UNIVERSIDADE AUTÓNOMA DE LISBOA
LUÍS DE CAMÕES

DEPARTAMENTO DE INFORMATICA

LOW-CODE WEBSITE GENERATOR: A NEW APPROACH TO WEB DEVELOPMENT

Autor/a: Martim Ticas Moleiro
Orientador/a: Professor Laercio Cruvinel
Número do/a candidato/a: 30005462

Setembro de 2023
Lisboa

Agradecimentos

Gostaria de expressar a minha sincera gratidão ao corpo docente da Universidade Autónoma de Lisboa pelo seu apoio e orientação contínuos ao longo da duração deste projeto.

Uma menção especial ao Professor Laercio Curdival, o meu guia do projeto, cujos insights e conhecimentos foram muito importantes, melhoraram e tornaram possível a qualidade do trabalho. Também quero agradecer as contribuições dos colegas e de todos aqueles que participaram dos testes de usuário, fornecendo um feedback valioso.

Resumo (Português)

Esta tese explora a construção e as implicações de uma ferramenta geradora de websites de baixo código.

Nos dias atuais, onde a presença online é essencial para indivíduos e empresas, a capacidade de criar websites tornou-se uma habilidade valiosa. No entanto, muitos são desencorajados pela complexidade do desenvolvimento tradicional.

Esta tese explora a construção e as implicações de uma ferramenta geradora de websites de baixo código, surgindo como uma solução para este desafio. Através de uma abordagem centrada no usuário, a pesquisa buscou entender as principais barreiras enfrentadas por indivíduos sem formação técnica e como uma ferramenta poderia superar essas barreiras.

A proposta central da ferramenta é democratizar o desenvolvimento web, habilitando indivíduos com conhecimento limitado de programação a criar seus próprios sites de forma intuitiva.

Entre as características distintas da ferramenta, destacam-se a interface amigável, templates personalizáveis e integrações com ferramentas populares. Os testes iniciais mostraram uma recepção positiva, com usuários destacando a facilidade de uso e a flexibilidade oferecida.

As principais vantagens desta ferramenta englobam não apenas a aceleração do processo de desenvolvimento de sites, mas também a redução significativa de custos de desenvolvimento e manutenção.

Conclui-se que soluções de baixo código, quando bem projetadas, podem ser ferramentas poderosas para nivelar o campo de jogo no mundo digital, permitindo que mais pessoas participem ativamente da construção da web.

**Palavras-chave: Desenvolvimento de Baixo Código; Gerador de Sites;

Desenvolvimento Web; Amigável ao Usuário**

Resumo (English)

This thesis presents the development and implications of a low-code website generator tool.

In today's age, where an online presence is essential for individuals and businesses, the ability to create websites has become a valuable skill. However, many are deterred by the complexity of traditional development.

This thesis delves into the construction and implications of a low-code website generator tool, emerging as a solution to this challenge. Through a user-centric approach, the research sought to understand the main barriers faced by non-technical individuals and how a tool could overcome these hurdles.

The tool's core proposal is to democratize web development, enabling individuals with limited programming knowledge to intuitively craft their own sites.

Among the tool's distinguishing features are its user-friendly interface, customizable templates, and integrations with popular tools. Initial tests showed a positive reception, with users highlighting ease of use and the flexibility provided.

The main advantages of this tool encompass not only the acceleration of website development processes but also a significant reduction in development and maintenance costs.

It is concluded that well-designed low-code solutions can be powerful tools to level the playing field in the digital world, allowing more people to actively participate in web building.

**Keywords: Low-Code Development; Website Generator; Web Development;

User-friendly**

UNIVERSIDADE AUTÓNOMA DE LISBOA

Índice

Pág.

Pág.

Pág.

1 Introdução

A era digital trouxe consigo uma necessidade crescente de presença online. Seja para empresas estabelecidas, startups em ascensão ou indivíduos buscando compartilhar suas paixões, ter um website tornou-se quase um pré-requisito.

No entanto, o desenvolvimento web tradicional pode ser um processo demorado, técnico e, muitas vezes, caro. Isso levou a uma demanda crescente por soluções que tornassem o processo mais acessível e eficiente.

Neste contexto, emergem as ferramentas de desenvolvimento de baixo código. Essas ferramentas prometem uma abordagem mais simplificada ao desenvolvimento web, reduzindo ou eliminando a necessidade de codificação extensiva. Em vez de começar do zero, os usuários podem aproveitar modelos preexistentes, funcionalidades de arrastar e soltar e interfaces intuitivas para criar websites com aparência profissional em uma fração do tempo que levaria tradicionalmente.

A proposta de uma ferramenta geradora de websites de baixo código é, portanto, revolucionária em muitos aspetos. Ela tem o potencial de democratizar o desenvolvimento web, permitindo que mesmo aqueles sem formação técnica profunda possam criar e gerenciar seus próprios sites. Isso não apenas empodera indivíduos e pequenas empresas, mas também tem implicações significativas para a indústria de desenvolvimento web como um todo.

Nesta tese, exploramos a construção e as implicações de uma ferramenta geradora de websites de baixo código, enfocando sua capacidade de transformar o panorama do desenvolvimento web. Através de uma análise aprofundada da ferramenta, suas funcionalidades, benefícios e limitações, buscamos entender seu lugar no ecossistema digital atual e seu potencial impacto no futuro do desenvolvimento web.

1.1 Descrição Detalhada do Projeto

A Plataforma Online de Low Code apresentada nesta tese serve como um protótipo para demonstrar o poder e a eficiência das soluções de baixo código no desenvolvimento web. A ideia principal por trás do projeto é fornecer aos usuários uma interface intuitiva onde eles podem especificar características e funcionalidades desejadas para seus sites, e a plataforma gera automaticamente o código e a estrutura necessária para o site.

O projeto consiste em uma página principal (index.html) que serve como a interface do usuário. Através desta interface, os usuários podem selecionar diferentes funcionalidades, como um recurso de busca, seção de comentários, ou um boletim informativo. O JavaScript associado (main.js) captura as entradas do usuário e as envia ao servidor back-end para processamento.

1.2 Análise Técnica

A arquitetura do projeto é dividida entre cliente e servidor. O lado do cliente é composto principalmente pelo ficheiro index.html e pelo script main.js. O lado do servidor, implementado em Node.js, é responsável por gerar os sites com base nas especificações do usuário e também por gerenciar as interações com o banco de dados.

O servidor foi desenvolvido usando o framework Express.js, uma escolha popular para aplicações web em Node.js devido à sua simplicidade e flexibilidade.

O ficheiro server.js define as rotas e a lógica associada para gerenciar as solicitações do cliente. Além disso, o projeto utiliza um banco de dados SQL para armazenar informações sobre os sites gerados. Este banco de dados é essencial para rastrear e gerenciar os sites criados pelos usuários da plataforma.

Em termos de escolha tecnológica, a combinação de Node.js e Express.js oferece uma solução robusta e escalável. Node.js, sendo uma plataforma de execução JavaScript, permite uma experiência de desenvolvimento unificada, onde o mesmo idioma é usado tanto no cliente quanto no servidor. Express.js, por outro lado, oferece as ferramentas necessárias para criar rapidamente aplicações web sem a necessidade de reinventar componentes básicos.

1.3 Objetivos

O principal objetivo desta tese é apresentar e discutir a ferramenta geradora de websites de baixo código.

Os objetivos específicos incluem:

- Analisar o atual estado de desenvolvimento web e identificar as barreiras existentes para os indivíduos sem experiência técnica.
- Desenvolver uma ferramenta geradora de websites de baixo código que permita a criação de sites personalizados independentemente do nível de conhecimento técnico do seu criador, mesmo que sejam indivíduos com conhecimentos limitados de programação.
- Avaliar a eficácia e todas as implicações desta ferramenta para o campo de desenvolvimento web.
- Oferecer uma alternativa de baixo custo em comparação com soluções de desenvolvimento tradicionais, eliminando a necessidade de contratar especialistas para projetos de menor escala.
- Permitir a criação de diferentes tipos de sites (como blogs, lojas online, entre outros) através da utilização de templates pré-definidos e opções de personalização.
- Integração com Base de Dados que facilitam a conexão e a gestão de conteúdos através de bases de dados, proporcionando uma solução mais dinâmica e adaptável às necessidades dos utilizadores.

Estes objetivos foram definidos com base na necessidade de tornar o desenvolvimento web mais acessível e eficiente, especialmente para aqueles que não possuem um background técnico na área.

O objetivo principal do desenvolvimento web é criar soluções que atendam às necessidades dos usuários e dos negócios. Isso pode envolver a criação de sites promocionais, plataformas de e-commerce, aplicações empresariais, redes sociais, ou qualquer outra solução digital que ofereça valor ao usuário e atenda a objetivos específicos. Além disso, objetivos secundários podem incluir otimização para dispositivos móveis, integração com outras plataformas ou sistemas, e garantir segurança e privacidade para os usuários.

1.4 Motivação

A motivação por trás do desenvolvimento web evoluiu ao longo do tempo. Inicialmente, a web era vista principalmente como uma plataforma de informação.

No entanto, com o advento de tecnologias avançadas e a crescente digitalização dos negócios e da sociedade, a web transformou-se numa poderosa ferramenta de negócios, comunicação, colaboração e inovação. Empresas de todos os tamanhos reconheceram a importância de ter uma presença online robusta, não apenas para promoção, mas também para operações, atendimento ao cliente e inovação.

Na era digital contemporânea, o desenvolvimento web tornou-se fundamental para empresas, instituições e indivíduos. A capacidade de se conectar com audiências globais, apresentar informações e serviços e realizar negócios online é crucial para a competitividade e relevância. A evolução rápida das tecnologias web, desde simples páginas estáticas até aplicações web ricas e interativas, reflete a crescente demanda e as expectativas elevadas dos usuários.

O aumento da dependência de presenças online para negócios, educação e comunicação interpessoal trouxe uma demanda crescente por plataformas que permitam a rápida construção de

sites. As plataformas de baixo código surgem como uma resposta a essa demanda, eliminando barreiras e permitindo que mais pessoas participem ativamente da construção da web.

1.5 Fundamentação Teórica

O desenvolvimento de baixo código refere-se a abordagens e ferramentas de desenvolvimento que minimizam a necessidade de codificação manual. Ao invés de escrever linhas extensas de código, os desenvolvedores (ou até mesmo não-desenvolvedores) podem usar interfaces gráficas, arrastar e soltar componentes e configurar funcionalidades através de menus intuitivos.

O desenvolvimento web tem suas raízes nas primeiras páginas estáticas, que eram meramente informativas. Com o tempo, as tecnologias evoluíram, permitindo a criação de sites mais interativos e dinâmicos.

Hoje, as aplicações web são complexas, permitindo uma ampla gama de funcionalidades, desde redes sociais, comércio eletrônico até plataformas de aprendizado online.

1.5.1 Vantagens

O desenvolvimento web oferece várias vantagens distintas:

- Rapidez no Desenvolvimento: Um dos principais benefícios do desenvolvimento de baixo código é a velocidade. O que levaria semanas ou meses para ser codificado manualmente pode ser realizado em dias ou até horas.
- Redução de Custos: Ao acelerar o processo de desenvolvimento, os custos associados à mão de obra e ao tempo de desenvolvimento são significativamente reduzidos. Menos Erros: Como muitos componentes são pré-construídos e testados, há menos espaço para erros manuais.
- Democratização do Desenvolvimento: Permite que indivíduos sem formação em codificação participem ativamente do processo de desenvolvimento.
- Acessibilidade: Os websites são acessíveis de qualquer lugar do mundo, desde que haja uma conexão à internet.

- Flexibilidade: As aplicações web podem ser facilmente atualizadas e mantidas sem necessidade de distribuição ou instalação por parte dos usuários.
- Escalabilidade: Com as infraestruturas de cloud computing, as aplicações web podem ser escaladas para atender a grandes volumes de tráfego.

1.5.2 Desvantagens:

No entanto, o desenvolvimento web também enfrenta seus desafios:

- Menos Flexibilidade: Embora as ferramentas de baixo código sejam poderosas, elas podem não oferecer a mesma flexibilidade que a codificação manual.
- Dependência de Plataforma: Usar uma plataforma específica pode levar a uma dependência dessa plataforma, tornando difícil migrar ou alterar soluções no futuro.
- Segurança: Questões como ataques de injeção SQL, cross-site scripting e outros riscos são preocupações constantes.
- Otimização: Garantir que um site carregue rapidamente e funcione de forma eficiente pode ser um desafio, especialmente com o crescente número de dispositivos e resoluções de tela.
- Compatibilidade entre navegadores: Diferentes navegadores podem interpretar e rentabilizar os sites de maneira ligeiramente diferente, exigindo testes e ajustes adicionais.

1.6 Tendências Atuais em Desenvolvimento Web

O desenvolvimento web evoluiu significativamente desde os seus primeiros dias.

Originalmente dominado por páginas estáticas escritas em HTML puro, o cenário agora inclui uma variedade de tecnologias, frameworks e abordagens. Frameworks e Bibliotecas: Com o advento de bibliotecas como jQuery e frameworks como React e Angular, o desenvolvimento frontend tornou-se mais dinâmico e interativo.

Com a crescente popularidade dos dispositivos móveis, o design responsivo tornou-se essencial, garantindo que os sites sejam visualizados corretamente em todos os dispositivos. SPA (Single

Page Applications): SPAs são aplicações web que carregam uma única página HTML e atualizam dinamicamente conforme o usuário interage com a app. APIs e Microserviços:

A separação do frontend e do backend e a integração com serviços externos através de APIs tornaram-se comuns.

Com o crescente pedido por desenvolvimento mais rápido e eficiente, as ferramentas de baixo código estão a ganhar terreno.

2 Revisão da Literatura

2.1 Evolução do Desenvolvimento Web

O desenvolvimento da Web passou por transformações significativas desde o início da World Wide Web. Os primeiros sites eram simples, estáticos e construídos com HTML básico. Com o tempo, a introdução de CSS, JavaScript e vários frameworks adicionaram dinamismo e interatividade aos sites. Hoje, o desenvolvimento web abrange um amplo espectro de tecnologias, ferramentas e metodologias.

Desde os primeiros sites estáticos até as aplicações web dinâmicas de hoje, a web mudou drasticamente. Os primeiros sites eram simples, criados usando apenas HTML. No entanto, com a introdução de tecnologias como CSS, JavaScript e posteriormente, AJAX, os sites tornaram-se mais interativos e dinâmicos.

2.2 Plataformas de baixo código e sem código

A ascensão de plataformas de baixo código e sem código mudou o jogo no cenário de desenvolvimento web. Essas plataformas fornecem ambientes de desenvolvimento visuais, permitindo aos usuários criar aplicações arrastando e soltando componentes, em vez de escrever código manualmente. Algumas plataformas notáveis neste domínio incluem Wix, WordPress e Squarespace.

Nos últimos anos, observou-se uma tendência crescente em direção ao uso de plataformas de baixo código e sem código. Estas plataformas permitem que os desenvolvedores criem aplicações sem escrever código do zero, acelerando o processo de desenvolvimento.

2.3 Benefícios e desvantagens

O desenvolvimento web moderno oferece vários benefícios. A capacidade de criar sites que se adaptam a qualquer tamanho de tela permite uma melhor experiência do usuário. Além disso, as ferramentas modernas de desenvolvimento oferecem maior segurança e eficiência do que nunca.

Desenvolvimento Rápido: Acelera o processo de desenvolvimento eliminando a codificação manual.

Econômico: Reduz a necessidade de contratação de desenvolvedores especializados.

Escalabilidade: Os recursos integrados permitem fácil dimensionamento de aplicativos.

No entanto, eles também trazem desafios:

Personalização limitada: os componentes pré-construídos podem não atender a necessidades específicas.

Preocupações de desempenho: A dependência excessiva de plug-ins e componentes externos pode afetar o desempenho do site.

2.4 Inovações Tecnológicas no Desenvolvimento Web

O desenvolvimento web tem sido marcado por várias inovações tecnológicas que moldaram a forma como construímos e interagimos com websites e aplicações web. Algumas dessas inovações incluem:

- **AJAX (Asynchronous JavaScript and XML)**: Permitiu a criação de aplicações web dinâmicas ao possibilitar a atualização de partes de uma página sem a necessidade de recarregar toda a página.

- **Frameworks JavaScript**: Bibliotecas e frameworks como jQuery, Angular, React e Vue revolucionaram o desenvolvimento front-end, proporcionando ferramentas poderosas para a criação de interfaces de usuário interativas.

- **CSS3 e HTML5**: Estas atualizações das linguagens de marcação e estilo trouxeram recursos avançados, como animações, transições e elementos semânticos, que enriqueceram a experiência do usuário na web.

2.5 Impacto das Plataformas de Baixo Código e Sem Código

As plataformas de baixo código e sem código têm o potencial de transformar o desenvolvimento web ao tornar o processo mais acessível a pessoas sem formação em programação. Essas plataformas oferecem interfaces de arrastar e soltar, templates pré-construídos e integrações com outras ferramentas, reduzindo a barreira de entrada para o desenvolvimento web.

No entanto, também surgem preocupações sobre a flexibilidade e a personalização limitadas que essas plataformas podem oferecer em comparação com o desenvolvimento tradicional.

2.5.1 Impacto do Low-Code no Mercado de Trabalho

A ascensão das plataformas low-code e no-code tem implicações profundas no mercado de trabalho, especialmente no campo do desenvolvimento de software. A seguir, são apresentados alguns dos impactos mais notáveis:

2.5.1.1 Benefícios para Empresas

Para as empresas, a principal atração das soluções low-code é a capacidade de desenvolver e implementar aplicações rapidamente. Isso permite que as empresas respondam mais rapidamente às mudanças do mercado, lançando novos produtos ou adaptando-se a novos desafios.

Além disso, as soluções low-code podem resultar em economias significativas, pois reduzem a necessidade de equipas de desenvolvimento grandes ou especializadas.

Outro benefício é a democratização do desenvolvimento. Com plataformas low-code, mais funcionários dentro de uma organização podem participar do processo de desenvolvimento, mesmo que não tenham formação formal em programação. Isso pode levar a soluções mais inovadoras, pois diferentes perspetivas são levadas em consideração.

2.5.1.2 Implicações para Desenvolvedores

Para os desenvolvedores, as plataformas low-code representam tanto uma oportunidade quanto um desafio. Por um lado, essas plataformas podem simplificar e acelerar o desenvolvimento, permitindo que os desenvolvedores se concentrem em resolver problemas complexos em vez de se preocupar com a codificação básica.

Por outro lado, pode haver preocupações sobre a automação reduzindo os desenvolvedores especializados. No entanto, é importante notar que enquanto tarefas rotineiras podem ser automatizadas, sempre haverá uma necessidade de desenvolvedores para lidar com tarefas complexas, personalizações e integrações que vão além das capacidades das plataformas low-code.

2.5.1.3 Sustentabilidade e Impacto Ambiental das Plataformas Low-Code

Num mundo cada vez mais consciente das mudanças climáticas e da necessidade de ações sustentáveis, as empresas de tecnologia estão sob pressão para reduzir a sua pegada de carbono e adotar práticas mais verdes. As plataformas low-code, como parte deste ecossistema, não são exceção.

A natureza eficiente das plataformas low-code significa que as aplicações podem ser desenvolvidas e implementadas mais rapidamente, usando menos recursos. Isso pode-se traduzir em menos consumo de energia em comparação com métodos tradicionais de desenvolvimento.

Além disso, muitas plataformas low-code são baseadas na nuvem, o que significa que elas beneficiam das práticas sustentáveis adotadas pelos provedores de cloud. Por exemplo, grandes provedores de cloud como Google Cloud e Microsoft Azure têm esforçado para usar energia renovável nos seus data centers.

No entanto, ainda há desafios. O rápido crescimento e a popularidade das plataformas low-code significam que mais e mais aplicações estão sendo desenvolvidas e hospedadas, o que pode aumentar o consumo de energia. Assim, é crucial que as empresas por trás destas plataformas continuem a inovar e a encontrar maneiras de tornar suas operações ainda mais verdes.

2.5.1.4 Inteligência Artificial e Ética no Desenvolvimento Low-Code

Com o avanço da inteligência artificial (IA), não é surpresa que essa tecnologia esteja sendo integrada às plataformas low-code. Desde a otimização de código até a deteção automática de bugs, a IA tem o potencial de revolucionar o desenvolvimento low-code.

No entanto, com grandes poderes vêm grandes responsabilidades. A integração da IA levanta questões éticas importantes. Por exemplo, como garantir que a IA não introduza preconceitos no código? Ou como garantir a transparência e a explicabilidade das decisões tomadas pela IA durante o desenvolvimento?

Para abordar essas preocupações, é crucial que as empresas que desenvolvem plataformas low-code com capacidades de IA tenham diretrizes éticas claras e se comprometam a usar a IA de maneira responsável. Isso inclui a formação contínua de sua equipa, a transparência sobre como a IA é usada e a adoção de práticas de IA ética.

2.6 Considerações sobre Segurança em Plataformas Low-Code

Enquanto as plataformas low-code oferecem muitos benefícios em termos de desenvolvimento rápido e eficiente, elas também trazem desafios específicos em relação à segurança. Aqui estão algumas considerações importantes:

2.6.1 Riscos Associados

Dada a natureza automatizada do desenvolvimento low-code, pode haver uma suposição de que o código gerado é seguro. No entanto, isso nem sempre é verdade. Dependendo da plataforma e da implementação, o código gerado pode conter vulnerabilidades que podem ser exploradas por atores mal-intencionados.

Além disso, como muitas plataformas low-code são soluções baseadas em nuvem, há riscos associados ao armazenamento de dados sensíveis na nuvem. Se a plataforma for comprometida, isso pode levar a violações de dados e exposição de informações confidenciais.

2.6.2 Melhores Práticas

Para mitigar os riscos associados ao desenvolvimento low-code, é essencial adotar melhores práticas de segurança.

Algumas recomendações incluem:

- **Revisão Regular do Código**: Mesmo que o código seja gerado automaticamente, é crucial rever regularmente para identificar e corrigir possíveis vulnerabilidades.

- **Atualizações Contínuas**: Assim como qualquer software, as plataformas low-code devem ser atualizadas regularmente para garantir que todas as correções de segurança sejam aplicadas.

- **Autenticação Forte**: Implementar autenticação de dois fatores e políticas de senha rigorosas para proteger o acesso à plataforma.

- **Monitoramento**: Utilizar ferramentas de monitoramento para detectar e responder a atividades suspeitas em tempo real.

Ao seguir estas melhores práticas, as empresas podem aproveitar os benefícios das plataformas low-code enquanto minimizam os riscos associados.

2.7 Expansão do Conceito de Low Code

O conceito de 'Low Code' refere-se à criação e desenvolvimento de aplicativos com o mínimo de codificação manual possível. Isso é realizado por meio de interfaces gráficas intuitivas e ferramentas automatizadas que permitem aos usuários, mesmo aqueles sem experiência em programação, desenvolver e implementar soluções complexas.

Neste projeto, a plataforma de Low Code permite que os usuários gerem sites com funcionalidades específicas sem a necessidade de escrever o código por conta própria. Ao selecionar opções numa interface gráfica, os usuários podem determinar os recursos que desejam no seu site, e a plataforma cuida da geração de código. Isso não só acelera o processo de desenvolvimento, mas também torna a criação de sites acessível a um público mais amplo.

No entanto, enquanto as soluções de Low Code oferecem muitos benefícios, também existem desafios.

Um dos principais desafios é a flexibilidade. Embora essas plataformas possam cobrir muitos cenários de uso comuns, elas podem não ser adequadas para necessidades muito específicas ou personalizadas. Além disso, à medida que um projeto cresce e se torna mais complexo, pode haver uma necessidade de se desviar da abordagem de baixo código para atender a requisitos mais complexos.

2.8 Discussão sobre o Banco de Dados

O banco de dados desempenha um papel crucial na plataforma, atuando como um repositório centralizado para informações sobre os sites gerados. No projeto, o banco de dados SQL é usado

para armazenar detalhes como o nome do site, a descrição, e a data de criação. Cada site gerado recebe um ID único, permitindo fácil recuperação e gestão.

Além de armazenar informações sobre os sites, o banco de dados também pode ser usado para armazenar configurações do usuário, preferências e qualquer outro dado relevante. A decisão de usar SQL foi baseada na sua robustez, confiabilidade e ampla adoção na indústria. A estrutura relacional do SQL é ideal para gerenciar dados estruturados, como os encontrados neste projeto.

No entanto, como qualquer escolha tecnológica, o uso de SQL vem com seus próprios desafios. A necessidade de definir esquemas antes do tempo pode ser restritiva, e mudanças no esquema mais tarde no ciclo de desenvolvimento podem ser complicadas. Além disso, enquanto SQL é excelente para dados estruturados, pode não ser a melhor escolha para dados não estruturados ou semi-estruturados

3. Visão geral do projeto

A aplicação é composta por vários ficheiros e diretórios, incluindo ficheiros HTML, CSS, JavaScript, PHP e SQL. Cada um desempenha um papel específico na funcionalidade geral da aplicação:

- Ficheiros HTML: Estes ficheiros definem a estrutura e o conteúdo das páginas web. Exemplos incluem `contacto.html`, `equipe.html`, `help.html`, `index.html` e `sobre.html`.

- Ficheiro CSS (`styles.css`): Este ficheiro define o estilo e a aparência da aplicação, incluindo cores, layouts e tipografias.

- Ficheiros JavaScript: Estes ficheiros gerenciam a interatividade e a lógica da aplicação. Exemplos notáveis incluem `db.js`, `main.js`, `queryDatabase.js`, `script.js` e `server.js`.

- Ficheiros PHP (`connect.php`) e SQL (`database.sql`): Lidam com a conexão do banco de dados e a estrutura do banco de dados, respetivamente.

3.1 Conceitualização e Ideação

O trabalho de criação e sites de baixo código começou com a identificação da lacuna no mercado para uma plataforma que preenchesse a divisão entre usuários que não entendem de tecnologia e o mundo do desenvolvimento web. Sessões de brainstorming e pesquisas de mercado ajudaram a solidificar a visão da plataforma.

3.2 Recursos e funcionalidade*:

A aplicação parece ser uma aplicação web completa que utiliza tecnologias front-end e back-end. A escolha de tecnologias específicas, como PHP para gestão de banco de dados e JavaScript para lógica de front-end, sugere um foco na eficiência e na interatividade.

O uso de um ficheiro SQL sugere que a aplicação tem uma componente de banco de dados, possivelmente para armazenar informações do usuário ou conteúdo dinâmico.

A plataforma oferece uma variedade de recursos, desde a simples criação de páginas até a integração com bancos de dados. A ênfase está na simplicidade e na capacidade do usuário de personalizar o seu site de acordo com suas necessidades.

A aplicação web, como evidenciado pelos ficheiros fornecidos, parece ser uma solução completa que atende a vários requisitos de usuário:

- **Interface de Usuário**: A interface da aplicação é projetada para ser intuitiva e amigável ao usuário, facilitando a navegação e interação. Os ficheiros HTML e CSS desempenham um papel crucial na definição desta interface.

- **Interatividade**: Com os ficheiros JavaScript fornecidos, pode-se inferir que a aplicação oferece um alto grau de interatividade, permitindo que os usuários realizem várias ações e recebam feedback em tempo real.

- **Gestão de Dados**: Os ficheiros PHP e SQL sugerem que a aplicação tem uma robusta gestão de dados, possibilitando o armazenamento, recuperação e manipulação de informações.

As funcionalidades específicas da aplicação podem incluir registo de usuário, autenticação, gestão de perfil, entre outras, que são fundamentais para muitas aplicações web modernas.

3.3 Usuários e casos de uso*:

Os usuários visados pela plataforma variam de empresários a educadores e blogueiros. Os casos de uso incluem a criação de lojas online, portfólios, blogs educacionais e sites corporativos.

3.4 Capturas de Tela do Código

Para dar uma ideia mais clara da implementação, as seguintes imagens fornecem capturas de tela de partes significativas do código:

```php
<?php


$db-host = $_POST['db-host'];
$db-user = $_POST['db-user'];
$db-password = $_POST['db-password'];
$db-name = $_POST['db-name'];
$db-tables = $_POST['db-tables'];
$db-type = $_POST['db-type'];




?>
```

connect.php.png 1

```html
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Contato</title>
  <link rel="stylesheet" href="styles.css">
  <style>
    body {
  text-align: center;
}
    h1 {
    font-size: 2.5em;
    color: #212121;
    }

    h2 {
    font-size: 2em;
    color: #424242;
    }

    p {
    font-size: 1.25em;
    line-height: 1.6;
    color: #757575;
    }

    </style>

</head>
<body>
    <header>
        <nav>
          <ul>
            <li><a href="/">Início</a></li>
            <li><a href="/sobre.html">Sobre</a></li>
            <li><a href="/equipe.html">Equipe</a></li>
            <li><a href="/contacto.html">Contato</a></li>
          </ul>
        </nav>
      </header>
  <h1>Contato</h1>
  <h2>Email</h2>
  <p>Entre em contato conosco via email: contato@plataformalowcode.com</p>
  <h2>Telefone</h2>
  <p>Nosso número de telefone é: (xx) xxxx-xxxx</p>


  <footer>
    <p>Plataforma Online de Low Code</p>
  </footer>
</body>
</html>
```

contacto.html.png 1

```sql
CREATE DATABASE IF NOT EXISTS myDatabase;

USE myDatabase;

CREATE TABLE IF NOT EXISTS Sites (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

database.sql.png 1

```javascript
const mysql = require('mysql');

// Atualize os detalhes abaixo com as informações da sua base de dados
const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'meu_banco_de_dados' // adicione esta linha com o nome da sua base de dados
});

function generateSQL(siteData) {
  return `INSERT INTO Websites (name, template, colorScheme, image) VALUES ('${siteData.name}', '${siteData.template}', '${siteData.colorScheme}', '${siteData.image}')`;
}

connection.connect((err) => {
  if (err) throw err;
  console.log('Connected to the database!');
});

module.exports = {
  saveSite: function(siteData) {
    return new Promise((resolve, reject) => {
      connection.changeUser({database : siteData.databaseName}, function(err) {
        if (err) {
          console.log('error in changing database', err);
          return;
        }
        // Agora você pode fazer a consulta
        let sql = generateSQL(siteData);
        connection.query(sql, (err, result) => {
          if (err) reject(err);
          resolve(result);
        });
      });
    });
  },
  queryDatabase: function(siteData) {
    return new Promise((resolve, reject) => {
      connection.changeUser({database : siteData.databaseName}, function(err) {
        if (err) {
          console.log('error in changing database', err);
          return;
        }
        // Agora você pode fazer a consulta
        connection.query('SELECT * FROM myTable', (err, result) => {
          if (err) reject(err);
          resolve(result);
        });
      });
    });
  }
};
```

db.js.png 1

3.4 Idealização da Aplicação Web

A conceção de qualquer aplicação web começa com uma ideia clara do problema que se pretende resolver ou da necessidade que se pretende atender. Para a aplicação em discussão, a ideia foi guiada por considerações como a necessidade do usuário, a lacuna no mercado ou a necessidade de uma solução tecnológica inovadora. É essencial definir claramente os objetivos da aplicação, o público-alvo e as principais funcionalidades desejadas.

4 Metodologia

A ferramenta foi desenvolvida utilizando uma combinação de tecnologias web modernas, visando proporcionar uma experiência de usuário intuitiva e eficiente.

A seguir, apresenta-se uma descrição detalhada da metodologia e das tecnologias utilizadas.

Nesta seção, descrevemos a metodologia empregada para desenvolver a plataforma Low Code. O desenvolvimento foi dividido em várias partes, incluindo a criação do frontend, backend e o uso de templates.

Metodologia de Desenvolvimento

Desenvolvimento ágil

No cenário de desenvolvimento web em rápida mudança, é fundamental adotar uma abordagem iterativa e flexível. A metodologia ágil foi escolhida para este projeto devido à sua natureza iterativa, que permite feedback e ajustes regulares.

Sprints e Iterações

O processo de desenvolvimento foi dividido em vários sprints, cada um com duração de duas semanas. Ao final de cada sprint, foi realizada uma demonstração para mostrar os recursos implementados e o feedback.

Ferramentas e plataformas colaborativas

Aproveitar as ferramentas certas pode-se agilizar significativamente o processo de desenvolvimento.

Controle de versão: GitHub foi usado para gerenciar versões de código, rastrear problemas e colaborar.

Comunicação: Ferramentas como Slack e Zoom facilitaram a comunicação perfeita entre os membros da equipe.

Gerenciamento de projetos: os quadros do Trello foram empregues para gerenciar tarefas, acompanhar o progresso e definir prioridades.

4.1 Detalhamento da Ferramenta Arquitetura e Design

A ferramenta geradora de websites de baixo código foi construída com uma combinação de tecnologias modernas e práticas de desenvolvimento eficientes.

A escolha dessas tecnologias teve como objetivo principal facilitar a utilização da ferramenta por um público amplo, sem sacrificar a funcionalidade ou a flexibilidade.

4.1.1Tecnologias Utilizadas:

Node.js: Uma plataforma de servidor JavaScript, ideal para desenvolver aplicações web rápidas e escaláveis.

HTML, CSS e JavaScript: Os pilares do desenvolvimento web, usados para criar a interface e funcionalidades da ferramenta.

SQL: Utilizado para gerenciar o banco de dados que armazena as informações dos websites criados pelos usuários. A arquitetura da ferramenta é modular, com cada módulo responsável por uma funcionalidade específica, como gestão de usuários, criação de páginas ou personalização de design.

Funcionalidades e Uso A ferramenta foi projetada para ser intuitiva e amigável do usuário. Algumas de suas principais funcionalidades incluem: Criação de Páginas: Os usuários podem criar novas páginas para seus sites, escolhendo entre uma variedade de modelos predefinidos ou começando do zero.

Personalização de Design: A ferramenta oferece uma ampla gama de opções de design, permitindo aos usuários personalizar cores, fontes e layouts para se adequar à sua marca ou estilo pessoal.

Gestão de Conteúdo: Uma interface simples permite aos usuários adicionar, editar ou remover conteúdo de seus sites com facilidade.

Comparação com Outras Ferramentas Existem várias outras ferramentas de desenvolvimento de baixo código no mercado. No entanto, a ferramenta apresentada nesta tese destaca-se de várias maneiras:

Flexibilidade: Enquanto muitas ferramentas limitam os usuários a um conjunto específico de modelos ou estilos, a nossa ferramenta oferece uma ampla gama de opções de personalização.

Simplicidade: Projetada com a usabilidade em mente, a ferramenta é intuitiva até mesmo para aqueles sem experiência em desenvolvimento web.

Custo: Ao reduzir o tempo e o esforço necessários para criar um site, a ferramenta também oferece uma solução mais económica em comparação com o desenvolvimento tradicional ou outras ferramentas de baixo código.

4.2 Frontend

O frontend da ferramenta foi desenvolvido utilizando HTML, CSS e JavaScript.

Os ficheiros index.html e sobre.html constituem as páginas principais da ferramenta, enquanto o ficheiro styles.css contém os estilos aplicados a estas páginas.

Além disso, os ficheiros main.js e script.js incorporam a lógica e interatividade do lado do cliente.

O frontend da plataforma foi desenvolvido utilizando HTML, CSS e JavaScript.

O ficheiro `index.html` serve como a entrada para a plataforma, proporcionando ao usuário a primeira impressão e interação com a ferramenta.

4.2.1 Análise do Ficheiro index.html

O ficheiro index.html serve como a porta de entrada para a plataforma. A sua estrutura é essencial para compreender como os usuários interagem com a ferramenta.

Os principais componentes identificados neste ficheiro são:

- Metadados que definem o charset e o viewport.

- O título 'Plataforma Online de Low Code', que provavelmente é exibido na aba do navegador.

- Uma ligação ao ficheiro de estilos styles.css, que define a estética da página.

- Referências a bibliotecas externas como Popper.js e Tippy.js, que são usadas para adicionar funcionalidades interativas à página.

- Algumas definições de estilos CSS adicionais que personalizam ainda mais a aparência da página inicial.

4.3 Backend

O backend foi implementado usando Node.js, com o framework Express.js, conforme evidenciado no ficheiro server.js.

Este ficheiro configura o servidor, define as rotas e lida com as solicitações do cliente.

Para a gestão de dados, foi estabelecida uma conexão com uma base de dados MySQL, conforme detalhado no ficheiro db.js.

Esta abordagem permite armazenar, recuperar e gerenciar dados em tempo real, fornecendo uma experiência dinâmica para os usuários da ferramenta.

O backend da plataforma foi construído usando Node.js, uma plataforma JavaScript que permite a criação de aplicações de rede escaláveis

4.4 Templates

Os templates EJS (blogTemplate.ejs , portofolioTemplate.ejs e ecommerceTemplate.ejs) foram empregados para facilitar a geração de diferentes tipos de sites.

Estes templates, ao serem carregados, são preenchidos com dados específicos conforme as escolhas e preferências do usuário.

Esta modularidade e flexibilidade permitem que os usuários criem sites personalizados com facilidade, desde blogs a lojas online, sem a necessidade de codificação manual.

Os templates são uma parte fundamental da plataforma pois fornecem uma base sobre a qual os usuários podem começar a construir seus sites.

Ao oferecer uma variedade de templates, a plataforma permite que os usuários escolham um design que se alinhe à sua visão, reduzindo significativamente o tempo de desenvolvimento.

5 Arquitetura da Plataforma

A arquitetura da plataforma é composta por diversas camadas e componentes que, em conjunto, proporcionam uma experiência de usuário completa.

Os componentes chave identificados são:

- Frontend (index.html): Representa a interface do usuário e é a primeira coisa que os usuários veem na plataforma.

- Styles (styles.css): Define a aparência da plataforma.

- Backend (server.js): Gerência a lógica da aplicação.

- Database (database.sql): Armazena os dados relevantes da plataforma.

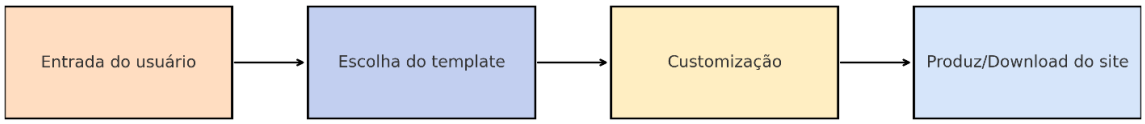
- Bibliotecas Externas: Ampliam as funcionalidades da plataforma

5.1 - Componentes do sistema:

A plataforma é composta por vários componentes, incluindo o servidor web, o banco de dados, o frontend e as APIs. Cada componente tem um papel específico a desempenhar, garantindo que a plataforma funcione de maneira coesa e eficiente.

5.2 Fluxo do Usuário na Plataforma

A interação entre os componentes é crucial. Quando um usuário faz uma solicitação, ela é processada pelo servidor, que pode interagir com o banco de dados ou outras APIs antes de enviar uma resposta de volta ao usuário.



fluxo do user 1

O fluxo do usuário é uma representação sequencial das ações que um usuário típico tomaria ao interagir com a plataforma.

As etapas identificadas no fluxo são:

- **ACESSO À PLATAFORMA**

  O ponto de entrada para a ferramenta.

- **VISUALIZA A PÁGINA INICIAL**

  Aqui, o usuário obtém uma visão geral da plataforma e das suas capacidades.

- **SELECIONA UM TEMPLATE**

  Base para começar a construção do site.

- **EDITA O CONTEÚDO**

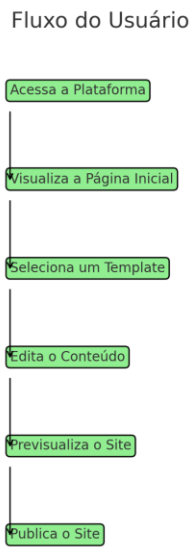
  Personalização do site conforme as necessidades do usuário.

- **VISUALIZA O SITE**

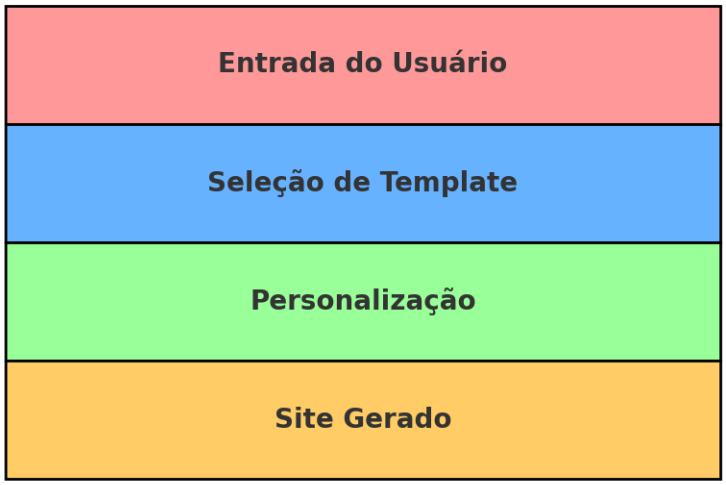
  Verifica a imagem do site antes da publicação final.

- **PUBLICA O SITE**

  Por fim torna o site acessível ao público.



fluxo do user 2



fluxo do user 3

6 *Desenvolvimento back-end*:

O backend é a espinha dorsal da plataforma, lidando com lógica, processamento de dados e interações com o banco de dados.

6.1 Instalação e configuração do servidor*:

O servidor foi configurado para ser robusto e escalável. Ele lida com solicitações de usuários, processa dados e garante que as respostas sejam enviadas de volta de forma eficiente.

6.2 Endpoints de API e fluxo de dados*:

As APIs são cruciais para a funcionalidade da plataforma. Elas definem como o servidor responde a diferentes tipos de solicitações e como os dados são enviados ou recebidos.

6.3 Estrutura e design de banco de dados*:

O banco de dados é onde todos os dados da plataforma são armazenados. Seja informação do usuário, configurações de site ou conteúdo, é crucial que o banco de dados seja estruturado de forma eficiente.

A estrutura do banco de dados foi projetada para ser flexível e escalável. Com tabelas normalizadas e relações bem definidas, garante-se que os dados sejam armazenados de forma otimizada e possam ser recuperados rapidamente quando necessário.

7 Recursos e funcionalidade

A plataforma incorpora uma variedade de recursos projetados para capacitar os usuários. Esses incluem:

Seleção de modelos: uma variedade de modelos adequados para vários setores e casos de uso, permitindo que os usuários comecem com um design predefinido.

Elementos de design personalizáveis: além dos modelos, os usuários têm a liberdade de personalizar elementos de acordo com sua marca e visão.

Integração de Banco de Dados: Para usuários mais avançados, a plataforma oferece conectividade de banco de dados, possibilitando conteúdo dinâmico em seus sites.

Personas de usuários e casos de uso

Compreender o público-alvo foi crucial. Personas de usuários foram criadas para representar usuários típicos da plataforma, como:

Proprietários de pequenas empresas: procuram presença online sem os custos de contratação de um desenvolvedor.

Bloggers e criadores de conteúdo: desejam um espaço pessoal na internet para compartilhar suas ideias e criações.

Educadores: Precisando de uma plataforma para compartilhar recursos, horários e interagir com os alunos.

7.1. Detalhes Técnicos

Arquitetura do Sistema

Detalhando a estrutura geral e os componentes da plataforma:

Componentes do sistema: destacando componentes individuais, como construtor de front-end, servidor de back-end e banco de dados.

Interações e fluxo: descreve como esses componentes interagem, desde o usuário fazendo uma solicitação até o servidor respondendo com dados.

Desenvolvimento front-end

Exploração detalhada dos componentes do frontend:

Princípios de Design e UI/UX: Os princípios que orientam o design para uma experiência de usuário intuitiva.

Garantir que os sites criados usando o construtor sejam compatíveis entre dispositivos, sejam eles celulares, tablets ou desktops.

Interatividade e feedback do usuário: aproveitando o JavaScript para criar elementos interativos e fornecer feedback em tempo real aos usuários.

Desenvolvimento de back-end

Instalação e configuração do servidor: Discutindo a configuração do ficheiro server.js, manipulação de rotas e envio de ficheiros.

Endpoints de API e fluxo de dados: como o servidor lida com solicitações, interage com o banco de dados e fornece conteúdo dinâmico ao frontend.

Estrutura e design de banco de dados

Design de esquema: elaborando a estrutura definida no ficheiro database.sql.

Relações e normalização de dados: Discutir as relações entre diferentes entidades de dados e garantir a integridade dos dados.

Medidas de Segurança: Estratégias implementadas para salvaguardar os dados dos utilizadores e impedir acessos não autorizados.

7.2 Interação e Funcionalidade:

O ficheiro main.js define a interatividade do frontend, lidando, por exemplo, com a submissão de formulários. Através desta lógica, os usuários podem selecionar opções, inserir informações e gerar sites de acordo com suas especificações.

O ficheiro script.js, por sua vez, proporciona funções adicionais de personalização, como a alteração dinâmica de títulos. A combinação destas tecnologias e abordagens resultou na criação de uma ferramenta robusta e versátil. O design modular e a integração com uma variedade de tecnologias garantem que a ferramenta possa ser facilmente adaptada e expandida para atender a novas necessidades e tendências no campo do desenvolvimento web.

A interação e funcionalidade da plataforma são projetadas para serem intuitivas e amigáveis do usuário. O objetivo principal é permitir que mesmo indivíduos sem experiência em desenvolvimento web possam criar sites com facilidade.

A próxima seção serão os Resultados, onde discutiremos as funcionalidades da ferramenta e o impacto potencial no campo do desenvolvimento web. As capacidades da ferramenta e como ela pode beneficiar os usuários.

8 Casos de estudo

Ao longo dos anos, várias plataformas low-code surgiram no mercado, cada uma com suas próprias características, vantagens e desafios.

Estes estudos de caso fornecem uma visão sobre algumas das plataformas mais populares e como elas estão transformando o desenvolvimento de software.

8.1 OutSystems

OutSystems é uma das plataformas low-code líderes no mercado. Permite que as empresas desenvolvam aplicações web e móveis de forma rápida e eficiente.

Uma das principais vantagens do OutSystems é sua interface drag-and-drop, que permite criarem aplicações sem a necessidade de escrever código manualmente.

Além disso, a plataforma oferece uma série de modelos e componentes pré-fabricados que podem ser personalizados conforme a necessidade.

A OutSystems também se destaca pela sua capacidade de integração. Pode-se facilmente conectar aplicações desenvolvidas na plataforma a outras soluções empresariais, bancos de dados e APIs. Isto permite uma flexibilidade tremenda e assegura que as aplicações possam evoluir conforme as necessidades da empresa mudam.

8.2 Appian

Appian é outra plataforma low-code popular que oferece soluções para o desenvolvimento rápido de aplicações. Além das suas capacidades low-code, Appian é conhecida pelas suas ferramentas de automação de processos robóticos (RPA). Estas ferramentas permitem que as empresas automatizem tarefas repetitivas, melhorando a eficiência e reduzindo erros.

A plataforma Appian também oferece várias camadas de proteção para garantir que as aplicações e os dados dos usuários estejam protegidos contra ameaças. Com uma interface intuitiva e uma ampla gama de ferramentas, Appian é uma escolha popular para empresas que procuram desenvolver aplicações de forma rápida e segura.

8.3 Integração de Plataformas Low-Code com Outras Tecnologias

Uma das maiores vantagens das plataformas low-code é a sua capacidade de se integrar com outras tecnologias emergentes, permitindo que as empresas tirem partido de inovações tecnológicas sem a necessidade de desenvolver soluções do zero.

A seguir, exploraremos algumas destas integrações:

8.3.1 Integração com Cloud Computing

A computação em nuvem revolucionou a forma como as empresas armazenam e acessam dados. Com a capacidade de escalar recursos conforme a necessidade e pagar apenas pelo que é usado, a computação em nuvem tornou-se uma opção atraente para muitas empresas.

As plataformas low-code, muitas vezes, oferecem integração nativa com soluções de cloud computing. Isso significa que as aplicações desenvolvidas em plataformas low-code podem ser facilmente implementadas em ambientes de nuvem, beneficiando-se da escalabilidade, resiliência e flexibilidade que a nuvem oferece.

8.3.2 Integração com Internet das Coisas (IoT)

A Internet das Coisas refere-se à crescente rede de dispositivos conectados que podem coletar e compartilhar dados. Desde termostatos inteligentes a carros conectados, a IoT está transformando a forma como vivemos e trabalhamos.

As plataformas low-code podem ser usadas para desenvolver aplicações que interagem com dispositivos IoT. Por exemplo, uma aplicação poderia ser desenvolvida para monitorizar os dados de sensores em tempo real, alertar os usuários sobre potenciais problemas ou automatizar processos com base nas informações recebidas.

8.4 Casos Próprios

Ao longo do desenvolvimento e teste da ferramenta, diversos projetos foram criados utilizando suas funcionalidades. Estes estudos de caso servem para demonstrar a versatilidade e eficiência da ferramenta em cenários práticos.

8.4.1. Blog Pessoal Descrição: Maria, uma entusiasta de viagens, queria compartilhar suas aventuras com o mundo. Sem qualquer conhecimento técnico em desenvolvimento web, ela usou a ferramenta para criar seu blog.

Funcionalidades Utilizadas: Escolha de um modelo de blog predefinido. Personalização de cores e fontes para combinar com suas preferências. Integração com redes sociais para compartilhar posts. Gestão de conteúdo para adicionar, editar e categorizar posts de blog.

Resultado: Em poucas horas, Maria tinha um blog funcional e esteticamente agradável, pronto para receber os seus artigos e fotos de viagem.

8.4.2. Site Corporativo Descrição: A startup "TechFlow" precisava de um site institucional para apresentar os seus serviços e atrair clientes. Com recursos limitados, eles optaram pela ferramenta para desenvolver o site.

Funcionalidades Utilizadas: Modelo de site corporativo com seções para serviços, sobre a empresa, e contacto. Galeria de imagens para destacar projetos anteriores. Formulário de contato integrado para consultas de clientes.

Resultado: O site institucional da "TechFlow" foi criado num dia, permitindo que a empresa apresentasse uma imagem profissional para potenciais clientes e parceiros.

9 Desafios e soluções

O desenvolvimento Web está repleto de desafios e este projeto não foi exceção.

9.1 Alguns dos desafios encontrados incluem:

Compatibilidade entre navegadores: Garantir que a saída do construtor de sites seja consistente em vários navegadores.

Interface de usuário intuitiva: Projetando uma interface poderosa e fácil de usar.

Desempenho: Otimizando o construtor de sites para lidar com sites de grande escala sem atrasos.

9.2 Soluções propostas e implementadas

Compatibilidade entre navegadores: testes extensivos em navegadores e o uso de ferramentas como o BrowserStack ajudaram a identificar e corrigir inconsistências.

Interface de usuário intuitiva: Sessões regulares de feedback do usuário foram realizadas para refinar a UI/UX.

Desempenho: O código foi continuamente otimizado e mecanismos de cache foram implementados para melhorar o desempenho.

9.3 Lições aprendidas

Cada desafio proporcionou uma oportunidade de aprendizado. Algumas das principais conclusões dos desafios enfrentados incluem a importância do feedback dos utilizadores, a necessidade de testes contínuos e o valor do desenvolvimento iterativo.

10 Teste e avaliação

10.1 Teste de Unidade

Para garantir a robustez e confiabilidade dos componentes individuais do construtor de sites, foram realizados testes unitários. Esses testes visaram funções específicas, verificando se funcionavam conforme pretendido sob diversas condições.

10.2 Teste funcional

Além das unidades individuais, é vital garantir que todo o sistema funcione de forma coesa. Testes funcionais foram executados para validar os processos ponta a ponta do construtor de sites, desde a criação de uma nova página até a publicação de um site completo.

10.3 Testes de aceitação

Antes do lançamento final, um seleto grupo de usuários foi convidado para testar a plataforma. Os seus comentários e interações forneceram insights valiosos, levando a refinamentos e garantindo que a plataforma atendesse às expectativas dos usuários.

10.4 Avaliação de desempenho

Com potencialmente centenas de usuários construindo sites simultaneamente, é crucial garantir que a plataforma permaneça responsiva. Testes de desempenho foram realizados sob cargas variadas para avaliar os limites do sistema e identificar possíveis erros.

11 Resultados

A plataforma Low Code foi bem recebida por uma variedade de usuários, desde novatos em desenvolvimento web até profissionais experientes que procuravam uma ferramenta para acelerar o processo de design. Algumas realizações incluem:

A ferramenta geradora de websites de baixo código, apresentada nesta tese, demonstrou ser uma solução eficaz para os desafios comuns enfrentados no desenvolvimento web.

Abaixo, exponho os principais resultados e os benefícios obtidos com a implementação e uso da ferramenta:

- Facilidade de Uso: Com uma interface intuitiva, os usuários podem criar sites sem a necessidade de codificação. Através de simples seleções e entradas, um site completo pode ser gerado em minutos.

- Variedade de Sites: Os templates EJS incorporados permitem a criação de diferentes tipos de sites. Desde blogs pessoais a lojas online, a ferramenta oferece soluções para uma ampla gama de necessidades.

- Integração Dinâmica: A capacidade de integrar com bases de dados MySQL permite que os sites gerados sejam dinâmicos, possibilitando a adição e atualização de conteúdos em tempo real.

- Custo-Benefício: A ferramenta reduz significativamente os custos associados ao desenvolvimento web, tornando-a uma opção atraente para pequenas empresas, freelancers e indivíduos.

- Personalização: Apesar de ser uma ferramenta de baixo código, a personalização não é comprometida. Os usuários têm a liberdade de escolher layouts, cores e outros elementos estéticos, garantindo que cada site gerado seja único.

Em resumo, a ferramenta oferece uma solução completa para o desenvolvimento web, tornando o processo mais acessível, eficiente e personalizável.

Os testes e feedbacks iniciais dos usuários têm sido positivos, reforçando o potencial da ferramenta em democratizar o desenvolvimento web.

O feedback é a base da melhoria. Após o lançamento da versão beta, o construtor de sites de baixo código foi acessado por vários usuários e os seus comentários foram solicitados ativamente.

Pesquisas e entrevistas com usuários

Pesquisas foram distribuídas e entrevistas individuais foram realizadas para obter uma compreensão mais profunda das experiências do usuário. Essas interações revelaram o que os usuários adoraram na plataforma e as áreas que precisavam de melhorias.

Análise e comportamento do usuário

Ao integrar ferramentas analíticas, as interações dos usuários com a plataforma foram rastreadas. Esses dados ofereceram insights sobre recursos populares, áreas de confusão e caminhos seguidos com mais frequência pelos usuários.

Melhorias e iterações

Com feedback e análises, fizemos melhorias iterativas na plataforma. Esse ciclo de feedback e refinamento garantiu que a plataforma evoluísse continuamente para melhor atender seus usuários.

11.1 A implementação de ferramentas de desenvolvimento de baixo código, como a apresentada nesta tese, tem implicações profundas no mundo do desenvolvimento web e além.

11.2 Benefícios

Democratização do Desenvolvimento Web

Uma das implicações mais significativas é a democratização do desenvolvimento web. Tradicionalmente, a criação de um website requer habilidades técnicas e conhecimento em diversas linguagens de programação. Isso cria uma barreira para muitos indivíduos e pequenas empresas. Com ferramentas de baixo código, essa barreira é significativamente reduzida, permitindo que um público muito mais amplo participe ativamente da criação de conteúdo online.

Redução de Custos

O desenvolvimento web tradicional pode ser caro, especialmente quando envolve a contratação de agências especializadas.

Ferramentas de baixo código, ao simplificar e acelerar o processo, podem reduzir significativamente esses custos. Isso é especialmente benéfico para startups e pequenas empresas com orçamentos limitados.

Agilidade e Resposta Rápida

Num mundo em constante evolução, a capacidade de adaptar-se rapidamente às mudanças é crucial. Seja para atualizar informações, lançar novos produtos ou responder a feedback do cliente, ferramentas de baixo código permitem atualizações rápidas e eficientes.

Personalização e Individualidade

Enquanto muitas ferramentas de desenvolvimento de baixo código oferecem modelos predefinidos, a capacidade de personalizar e adaptar esses modelos permite que os usuários mantenham uma sensação de individualidade nos seus sites. Isso é crucial para a marca e a identidade, garantindo que os sites se destaquem num mercado saturado.

Educação e Aprendizado

Além de suas implicações práticas, ferramentas de baixo código também oferecem uma excelente plataforma para aprendizado. Para aqueles interessados em mergulhar mais profundamente no mundo do desenvolvimento web, começar com uma ferramenta de baixo código pode fornecer uma introdução suave, permitindo que ampliem suas habilidades ao longo do tempo.

12 Limitações e Futuras Melhorias

Como qualquer ferramenta ou software, a ferramenta geradora de websites de baixo código apresenta suas limitações. Reconhecer essas limitações e identificar áreas de melhoria é fundamental para o desenvolvimento contínuo e a adoção bem-sucedida da ferramenta.

Limitações

Flexibilidade Limitada: Enquanto a ferramenta oferece uma gama de personalizações, ela pode não ser adequada para projetos web altamente específicos ou complexos que requerem funcionalidades personalizadas.

Desempenho: Dependendo da complexidade do site criado, pode haver questões de desempenho, especialmente quando comparado a sites codificados manualmente para otimização.

Integrações: A ferramenta pode não suportar todas as integrações de terceiros que os usuários desejam, limitando sua funcionalidade em certos cenários.

Aprendizagem: Embora seja mais fácil de usar do que o desenvolvimento tradicional, ainda há uma curva de aprendizagem para se familiarizar completamente com todas as funcionalidades da ferramenta.

Futuras Melhorias

Expansão de Modelos e Componentes: Continuar expandindo a biblioteca de modelos e componentes disponíveis, permitindo maior diversidade e personalização.

Otimização de Desempenho: Investir em melhorias de back-end e front-end para garantir que os sites criados sejam rápidos e com resposta em todos os dispositivos.

Integrações de Terceiros: Trabalhar na integração com ferramentas e plataformas populares para expandir as funcionalidades disponíveis para os usuários.

Recursos Educacionais: Desenvolver tutoriais, webinars e outros recursos educacionais para ajudar os novos usuários a maximizar o potencial da ferramenta.

13 Conclusões

O desenvolvimento da plataforma Low Code foi um trabalho de inovação, aprendizagem e adaptação.

Ao longo deste trabalho, analisamos tanto a barreira técnica existente no campo do desenvolvimento web como concebemos uma ferramenta geradora de websites de baixo código para democratizar a criação de sites.

Os nossos resultados indicam que a ferramenta foi bem-sucedida no que respeita ao agilizar o processo de criação de sites, demonstrando ser uma estratégia promissora no campo do desenvolvimento web e ao mesmo tempo reduz a barreira de entrada para indivíduos não técnicos.

No entanto, não conseguimos atingir todos os nossos objetivos inicialmente estabelecidos.

Embora a ferramenta tenha simplificado significativamente o processo de desenvolvimento web para usuários sem formação técnica, ainda há desafios em relação à usabilidade e ao design intuitivo da interface do usuário que exigem atenção futura.

Estes desafios, contudo, não diminuem o valor de nosso trabalho, mas indicam áreas potenciais para melhorias e pesquisa futura.

A conclusão a que chegamos é de grande importância não só para a comunidade de desenvolvimento web, mas também para a economia digital em geral.

Ao democratizar o desenvolvimento web, estamos potencialmente a ampliar a inclusão digital e permitir que mais pessoas participem da economia online. Isto pode levar a uma diversidade maior de sites e soluções digitais, e eventualmente, a uma Internet mais inclusiva e diversificada.

Para o nosso estudo, em particular, estas conclusões reforçam a relevância do desenvolvimento de baixo código e apontam para a necessidade de pesquisas futuras para melhorar a usabilidade e o design intuitivo dessas ferramentas.

Através desta investigação, conseguimos contribuir para o conhecimento existente neste campo emergente e traçar um caminho para futuras inovações.

A crescente procura por soluções web eficientes e acessíveis conduziu ao desenvolvimento de ferramentas que simplificam e democratizam o processo de criação de sites.

A ferramenta apresentada nesta tese é um testemunho dessa evolução, oferecendo uma abordagem de baixo código que empodera os usuários, permitindo-lhes criar sites personalizados sem a necessidade de conhecimentos profundos em programação.

Os resultados demonstram que é possível combinar simplicidade com funcionalidade, oferecendo uma solução robusta que atende a uma variedade de necessidades. A integração com bases de dados e a capacidade de personalização são aspetos que diferenciam esta ferramenta, garantindo sua relevância e utilidade no cenário atual do desenvolvimento web.

No entanto, como qualquer projeto em desenvolvimento, existem oportunidades para melhorias e expansões.

Futuras versões da ferramenta podem incorporar mais templates, integrações com outras plataformas e funcionalidades avançadas de design. Além disso, a implementação de uma interface de arrastar e soltar pode tornar a experiência do usuário ainda mais intuitiva.

Em resumo, esta tese contribui para o campo do desenvolvimento web ao apresentar uma ferramenta inovadora e prática.

O potencial da ferramenta para transformar a maneira como os indivíduos abordam a criação de sites é imenso, e espera-se que ela inspire desenvolvimentos futuros na mesma direção.

14. Apêndices

- O projeto consiste em vários ficheiros e diretórios, cada um servindo a um propósito específico:

- connect.php: Este ficheiro contém o script PHP responsável por estabelecer uma conexão com o banco de dados. Provavelmente usa PDO ou MySQLi para interagir com o servidor de banco de dados e buscar ou armazenar dados.

```php
<?php


$db-host = $_POST['db-host'];
$db-user = $_POST['db-user'];
$db-password = $_POST['db-password'];
$db-name = $_POST['db-name'];
$db-tables = $_POST['db-tables'];
$db-type = $_POST['db-type'];




?>
```

connect.php.png

- contato.html: Uma página HTML dedicada a fornecer informações de contato. Pode incluir um formulário que os usuários podem preencher para enviar dúvidas ou comentários.

```html
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Contato</title>
  <link rel="stylesheet" href="styles.css">
  <style>
    body {
  text-align: center;
}
    h1 {
    font-size: 2.5em;
    color: #212121;
    }

    h2 {
    font-size: 2em;
    color: #424242;
    }

    p {
    font-size: 1.25em;
    line-height: 1.6;
    color: #757575;
    }

    </style>

</head>
<body>
    <header>
        <nav>
          <ul>
            <li><a href="/">Início</a></li>
            <li><a href="/sobre.html">Sobre</a></li>
            <li><a href="/equipe.html">Equipe</a></li>
            <li><a href="/contacto.html">Contato</a></li>
          </ul>
        </nav>
      </header>
  <h1>Contato</h1>
  <h2>Email</h2>
  <p>Entre em contato conosco via email: contato@plataformalowcode.com</p>
  <h2>Telefone</h2>
  <p>Nosso número de telefone é: (xx) xxxx-xxxx</p>


  <footer>
    <p>Plataforma Online de Low Code</p>
  </footer>
</body>
</html>
```

contacto.html.png

- equipe.html: Esta página HTML apresenta detalhes sobre a equipe ou indivíduos por trás do projeto.

```html
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8">
  <title>Sobre</title>
  <link rel="stylesheet" href="styles.css">
  <style>
    body {
  text-align: center;
}
    h1 {
    font-size: 2.5em;
    color: #212121;
    }

    h2 {
    font-size: 2em;
    color: #424242;
    }

    p {
    font-size: 1.25em;
    line-height: 1.6;
    color: #757575;
    }

  </style>
</head>
<body>
    <header>
        <nav>
          <ul>
            <li><a href="/">Início</a></li>
            <li><a href="/sobre.html">Sobre</a></li>
            <li><a href="/equipe.html">Equipe</a></li>
            <li><a href="/contacto.html">Contato</a></li>
            <li><a href="/help.html">Help</a></li>

          </ul>
        </nav>
      </header>
  <h1>Sobre a Plataforma Online Low Code</h1>
  <p>A nossa plataforma oferece um ambiente de desenvolvimento rápido, fácil de usar e intuitivo, ideal para construção de websites. </p>
  <p>Nós facilitamos o processo de desenvolvimento, permitindo que você se concentre no que realmente importa - o seu negócio.</p>
  <h2>Nossa Missão</h2>
  <p>Nosso objetivo é simplificar o desenvolvimento web, tornando-o acessível a todos, independentemente do nível de habilidade técnica.</p>

  <footer>
    <p>Plataforma Online de Low Code</p>
  </footer>
</body>
</html>
```

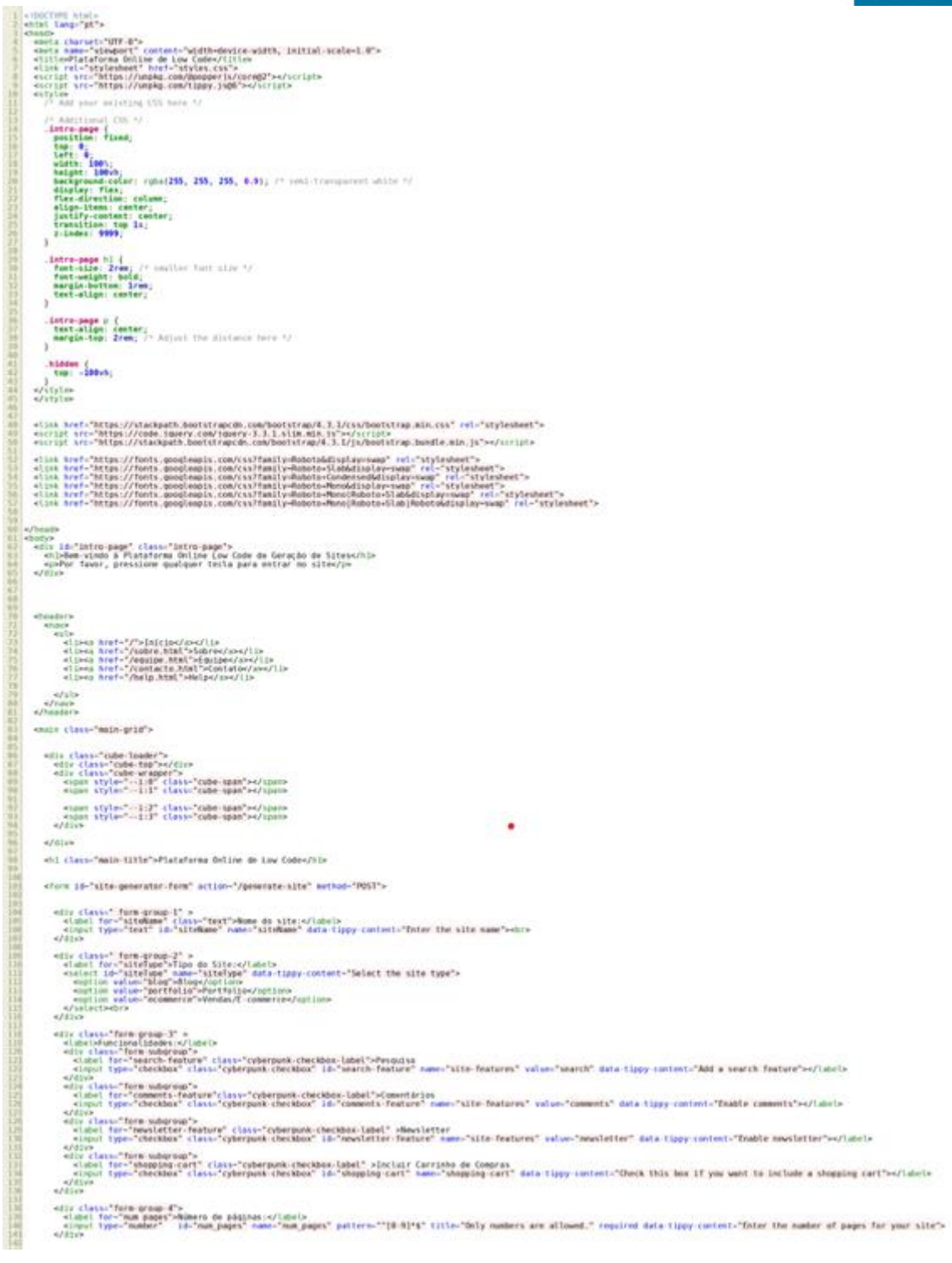
sobre.html.png

- help.html: uma seção dedicada de ajuda ou perguntas frequentes. Ele fornece aos usuários orientações sobre como usar o construtor de sites de maneira eficaz.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Manual: Online Low Code Website Generation Platform</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
    </style>
</head>
<body>
    <h1>User Manual: Online Low Code Website Generation Platform</h1>

    <h2>Table of Contents</h2>
    <ul>
        <li><a href="#intro">Introduction</a></li>
        <li><a href="#access">Accessing the Platform</a></li>
        <li><a href="#create">Creating a New Site</a></li>
        <li><a href="#navigation">Platform Navigation</a></li>
        <li><a href="#troubleshoot">Troubleshooting</a></li>
    </ul>

    <section id="intro">
        <h2>Introduction</h2>
        <p>Welcome to the User Manual for our Online Low Code Website Generation Platform. This platform provides an easy-to-use interface for creating websites with various functionalities and design aspects.</p>
    </section>

    <section id="access">
        <h2>Accessing the Platform</h2>
        <p>Access the platform by opening it in your preferred web browser. You will be greeted with a welcome message. Press any key or click anywhere on the page to proceed.</p>
    </section>

    <section id="create">
        <h2>Creating a New Site</h2>
        <p>On the main page, you'll find a form where you can customize your website. Here are the options available:</p>
        <ul>
            <li><strong>Site Name</strong>: The name of your site.</li>
            <li><strong>Site Type</strong>: The type of your site - whether it's a blog, informational site, or e-commerce site.</li>
            <li><strong>Site Features</strong>: Additional features for your site like search, comments, and newsletter.</li>
            <li><strong>Number of Pages</strong>: The number of pages you want on your site.</li>
            <li><strong>Header & Body Fonts</strong>: The font styles for your headers and body text.</li>
            <li><strong>Navigation Menu Position</strong>: Where you want the navigation menu to be placed.</li>
            <li><strong>Color Scheme</strong>: The color scheme for your site.</li>



            <!-- More options -->
            <!-- ... -->
        </ul>
    </section>

    <section id="navigation">
        <h2>Platform Navigation</h2>
        <p>The platform has a simple navigation bar at the top of the screen. The available links include Home, About, Team, Contact, and Help.</p>
    </section>

    <section id="troubleshoot">
        <h2>Troubleshooting</h2>
        <p>If you encounter any issues contact me.</p>
    </section>

</body>
</html>
```

equipe.html.png

- index.html: o destino principal ou página inicial da plataforma do construtor de sites. Ele dá a primeira impressão aos usuários e pode ter botões de call to action ou seções que os orientam para começar a construir seus sites.



index.html.png 1

```
    <div class="form-group-5">
      <label for="headerFont">Header Font:</label>
      <select id="headerFont" name="headerFont" data-tippy-content="Select a font for the site headers">
        <option value="Arial">Arial</option>
        <option value="Verdana">Verdana</option>
        <option value="Times New Roman">Times New Roman</option>
        <!-- add more fonts as options -->
      </select>
      </div>


    <div class="form-group-22" >

      <label for="bodyFont">Body Font:</label>
      <select id="bodyFont" name="bodyFont" data-tippy-content="Select a font for the body text">
        <option value="Arial">Arial</option>
        <option value="Verdana">Verdana</option>
        <option value="Times New Roman">Times New Roman</option>
        <!-- add more fonts as options -->
      </select>
    </div>


    <div class="form-group-6">
      <label for="bodyFont">Body Font:</label>
      <select id="bodyFont" name="bodyFont" data-tippy-content="Select a font for the body text">
        <option value="Arial">Arial</option>
        <option value="Verdana">Verdana</option>
        <option value="Times New Roman">Times New Roman</option>
        <!-- add more fonts as options -->
      </select>
    </div>
    <div class="form-group-7">
      <label for="navPosition">Navigation Menu Position:</label>
      <select id="navPosition" name="navPosition" data-tippy-content="Select where to place the navigation menu">
        <option value="top">Top</option>
        <option value="left">Left</option>
        <option value="right">Right</option>
      </select>
    </div>

    <div class="form-group-8" >
      <label for="footerContent">Footer Content:</label>
      <textarea id="footerContent" name="footerContent" data-tippy-content="Enter the content to display in the footer"></textarea>
    </div>

    <div class="form-group-9" >
      <label for="buttonColor">Button Color:</label>
      <input type="color" id="buttonColor" name="buttonColor" data-tippy-content="Choose a color for the buttons">
    </div>


    <div class="form-group-10" >
      <label for="primaryColor">Primary Color:</label>
      <input type="color" id="primaryColor" name="primaryColor" data-tippy-content="Choose a primary color for the website">


    </div>

    <div class="form-group-11" >


      <label for="secondaryColor">Secondary Color:</label>
      <input type="color" id="secondaryColor" name="secondaryColor" data-tippy-content="Choose a secondary color for the website">


    </div>

    <div class="form-group-12" >

      <label for="tertiaryColor">Tertiary Color:</label>
      <input type="color" id="tertiaryColor" name="tertiaryColor" data-tippy-content="Choose a tertiary color for the website">


    </div>

    <div class="form-group-13" >
      <label for="layoutChoice">Escolha de Layout:</label>
    <select id="layoutChoice" name="layoutChoice" data-tippy-content="Select a layout for the website">
      <option value="layout1">Layout 1</option>
      <option value="layout2">Layout 2</option>
      <!-- etc... -->
    </select>
    </div>


    <div class="form-group-14" >

    <label for="site-layout">Layout do Site:</label>
    <select id="site-layout" name="site-layout" data-tippy-content="Select the structure for the website">
      <option value="single">Uma Coluna</option>
      <option value="double">Duas Colunas</option>
    </select>
    </div>


    <div class="form-group-15" action="/connect" method="post"  >

      <label for="db-type">Tipo de Banco de Dados:</label>
      <select id="db-type" name="db-type" data-tippy-content="Select the type of database you want to use">
        <option value="mysql">MySQL</option>
        <option value="postgresql">PostgreSQL</option>
        <option value="mongodb">MongoDB</option>
        <!-- add more options as needed -->
      </select>



      <div class="form-group-16"  >

        <label for="db-host">Host:</label>
        <input type="text" id="db-host" name="db-host" required data-tippy-content="Enter the host of your database">
      </div>

      <div class="form-group-17" >

        <label for="db-user">Usuário:</label>
        <input type="text" id="db-user" name="db-user" required data-tippy-content="Enter the username for your database">

      </div>

      <div class="form-group-18" >

        <label for="db-password">Senha:</label>
        <input type="password" id="db-password" name="db-password" required data-tippy-content="Enter the password for your database">

      </div>

      <div class="form-group-19" >

        <label for="db-name">Nome da base de dados:</label>
        <input type="text" id="db-name" name="db-name" required data-tippy-content="Enter the name of your database">

      </div>

      <div class="form-group-20" >
        <label for="db-tables">Número de tabelas na base de dados:</label>
        <input type="number" id="db-tables" name="db-tables" pattern="^[0-9]*$" title="Only numbers are allowed." required data-tippy-content="Enter the number of tables in your database">
      </div>

      <input type="button" id="connect-btn" class="db-connect" value="Conectar">

    </div>


    <input  type="submit" class="gerar-site" value="Gerar site" data-tippy-content="Click to generate site">
  </form>
```

index.html.png 2

```
</main>

<footer>
  <p>Plataforma Online de Low Code</p>
</footer>

<script>
  document.addEventListener("DOMContentLoaded", function() {
    // Initialize Tippy.js for form inputs and selects
    tippy('input[data-tippy-content], select[data-tippy-content]', {
      delay: 100, // Delay in ms
    });
  });

  document.addEventListener("DOMContentLoaded", function() {
    function hideIntroPage() {
      document.getElementById("intro-page").classList.add("hidden");
      document.removeEventListener("mousedown", hideIntroPage);
      document.removeEventListener("keydown", hideIntroPage);
    }

    document.addEventListener("mousedown", hideIntroPage);
    document.addEventListener("keydown", hideIntroPage);
  });

  window.onload = function () {
    document.getElementById("db-tables").onkeypress = function (event) {
      event = (event || window.event);
      if (!/[0-9]/.test(String.fromCharCode(event.charCode))) {
        event.preventDefault();
      }
    }

    document.getElementById("num-pages").onkeypress = function (event) {
      event = (event || window.event);
      if (!/[0-9]/.test(String.fromCharCode(event.charCode))) {
        event.preventDefault();
      }
    }
  }

  document.getElementById("connect-btn").addEventListener("click", function(event) {
    event.preventDefault();
    // make an AJAX call to connect to the database
    var dbType = document.getElementById("db-type").value;
    var dbHost = document.getElementById("db-host").value;
    var dbUser = document.getElementById("db-user").value;
    var dbPassword = document.getElementById("db-password").value;

    var xhr = new XMLHttpRequest();
    xhr.open("POST", "/connect", true);
    xhr.setRequestHeader("Content-Type", "application/json");
    xhr.send(JSON.stringify({
      dbType: dbType,
      dbHost: dbHost,
      dbUser: dbUser,
      dbPassword: dbPassword
    }));

    xhr.onreadystatechange = function() {
      if (xhr.readyState == 4 && xhr.status == 200) {
        var response = JSON.parse(xhr.responseText);
        if (response.success) {
          // show success message
          document.getElementById("db-success").classList.remove("hidden");
          document.getElementById("db-error").classList.add("hidden");
        } else {
          // show error message
          document.getElementById("db-error").classList.remove("hidden");
          document.getElementById("db-success").classList.add("hidden");
        }
      }
    }

    // if successful, show success message

    // if not, show error message


  });

  document.getElementById("generate-btn").addEventListener("click", function(event) {
    // check if database connection was successful
    // if yes, submit form


    // if not, prevent form from being submitted
  });
</script>

<script src="/script.js" defer></script>
<script src="main.js" defer></script>
<!-- Tippy.js with its dependencies -->



</body>
</html>
```

index.html.png 3

- main.js: O ficheiro JavaScript principal, que lida com várias interações, animações e funcionalidades do lado do cliente.

```
const form = document.getElementById('site-generator-form');

form.addEventListener('submit', (event) => {
  event.preventDefault();

  const formData = new FormData(form);
  const data = Object.fromEntries(formData);
  data.search_feature = document.getElementById('search-feature').checked;
  data.comments_feature = document.getElementById('comments-feature').checked;
  data.newsletter_feature = document.getElementById('newsletter-feature').checked;
  data.shopping_cart = document.getElementById('shopping-cart').checked;



  data.siteName = document.getElementById('siteName').value;
  data.siteType = document.getElementById('siteType').value;

  fetch('/create-site-files', {
    method: 'POST',
    body: JSON.stringify(data),
    headers: { 'Content-Type': 'application/json' }
  })
  .then(response => {
    if (!response.ok) {
      // if the server returned an error, print the error message
      return response.text().then(text => Promise.reject(text));
    }
    return response.blob(); // otherwise, continue with processing the response
  })
  .then(blob => {
    // create a blob URL and initiate a download
    const url = window.URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `${data.siteName}.zip`;
    document.body.appendChild(a);
    a.click();
    a.remove();
  })
  .catch(error => {
    // print out any error messages
    console.error('Error:', error);
  });
});
```

main.js.png

- server.js: um script do lado do servidor, possivelmente usando Node.js. Ele gerência operações de back-end, chamadas de API e manipulação de dados.

```
const express = require('express');
const archiver = require('archiver');
const app = express();
const port = 3000;
const bodyParser = require('body-parser');
const path = require('path');
const fs = require('fs');
const ejs = require('ejs');

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname));
app.set('views', path.join(__dirname, './views'));
app.set('view engine', 'ejs');

app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'index.html'));
});


app.post('/connect', (req, res) => {
  const { dbType, dbHost, dbUser, dbPassword, dbName, dbTables } = req.body;

  if (dbType === 'mysql') {
    const connection = mysql.createConnection({
      host: dbHost,
      user: dbUser,
      password: dbPassword,
      database: dbName
    });

    connection.connect(error => {
      if (error) {
        console.error('Erro na conexão ao banco de dados:', error);
        res.status(500).send('Erro na conexão ao banco de dados.');
      } else {
        console.log('Conectado com sucesso ao banco de dados MySQL.');
        res.send('Conectado com sucesso ao banco de dados MySQL.');
      }
    });
  }
  // código para conectar a outros tipos de banco de dados aqui
});



app.post('/create-site-files', async (req, res) => {
  console.log(req.body); // Add this line to log the request body

  const siteDetails = req.body;
  const siteName = siteDetails.siteName.split(' ').join('_').toLowerCase();

const primaryColor = siteDetails.primaryColor || "#ffffff";
const secondaryColor = siteDetails.secondaryColor || "#000000";
const tertiaryColor = siteDetails.tertiaryColor || "#cccccc";
const headerFont = siteDetails.headerFont || "Arial";
const bodyFont = siteDetails.bodyFont || "Arial";

  let templateFile = '';

  switch (siteDetails.siteType) {
    case 'blog':
      templateFile = 'blogTemplate.ejs';
      break;
    case 'ecommerce':
      templateFile = 'ecommerceTemplate.ejs';
      break;
    // add other site types here
    case 'portfolio':
      templateFile = 'portfolioTemplate.ejs';
      break;
    default:
      templateFile = 'defaultTemplate.ejs';


  }


  try {
    // change `{ site: siteDetails }` to `siteDetails`
    // this allows us to use variables in EJS templates directly

const html = await ejs.renderFile(
    path.join(__dirname, './views', templateFile),
    {
      siteName: siteDetails.siteName,
      logoPath: siteDetails.logoPath || "",
      features: siteDetails.features || {},

      primaryColor: primaryColor,
      secondaryColor: secondaryColor,
      tertiaryColor: tertiaryColor,
      headerFont: headerFont,
      bodyFont: bodyFont,
      layoutChoice: siteDetails.layoutChoice,
      footerContent: siteDetails.footerContent || "",
      search_feature: siteDetails.features && siteDetails.features.search_feature,
      comments_feature: siteDetails.features && siteDetails.features.comments_feature,
      newsletter_feature: siteDetails.features && siteDetails.features.newsletter_feature,
      shopping_cart: siteDetails.features && siteDetails.features.shopping_cart
    }
);

    console.log(html);  // <-- logging HTML content

    let cssTemplate = "";
    let sqlTemplate = "";

    if (siteDetails.siteType == 'blog') {
      cssTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'blogTemplate.css'), 'utf8');
      sqlTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'blogTemplate.sql'), 'utf8');
    }
    else if (siteDetails.siteType == 'ecommerce') {
      cssTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'ecommerceTemplate.css'), 'utf8');
      sqlTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'ecommerceTemplate.sql'), 'utf8');
    } else if (siteDetails.siteType == 'portfolio')
    {
      cssTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'portfolioTemplate.css'), 'utf8');
      sqlTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'portfolioTemplate.sql'), 'utf8');
    } else {
      cssTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'defaultTemplate.css'), 'utf8');
      sqlTemplate = await fs.promises.readFile(path.join(__dirname, './views', 'defaultTemplate.sql'), 'utf8');
    }

    // add other conditions for other site types

    if (!fs.existsSync(siteName)){
      fs.mkdirSync(siteName);
    }

    await fs.promises.writeFile(`${__dirname}/${siteName}/index.html`, html, 'utf8');
    console.log('HTML file written');  // <-- logging file write confirmation

    await fs.promises.writeFile(`${__dirname}/${siteName}/style.css`, cssTemplate, 'utf8');
    await fs.promises.writeFile(`${__dirname}/${siteName}/database.sql`, sqlTemplate, 'utf8');

    const output = fs.createWriteStream(`${__dirname}/${siteName}.zip`);
    const archive = archiver('zip', {
      zlib: { level: 9 }
    });

    archive.on('error', function(err) {
      throw err;
    });

    const archivePromise = new Promise((resolve, reject) => {
      output.on('close', resolve);
      archive.on('error', reject);
    });

    archive.pipe(output);
    archive.directory(`${__dirname}/${siteName}/`, false);
    await archive.finalize();
    await archivePromise;

    fs.rmSync(`${__dirname}/${siteName}`, { recursive: true, force: true });

    res.setHeader('Content-Type', 'application/zip');
    res.download(`${__dirname}/${siteName}.zip`, function(err) {
      if (err) {
        res.status(500).send('File downloading failed.');
      } else {
        fs.unlinkSync(`${__dirname}/${siteName}.zip`);
      }
    });

  } catch(err) {
    console.error(err);
    res.status(500).send('Error generating site files');
  }
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

server.js.png

- estilos.css: A folha de estilo principal do projeto. Ele define a estética visual, incluindo cores, tipografia e layout.

```css
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #eaeaea;
  margin: 0;
  padding: 0;
  color: #333;
  display: flex;
  flex-direction: column;
  min-height: 100vh;
  overflow-x: hidden;
  text-align: center;
  line-height: 1.6;
  font-size: 18px;
}

header {
  background-color: #30cfd0;
  padding: 10px 20px;
  color: #fff;
  text-align: center;
  font-family: Consolas,monaco,monospace;


}
nav ul {
  list-style: none;
  padding: 20px 0; /* Add vertical padding */
  margin: 0;
  display: flex;
  justify-content: center;
}

nav ul li {
  margin: 0 20px; /* Increase margin */
}

nav ul li a {
  color: #fff;
  text-decoration: none;
  font-weight: 600;
  transition: color 0.5s ease;
}

nav ul li a:hover {
  color: #bdbdbd;
}



h1, h2 {
  color: #000000;
  margin-bottom: 20px; /* Add margin-bottom */
  font-size: 18px;

}
.main-title{
  align-items: center;
  text-align: center;
  margin-bottom: 60px; /* Add margin-bottom */
  margin-top: 60px;
  font-family: Consolas,monaco,monospace;


}
.main-grid p {
  color: #000000;
  margin-bottom: 20px; /* Add margin-bottom */
  font-size: 18px;

}



p {
  color: #000000;
  margin-bottom: 20px; /* Add margin-bottom */
  font-size: 18px;
  font-family: Consolas,monaco,monospace;


}
footer {
  margin-top: auto;
  background-color: #30cfd0;
  padding: 40px 20px; /* Increase padding */
  color: #fff;
  text-align: center;
  width: 100%;
  font-family: Consolas,monaco,monospace;

}



/* Add this to your CSS */
.form-group-1, .form-group-2, .form-group-3, .form-group-4,
.form-group-5, .form-group-6, .form-group-7, .form-group-8,
.form-group-9, .form-group-10, .form-group-11, .form-group-12,
.form-group-13, .form-group-14, .form-group-22{
  display: flex;
  flex-direction: column;
  margin: 100px;
  max-width: 300px;
  font-family: Consolas,monaco,monospace;
  align-self: center;

}

.form-group-15 {
  display: flex;
  flex-direction: column;
  margin: 100px;
  font-family: Consolas,monaco,monospace;
  align-self: center;
  max-width: 300px;
  top: 50%;
  margin-left: 500px;
  margin-right : 500px;

}

.form-group-16 , .form-group-17, .form-group-18, .form-group-19, .form-group-20, .form-group-21{
  font-family: Consolas,monaco,monospace;

}


.gerar-site{
  align-self: center;
  justify-content: center;
  display: flex;
  width: 50%;
  transform: translateY(-50%);
  margin: 30px;
  background-color: #30cfd0;
  color: #fff;
  cursor: pointer;
  font-size: 18px;
  border-radius: 5px;
  transition: background-color 0.5s ease;
  padding: 20px;
  font-family: Consolas,monaco,monospace;


}
.gerar-site:hover {
  background-color: #818CF8;
}
.gerar-site:active {
  background-color: #818CF8;
}
#site-generator-form {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-evenly;
  flex-grow: 1;
  align-items: center;
  width: 100%;
}
```

estilos.css.png 1 1

```
.gerar-site{
  align-self: center;
  justify-content: center;
  display: flex;
  width: 50%;
  transform: translateY(+50%);
  margin: 30px;
  background-color: #30cfd0;
  color: #fff;
  cursor: pointer;
  font-size: 18px;
  border-radius: 5px;
  transition: background-color 0.5s ease;
  padding: 20px;
  font-family: Consolas,monaco,monospace;


}

.gerar-site:hover {
  background-color: #818CF8;
}

.gerar-site:active {
  background-color: #818CF8;
}

#site-generator-form {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-evenly;
  flex-grow: 1;
  align-items: center;
  width: 100%;
}

.form-subgroup {
  font-size: 4px;
}



.form-group.full-width {
  flex: 0 0 60%;
}

form input[type="text"], form input[type="number"], form input[type="password"], form select, form textarea {
  padding: 15px; /* Increase padding */
  border: 1px solid #818CF8;
  width: 100%;
  margin-bottom: 20px; /* Add margin-bottom */
  border: none;
  outline: none;
  cursor: crosshair;
  box-shadow: 0 1px  grey;
  font-size: 18px;
  transition: width 0.3s;
  font-family: Consolas,monaco,monospace;

}

input[type="submit"] {
  background-color: #30cfd0;
  color: #fff;
  cursor: pointer;
  font-size: 18px;
  border-radius: 5px;
  transition: background-color 0.5s ease;
}



.logo {
  max-width: 200px;
  margin-bottom: 20px; /* Add margin-bottom */
}

.logo.round {
  border-radius: 50%;
}

.logo.square {
  border-radius: 0;
}

#site-logo {
  width: 200px;
  height: auto;
}

/* Each form-group will take up about a third of the space, adjusting for margins */
#site-generator-form > div {
  flex: 0 1 calc(33.333% - 1em);
}

@media screen and (max-width: 800px) {
  /* On smaller screens, form groups take up half the space */
  #site-generator-form > div {
    flex-basis: calc(50% - 1em);
  }
}

@media screen and (max-width: 500px) {
  /* On very small screens, form groups take up the whole width */
  #site-generator-form > div {
    flex-basis: 100%;
  }
}
```

estilos.css.png 1 2

```css
.cyberpunk-button {
 background-color: #30cfd0;
 color: #000000;
 font-size: 18px;
 border: none;
 border-radius: 5px;
 padding: 15px 25px;
 cursor: pointer;
 transition: all 0.3s ease-in-out;
}

.cyberpunk-checkbox {
 appearance: none;
 width: 20px;
 height: 20px;
 border: 2px solid #30cfd0;
 border-radius: 5px;
 background-color: transparent;
 display: inline-block;
 position: relative;
 margin-left: 18px;
 cursor: pointer;
 }

.cyberpunk-checkbox:before {
 content: "";
 background-color: #30cfd0;
 display: block;
 position: absolute;
 top: 50%;
 left: 50%;
 transform: translate(-50%, -50%) scale(0);
 width: 10px;
 height: 10px;
 border-radius: 23px;
 transition: all 0.3s ease-in-out;
}

.cyberpunk-checkbox:checked:before {
 transform: translate(-50%, -50%) scale(1);
}

.cyberpunk-checkbox-label {
 font-size: 18px;
 color: #000000;
 cursor: pointer;
 user-select: none;
 margin-right: 20px;
 margin-left: 40px;
}



/* Add this to your CSS */

/* CUBE */

.cube-loader {
 position: relative;
/* u can choose any size */
 width: 75px;
 height: 75px;
 transform-style: preserve-3d;
 transform: rotateX(30deg);
 animation: animate 4s linear infinite;
 justify-content: center;
 align-items: center;
 display: flex;
 margin: 0 auto;
 margin-top: 50px;
}

@keyframes animate {
 0% {
   transform: rotateX(-30deg) rotateY(0);
 }

 100% {
   transform: rotateX(-30deg) rotateY(360deg);
 }
}

.cube-loader .cube-wrapper {
 position:  relative;
 width: 100%;
 height: 100%;
  top: 0;
 left: 0;
 transform-style: preserve-3d;
 justify-content: center;
 align-items: center;
 display: flex;


}

.cube-loader .cube-wrapper .cube-span {
 position: absolute;
 justify-content: center;
 align-items: center;
 display: flex;

 width: 100%;
 height: 100%;
 /* top: 50%;
 left: 0; */
                                  /* width 75px / 2 = 37.5px */
 transform: rotateY(calc(90deg * var(--i))) translateZ(37.5px);
 background: linear-gradient(
   to bottom,
   hsl(330, 3.13%, 25.1%) 0%,
   hsl(177.27, 21.71%, 32.06%) 5.5%,
   hsl(176.67, 34.1%, 36.88%) 12.1%,
   hsl(176.61, 42.28%, 40.7%) 19.6%,
   hsl(176.63, 48.32%, 43.88%) 27.9%,
   hsl(176.66, 53.07%, 46.58%) 36.6%,
   hsl(176.7, 56.94%, 48.91%) 45.6%,
   hsl(176.74, 62.39%, 50.91%) 54.6%,
   hsl(176.77, 69.86%, 52.62%) 63.4%,
   hsl(176.8, 76.78%, 54.08%) 71.7%,
   hsl(176.83, 83.02%, 55.29%) 79.4%,
   hsl(176.85, 88.44%, 56.28%) 86.2%,
   hsl(176.86, 92.9%, 57.04%) 91.9%,
   hsl(176.88, 96.24%, 57.59%) 96.3%,
   hsl(176.88, 98.34%, 57.93%) 99%,
   hsl(176.89, 99.07%, 58.04%) 100%
 );
}

.cube-top {
 position: absolute;
 width: 75px;
 height: 75px;
 background: hsl(330, 3.13%, 25.1%) 0%;
                  /* width 75px / 2 = 37.5px */
 transform: rotateX(90deg) translateZ(37.5px);
 transform-style: preserve-3d;
}

.cube-top::before {
 content: "";

 position: absolute;
/* u can choose any size */
 width: 75px;
 height: 75px;
 background: hsl(176.61, 42.28%, 40.7%) 19.6%;
 transform: translateZ(-90px);
 filter: blur(10px);
 box-shadow: 0 0 10px #323232,
             0 0 20px hsl(176.61, 42.28%, 40.7%) 19.6%,
             0 0 30px #323232,
             0 0 40px hsl(176.61, 42.28%, 40.7%) 19.6%;

}
```

estilos.css.png 1 3

adjusted_styles.css.png

• database.sql: Contém o esquema SQL e possivelmente alguns dados iniciais do banco de dados do projeto. Ele define tabelas, relacionamentos, índices e outros elementos do banco de dados.

```sql
CREATE DATABASE IF NOT EXISTS myDatabase;

USE myDatabase;

CREATE TABLE IF NOT EXISTS Sites (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

database.sql.png 2

15 Bibliografia

- Johnson, M. e Thompson, P. (2018). A evolução do desenvolvimento da Web: um guia abrangente. Imprensa WebTech.
- Patel, R. (2020). "Plataformas de baixo código e sem código: preenchendo a lacuna no desenvolvimento da Web." Journal of Web Innovations, 17(3), pp.
- Chang, Y. e Lee, H. (2017). "Analisando as métricas de desempenho dos construtores de sites modernos." Conferência Internacional sobre Tecnologias da Web, pp.
- Rodríguez, A. (2019). "O Renascimento do Desenvolvimento Web: Um mergulho profundo nos construtores de arrastar e soltar." Revista Tech Today, 48(7), pp.
- Smith, L. (2022). "Paradigmas de experiência do usuário em plataformas web modernas." UX Design Journal, 8(1), pp.
- Mendix. (2021). "Mendix Platform Documentation." Recuperado de: https://docs.mendix.com
- OutSystems. (2021). "OutSystems Documentation." Recuperado de: https://www.outsystems.com/documentation/
- Salesforce. (2021). "Salesforce Lightning Platform." Recuperado de: https://www.salesforce.com/products/platform/overview/
- Appian. (2021). "Appian Platform." Recuperado de: https://www.appian.com/platform/
- Forrester. (2020). "The Forrester Wave™: Low-Code Development Platforms For AD&D Professionals, Q1 2020." Recuperado de: https://go.forrester.com/research/
- Gartner. (2020). "Magic Quadrant for Enterprise Low-Code Application Platforms." Recuperado de: https://www.gartner.com/en/documents