Ed.). (2019). *Joomla Content Management System (CMS) - try it for free!* (2019). Joomla! https://www.joomla.org/

Open Text Corporation (Ed.). (2023). *Vignette*. (n.d.). OpenText. Retrieved June 30, 2023, from https://www.opentext.com/about/brands/vignette

Oracle (Ed.). (2022). *The who, what, and types of content management systems*. Accessed 12 May, from https://www.oracle.com/pt/content-management/what-is-cms/

Paruchuri, H. (2018). AI Health Check Monitoring and Managing Content Up and Data in CMS World. *Malaysian Journal of Medical and Biological Research*, *5*(2), 141–146. https://doi.org/10.18034/mjmbr.v5i2.554

Pavlović, N., Šarac, M., Adamović, S., & Mravik, M. (2019). *Enchantment of Magento CMS Security*. https://doi.org/10.15308/sinteza-2019-223-228

Q-Success. (Ed.). (2023). Usage Statistics and Market Share of WordPress, June 2023. (n.d.). https://w3techs.com/technologies/details/cm-wordpress

Refsnes Data (Ed.). (2018). *Web Content Accessibility Guidelines (WCAG) Overview*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/standards-guidelines/wcag/

Saraiva, J. (2013). *Development of CMS-based Web Applications with a Multi-Language Model-Driven Approach.* (Dissertação de doutoramento, Instituto Superior Técnico, Lisboa, Portugal)

Shatrov, K. (2018, August 8). *E-Commerce at Scale: Inside Shopify's Tech Stack - Stackshare.io*. (2018, August 8). Shopify. https://shopify.engineering/e-commerce-at-scale-inside-shopifys-tech-stack

TYPO3 Association (Ed.). (2023). *TYPO3 - the professional, flexible CMS*. Retrieved May 12, 2023, from https://typo3.org/

Varty, J. (2023, May 3). List of Top CMS Security Vulnerabilities. Agility CMS. https://agilitycms.com/resources/posts/cms-security-vulnerabilities

W3Schools. (n.d.). *SQL Injection*. W3schools.com. Retrieved May 15, 2023, from https://www.w3schools.com/sql/sql_injection.asp

Yermolenko, A., & Golchevskiy, Y. (2021). Developing Web Content Management Systems – from the Past to the Future. *SHS Web of Conferences*, *110*, 05007. https://doi.org/10.1051/shsconf/202111005007