INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA
CAMPUS ARIQUEMES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

WANDERSON SOARES DOS SANTOS

DESENVOLVIMENTO DE UM WEBSITE PARA CONSTRUÇÃO DE PORTFÓLIOS DE PROGRAMADORES COM INTEGRAÇÃO À API DO GITHUB

Ariquemes
2026

WANDERSON SOARES DOS SANTOS

DESENVOLVIMENTO DE UM WEBSITE PARA CONSTRUÇÃO DE PORTFÓLIOS DE PROGRAMADORES COM INTEGRAÇÃO À API DO GITHUB

Relatório técnico entregue como Trabalho Conclusão de Curso ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – (IFRO), Campus Ariquemes, como requisito parcial para obtenção do grau de tecnólogo junto ao curso de Análise e Desenvolvimento de Sistemas.
Orientador: Prof. Especialista Marcos Alves Faino

Ariquemes
2026

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Santos, Wanderson Soares dos.
Desenvolvimento de um website para construção de portfólios de programadores com integração à API do GitHub / Wanderson Soares dos Santos. - Ariquemes, 2026.
39 f. : il.

Orientador(a): Profº. Marcos Alves Faino.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Ariquemes, 2026.

1. Aplicativo Web. 2. Portfólio de programadores. 3. GitHub. 4. Python. 5. Django. I. Faino, Marcos Alves (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Renilce Silva Morais, CRB-11/906

WANDERSON SOARES DOS SANTOS

DESENVOLVIMENTO DE UM WEBSITE PARA CONSTRUÇÃO DE PORTFÓLIOS DE PROGRAMADORES COM INTEGRAÇÃO À API DO GITHUB

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de “Analista e Desenvolvedor de Sistemas” e aprovado em sua forma final pelo Curso de Tecnologia em Análise e Desenvolvimento de Sistemas.

Ariquemes, 19 de Março de 2026.

Banca Examinadora:

gov.br
Documento assinado digitalmente
MARCOS ALVES FAINO
Data: 30/04/2026 13:00:32-0300
Verifique em https://validar.iti.gov.br

Prof. Especialista Marcos Alves Faino
(Orientador)
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

ICP Brasil
Documento assinado digitalmente
ANDREY ALENCAR QUADROS
Data: 30/04/2026 12:43:28-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Andrey Alencar Quadros
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

gov.br
Documento assinado digitalmente
LUCIANO TOPOLNIAK
Data: 30/04/2026 12:34:05-0300
Verifique em https://validar.iti.gov.br

Prof. Mestre Luciano Topolniak
Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO - Campus Ariquemes)

AGRADECIMENTOS

Agradeço primeiramente a Deus, que permitiu e me deu forças para chegar até aqui. Agradeço à minha família, e em especial à minha mãe, que nunca mediu esforços para garantir meus estudos e foi meu principal incentivo para que eu não desistisse.

Ao meu orientador, Prof. Marcos Alves Faino, obrigado pela paciência e ensinamentos durante o desenvolvimento desse trabalho. Agradeço também aos demais professores do curso que contribuíram significativamente para minha formação profissional e conhecimentos ao longo desses anos.

Por fim, agradeço a todos os colegas e amigos que fizeram parte dessa caminhada e ajudaram a superar os desafios, e também a cada um que de alguma forma contribuiu para a realização deste trabalho.

RESUMO

Este relatório apresenta o desenvolvimento de uma aplicação web voltada à criação de portfólios personalizados para programadores, utilizando a integração com o GitHub como principal fonte de dados. A proposta surgiu da necessidade de uma ferramenta prática e intuitiva que facilitasse a organização e exibição de projetos, permitindo aos desenvolvedores destacar seus repositórios de forma visual e acessível. Apesar da existência de soluções similares, muitas carecem de flexibilidade e foco nas reais necessidades dos usuários. Diante disso, foi concebida uma plataforma que automatiza a importação de projetos do GitHub, organiza os repositórios em destaque e disponibiliza informações relevantes sobre cada trabalho. A aplicação conta com design responsivo, visando à usabilidade em diferentes dispositivos. Este documento contempla ainda a contextualização do problema, os objetivos geral e específicos, bem como as metodologias e ferramentas a serem adotadas durante o processo de desenvolvimento. O sistema busca contribuir para a valorização e visibilidade dos programadores, especialmente aqueles em início de carreira.

Palavras-chave: Aplicativo Web; Portfólio de Programadores; GitHub; Python; Django.

ABSTRACT

This report presents the development of a web application aimed at creating personalized portfolios for programmers, using GitHub integration as its main data source. The proposal emerged from the need for a practical and intuitive tool that simplifies the organization and display of projects, allowing developers to highlight their repositories in a visual and accessible way. Although similar solutions exist, many lack the flexibility and user-centric approach required to meet real needs. In response, a platform was designed to automate the import of GitHub projects, organize featured repositories, and provide relevant information about each one. The application will also feature a responsive design to ensure usability across different devices. This document includes the context of the problem, general and specific objectives, as well as the methodologies and tools to be used throughout the development process. The system aims to enhance the visibility and recognition of programmers, especially those at the beginning of their careers.

Keywords: Web Application; Developer Portfolio; GitHub; Python; Django.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CDN	*Content Delivery Network*
CSS	*Cascading Style Sheets*
DRY	*Don't Repeat Yourself*
HTML	*HyperText Markup Language*
IO	*Input/Output*
MVC	*Model-View-Controller*
MVT	*Model-View-Template*
OAuth	*Open Authorization*
ORM	*Object-Relational Mapping*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
TI	Tecnologia da Informação
UI	*User Interface*
URL	*Uniform Resource Locator*
UX	*User Experience*

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho tem o objetivo de abordar o desenvolvimento de um aplicativo *web* para apoio na construção e gerenciamento de portfólios digitais de programadores. Apesar de existirem repositórios públicos em plataformas como o GitHub, muitos desenvolvedores encontram dificuldades para apresentar seus projetos de modo organizado e atrativo a recrutadores e outros profissionais, o que pode comprometer sua visibilidade profissional e diminuir oportunidades de inserção no mercado de trabalho. Justifica-se, portanto, a criação de soluções que facilitem organizar e apresentar projetos técnicos em ambientes digitais.

O portfólio profissional é um elemento-chave na trajetória de um programador: organiza e evidencia habilidades, projetos e contribuições concretas, além de favorecer o *networking* com recrutadores e outros profissionais. Com a digitalização do mercado de trabalho, especialmente no setor de TI (Tecnologia da Informação), o portfólio digital passou a ser um requisito para apresentar, de forma clara e estruturada, as competências técnicas e criativas do desenvolvedor.

Com base nesse contexto, o presente projeto tem como foco o desenvolvimento de um aplicativo *web*, utilizando a linguagem de programação Python e o *framework* Django, para auxiliar programadores na construção e gerenciamento de seus portfólios. A aplicação permitirá a integração com o GitHub, possibilitando a importação de projetos, a organização de repositórios destacados e a exibição de informações detalhadas sobre cada projeto. Dessa forma, o sistema visa proporcionar uma maneira prática e eficiente de criar portfólios personalizados, melhorando a visibilidade e a apresentação do trabalho dos desenvolvedores.

1.1 OBJETIVO GERAL

Desenvolver uma aplicação *web* capaz de gerar portfólios personalizados a partir dos dados do GitHub, promovendo a visibilidade de desenvolvedores por meio de páginas responsivas e fáceis de configurar.

1.2 OBJETIVOS ESPECÍFICOS

1. Implementar autenticação segura via OAuth (*Open Authorization*) com o GitHub e importar metadados dos repositórios do usuário.
2. Permitir a seleção, organização e destaque de projetos para composição do portfólio.
3. Oferecer templates responsivos e opções de personalização visual (foto, cores, tipografia e descrições).

4. Modelar a arquitetura do sistema garantindo escalabilidade e eficiência na integração com a API do GitHub.
5. Projetar e implementar o esquema de banco de dados para armazenar perfis de usuários e configurações de portfólios.
6. Desenvolver uma interface de usuário intuitiva e acessível, facilitando a navegação e a personalização dos portfólios.

1.3 ABRANGÊNCIA DO TRABALHO

O escopo limita-se a repositórios públicos do GitHub e ao desenvolvimento de uma versão *web*; não contempla, nesta etapa, integração com outras plataformas (ex: GitLab) nem importação de repositórios privados.

2 REVISÃO BIBLIOGRÁFICA

2.1 O PAPEL DO PORTFÓLIO NA CARREIRA DE TI

Os métodos tradicionais de avaliação curricular na área de TI (Tecnologia da Informação), ainda que utilizados, são insuficientes para demonstrar a capacidade técnica real de um profissional. O currículo lista competências, mas o portfólio as comprova. Nesse sentido, o portfólio digital se torna uma ferramenta indispensável, que serve como uma vitrine para projetos, habilidades e metodologias aplicadas.

O portfólio tem como função principal validar o conhecimento que o profissional admite possuir. Para desenvolvedores de *software*, a demonstração de sistemas funcionais provê uma validação de competências técnicas superior à argumentação verbal, o que se consolida como uma estratégia de *marketing* pessoal altamente eficaz.

Um projeto com código acessível, descrição clara do que foi realizado e qual problema foi resolvido, consegue oferecer uma evidência clara de competência que um certificado, curso ou lista de tecnologias não é capaz de fornecer. Essa abordagem é atestada por autores como Sonmez (2020), que defende a criação de um portfólio e autopromoção como ferramentas essenciais para o marketing pessoal do desenvolvedor de *software*.

Além de validar as competências técnicas do programador, o portfólio serve como um componente que representa a marca pessoal do profissional. No mundo atual, a presença online de um desenvolvedor, seja com seu portfólio, perfil no Linkedin e contribuições em blogs ou projetos de código aberto, é o que cria sua reputação digital.

O portfólio funciona como uma etapa muito importante nos processos seletivos. Plataformas como o GitHub funcionam como um currículo dinâmico, onde as contribuições e projetos públicos são frequentemente avaliados por recrutadores para formar impressões sobre as habilidades técnicas do desenvolvedor, permitindo uma avaliação mais profunda da qualidade de código e escolha de arquitetura (Marlow; Dabbish; Herbsleb, 2013).

Dessa forma, a construção de um portfólio digital e uma presença online coesa não é mais um diferencial, mas sim um pré-requisito fundamental para a competitividade e o avanço profissional em TI (Tecnologia da Informação).

2.2 A IMPORTÂNCIA DO GITHUB PARA DESENVOLVEDORES

Lançado em 2008, o GitHub superou sua função inicial de ser apenas uma interface web para o sistema de controle de versão Git, criado por Linus Torvalds (Chacon; Straub, 2014).

O GitHub mudou a indústria popularizando o conceito de “codificação social”, ele não funciona só como uma forma de repositório para guardar código, mas também

como uma plataforma para colaboração. Funções como *issues* e, principalmente, o modelo de colaboração baseado em *pull requests*, tornaram-se o padrão da indústria para integrar contribuições de código de forma distribuída, tanto em projetos de código aberto quanto em empresas privadas (Gousios; Pinzger; Deursen, 2014).

Essa plataforma se consolidou como principal lar para comunidade de código aberto, um grande exemplo disso é o aplicativo Telegram, aplicativo de mensagens, que possui código aberto, possibilitando desenvolvedores de “criarem” seu próprio Telegram. Um desenvolvedor pode contribuir em projetos globais, visíveis publicamente e criar uma nova forma de aprendizado e construção de reputação.

A plataforma consolidou-se não apenas como um ambiente de codificação, mas como uma rede social de desenvolvedores, impactando diretamente a forma como equipes colaboram e recrutadores buscam talentos ao redor do mundo (Marlow; Dabbish; Herbsleb, 2013).

Outra vantagem do GitHub é o fato de permitir que recrutadores não precisem se limitar a ler apenas a descrição de projetos, eles podem analisar diretamente o código-fonte desenvolvido. O gráfico de contribuições, que mostra a frequência de atividade do desenvolvedor, junto com projetos que podem ser fixados com base na importância que o desenvolvedor dá para os mesmos, o que fornece uma visão clara da consistência, de tecnologias dominadas e quais são os interesses do profissional.

Portanto, um perfil bem organizado do GitHub funciona como uma ótima vitrine para comprovar as atividades e competências técnicas do desenvolvedor, justificando a necessidade de ferramentas que facilitem a apresentação visual dessas informações para o mercado de trabalho, que é o problema que este trabalho busca solucionar.

2.3 *FRAMEWORKS WEB*: DJANGO E SUAS ALTERNATIVAS

Construir uma aplicação *web* no mundo atual raramente começa do zero. *Frameworks web* fornecem uma estrutura base, com componentes reutilizáveis e uma arquitetura definida, permitindo focar na lógica de negócio da aplicação em vez de "reinventar a roda"(Sommerville, 2011). Encontrar o *framework* é uma decisão de arquitetura crucial que impacta na velocidade de desenvolvimento, performance e principalmente na manutenção do projeto.

O Django, um *framework back-end* de alto nível escrito em Python, se destaca por possuir um vasto conjunto de funcionalidades essenciais prontas para uso (a chamada filosofia "baterias inclusas"), como um sistema de autenticação de usuários, um painel administrativo e um poderoso ORM (*Object-Relational Mapping*). O Django possui a arquitetura MVT (*Model-View-Template*), uma variação do padrão MVC (*Model-View-Controller*), que promove uma clara separação de responsabilidades e estrutura arquitetural robusta (Vincent, 2020):

- ***Model:*** Define a estrutura dos dados, conectando-se ao banco de dados através do ORM (*Object-Relational Mapping*).
- ***View:*** (No Django) É a lógica de negócio que processa a requisição do usuário e retorna uma resposta.
- ***Template:*** É a camada de apresentação, responsável por renderizar a interface para o usuário.

2.3.1 Alternativas ao Django

Flask: É um outro *framework* Python, que segue uma filosofia oposta à do Django. É um "*microframework*"minimalista, fornecendo apenas o essencial para roteamento de URLs e processamento de requisições. Para funcionalidades como ORM (*Object-Relational Mapping*) ou autenticação, o desenvolvedor precisa integrar bibliotecas externas. Isso oferece grande flexibilidade, mas pode aumentar a complexidade de configuração e manutenção em projetos maiores, o que é uma abordagem distinta do Django.

Node.js (com Express.js): No ecossistema do JavaScript, o Node.js é um ambiente de execução que permite rodar JavaScript no back-end. O Express.js é o *framework* minimalista mais popular para ele. Sua principal característica é a arquitetura assíncrona e orientada a eventos, ideal para aplicações que exigem alta concorrência e operações de IO (*Input/Output*) intensivas, como chats em tempo real. Assim como o Flask, ele exige a composição manual de bibliotecas para formar um *stack* completo (ex: ORMs como Prisma ou Sequelize) (Casciaro; Mammino, 2020).

2.3.2 Por que escolher o Django para o GitFólio

A escolha do Django para o desenvolvimento do GitFólio foi prática e baseada nos requisitos da aplicação. O GitFólio precisa de:

- Um sistema robusto para que usuários criem contas e se autentiquem.
- Um banco de dados para persistir as informações do usuário e suas preferências de portfólio (ex: quais projetos exibir).
- Um painel de gerenciamento para facilitar a administração da aplicação.

O Django fornece todas essas funcionalidades de forma nativa, segura e integrada. Seu sistema de autenticação e seu painel administrativo (Django Admin) estão prontos para uso, economizando centenas de horas de desenvolvimento. O ORM (*Object-Relational Mapping*) do Django permitiu modelar os dados da aplicação de forma rápida e segura, abstraindo a complexidade do SQL (*Structured Query Language*).

Embora o Flask ou o Node.js/Express também pudessem ser usados, eles necessitam a integração e configuração manual de múltiplas bibliotecas de terceiros para replicar as funcionalidades que o Django oferece nativamente. Dado o escopo de um Trabalho de Conclusão de Curso, a filosofia do Django permitiu focar no desenvolvimento da lógica de negócio principal do GitFólio — a integração com a API (*Application Programming Interface*) do GitHub e a renderização dos portfólios.

2.4 DESIGN DE INTERFACES E USABILIDADE

Uma aplicação *web* pode ser tecnicamente robusta, mas falhará em atingir seus objetivos se os usuários não conseguirem ou não quiserem interagir com ela. A área que estuda essa interação é dividida em dois conceitos complementares: Experiência do Usuário (UX - *User Experience*) e Interface do Usuário (UI - *User Interface*). De forma simplificada, o UI refere-se aos elementos visuais, enquanto o UX cuida de toda a jornada e dos sentimentos do usuário ao usar o produto (Norman, 2013).

No centro de uma boa Experiência do Usuário está a "usabilidade". O princípio fundamental é o encapsulado por Steve Krug em seu livro "Não Me Faça Pensar"(*Don't Make Me Think*), que postula que as interfaces devem ser autoexplicativas e livres de atrito, permitindo que o usuário realize suas tarefas da forma mais direta possível (Krug, 2014).

O consumo de conteúdo *web* não se restringe mais a desktops, sendo os dispositivos móveis uma parcela majoritária do tráfego. Abordagens como "*Design* Responsivo"(que adapta o *layout* a diferentes telas) e "*Mobile-First*"(que prioriza o *design* em telas pequenas) são cruciais para garantir uma boa experiência em qualquer dispositivo (Krug, 2014).

Para o GitFólio, esses conceitos são cruciais. Um recrutador pode, por exemplo, acessar o portfólio de um candidato rapidamente pelo celular logo após receber um *link*, se o site for de difícil navegação no dispositivo móvel, a oportunidade pode ser perdida.

Portanto, a aplicação de princípios de usabilidade e a adoção de um *design* responsivo com foco em *mobile-first* não são meros detalhes estéticos, mas sim requisitos fundamentais para que o GitFólio cumpra seu objetivo de apresentar eficazmente o perfil do desenvolvedor ao mercado.

3 METODOLOGIA

3.1 METODOLOGIA DE DESENVOLVIMENTO

O processo de desenvolvimento do GitFólio seguiu uma abordagem iterativa e incremental. Essa metodologia foi escolhida para se adequar à natureza de desenvolvimento individual do projeto, permitindo que a aplicação fosse construída e validada em módulos funcionais. Dessa forma, foi possível focar na criação das interfaces básicas, avançar para a integração com a API do GitHub e, a cada etapa concluída, realizar as validações necessárias antes de prosseguir para a próxima funcionalidade.

A primeira fase consistiu na elaboração e análise detalhada dos requisitos. Nesta etapa, foram definidas as funcionalidades essenciais, como o cadastro via GitHub, a importação de repositórios, a gestão de tecnologias e a personalização da interface, incluindo a seção "Sobre Mim"e o *design* responsivo. Requisitos não funcionais, como desempenho, segurança (OAuth) e compatibilidade com dispositivos móveis, também foram minuciosamente estabelecidos.

Após a definição dos requisitos, a fase de modelagem arquitetural e *design* de interfaces foi iniciada utilizando o Figma (ferramenta *online* de *design* colaborativo utilizada para criar interfaces e protótipos). Durante este período, foram criados os diagramas de classes e de entidade-relacionamento do banco de dados no Django, além dos *wireframes* (representações estruturais de baixa fidelidade que mostram a disposição dos elementos de uma interface) e *mockups* (modelos visuais de média ou alta fidelidade que apresentam a aparência final da interface) das telas em Bootstrap. As decisões de UX (*User Experience*) priorizaram a fluidez dos fluxos de navegação e a hierarquia das informações, com o objetivo de garantir uma apresentação clara e intuitiva.

A fase de implementação começou após a validação dos *layouts*. O *backend* foi desenvolvido com Python e o *framework* Django, respeitando boas práticas de código como DRY (*Don't Repeat Yourself*) e o padrão MVT (*Model-View-Template*). No *frontend*, as páginas responsivas foram construídas com HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*) e componentes do Bootstrap, com o uso da *Fetch* API para realizar requisições assíncronas, resultando em um aplicativo mais performático.

Paralelamente, foram realizados testes sequenciais ao final de cada módulo. Testes unitários e de integração validaram as regras de negócio e os *endpoints* REST (*Representational State Transfer*). Testes manuais e de usabilidade, executados em ambiente local, confirmaram o funcionamento da importação de repositórios, dos formulários e da responsividade das páginas. Este ciclo de validação contínuo assegurou a robustez e a qualidade final do sistema. Todas as alterações de código foram adicionadas diretamente na *branch master* no repositório do projeto no GitHub, um fluxo

simplificado adequado para um projeto com um único desenvolvedor e escopo fechado.

3.2 ANÁLISE E LEVANTAMENTO DE REQUISITOS

A fase inicial do projeto concentrou-se na análise e no detalhamento dos requisitos, um processo crucial para garantir que o produto final atendesse às necessidades do público-alvo e resolvesse o problema central. Esta etapa foi dividida em requisitos funcionais (o "o quê"o sistema deve fazer) e requisitos não funcionais (o "como"o sistema deve se comportar).

3.2.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades essenciais do sistema, permitindo que os usuários alcancem seus objetivos. Para o GitFólio, foram identificados os seguintes requisitos:

- **Autenticação de Usuário:** O sistema deve permitir que o usuário se cadastre e acesse sua conta exclusivamente por meio de autenticação via OAuth com o GitHub.
- **Importação de Repositórios:** O usuário deve ser capaz de importar seus repositórios públicos do GitHub para o portfólio. A aplicação deve exibir uma lista de repositórios disponíveis para seleção.
- **Gestão de Projetos:** O sistema deve permitir a adição, edição, remoção e reordenação de projetos no portfólio. Cada projeto deve ter campos editáveis para nome, descrição, imagem e *links* externos.
- **Gerenciamento de Tecnologias:** O usuário deve poder adicionar, remover e classificar as tecnologias que domina (*Frontend* e *Backend*), organizando-as manualmente no portfólio.
- **Personalização de Conteúdo:** O sistema deve permitir que o usuário personalize as seções "Sobre Mim"e "*Hero*", incluindo títulos, descrições, imagens e links para currículo e redes sociais.
- **Configuração Visual:** O usuário deve ter controle sobre as cores do portfólio, com a opção de temas claro e escuro, além de uma paleta de cores personalizada.
- **Publicação do Portfólio:** O sistema deve permitir que o usuário defina seu portfólio como público ou privado. Caso seja público, um link único deve ser gerado para compartilhamento.

3.2.2 Requisitos Não Funcionais

Os requisitos não funcionais descrevem as qualidades e restrições do sistema, garantindo sua performance e segurança. Os requisitos não funcionais levantados foram:

- **Usabilidade:** A interface do usuário deve ser intuitiva e de fácil navegação, permitindo que usuários com pouca familiaridade com ferramentas de portfólio possam usá-la sem dificuldades.
- ***Design* Responsivo:** A aplicação deve se adaptar perfeitamente a diferentes tamanhos de tela (*desktop*, *tablet*, *smartphone*), proporcionando uma experiência de usuário consistente e de alta qualidade em qualquer dispositivo.
- **Segurança:** A autenticação via OAuth com o GitHub deve ser segura, protegendo os dados do usuário. As interações com a API do GitHub e as operações de banco de dados devem ser robustas.
- **Desempenho:** A aplicação deve carregar rapidamente as informações e as imagens, e as interações com a API do GitHub devem ser otimizadas para evitar latência.
- **Compatibilidade:** A aplicação deve ser compatível com os principais navegadores *web*.
- **Manutenção:** O código-fonte deve ser bem estruturado e seguir boas práticas de programação para facilitar futuras manutenções e evoluções.

A partir desses requisitos, foi possível direcionar as fases subsequentes de *design* e desenvolvimento, garantindo que o produto final atendesse a todas as especificações e expectativas iniciais.

3.3 FERRAMENTAS E TECNOLOGIAS UTILIZADAS

O desenvolvimento do projeto foi viabilizado por um conjunto de ferramentas e tecnologias selecionadas para garantir a robustez, escalabilidade e usabilidade do sistema. A escolha de cada tecnologia foi baseada em sua relevância no mercado, documentação e facilidade de integração.

3.3.1 Python

Para a construção do *backend*, a linguagem de programação Python foi a escolha principal, devido à sua versatilidade e ao vasto ecossistema de bibliotecas que permitem o desenvolvimento rápido e eficiente de aplicações *web*. A origem do Python, criada no início dos anos 1990 por Guido van Rossum, fundamenta sua filosofia de ser uma linguagem legível e intuitiva, conforme seu criador destacou: "uma linguagem

fácil e intuitiva enquanto que ainda sendo tão poderosa quanto as maiores competidoras"(Rossum, 1999).

3.3.2 Django

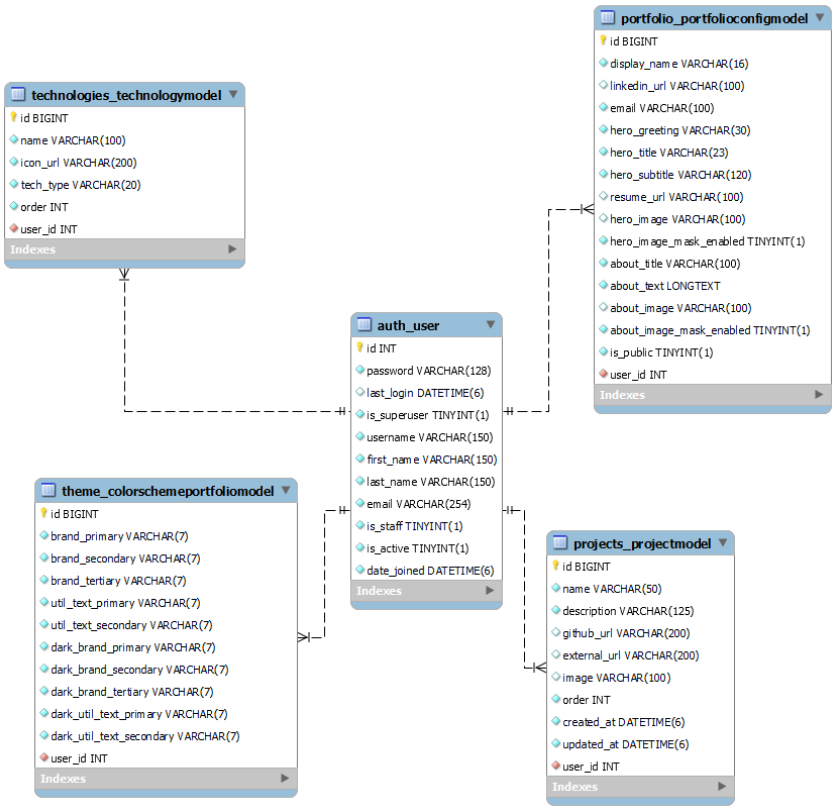
O *backend* do projeto foi estruturado com o *framework* Django, que se destaca por sua filosofia "baterias incluídas". Ele gerencia a comunicação entre o banco de dados e o *frontend*, além de fornecer recursos essenciais para autenticação de usuários, rotas, validação de dados e segurança. A robustez e a ampla documentação do Django foram cruciais para a agilidade do desenvolvimento, permitindo a rápida criação da base do sistema e a fácil integração com outras tecnologias. O sistema de ORM (*Object-Relational Mapping*) do Django facilitou o mapeamento de dados, permitindo a manipulação de informações como usuários, projetos e tecnologias de forma intuitiva.

3.3.3 Banco de Dados (MySQL)

O MySQL foi o banco de dados escolhido para persistir as informações do sistema, como dados de usuários, projetos e configurações. A integração foi realizada por meio do ORM (*Object-Relational Mapping*) do Django, garantindo a integridade e a consistência dos dados. A decisão de utilizar o MySQL também considerou sua ampla aceitação no mercado, confiabilidade e a facilidade de integração com a plataforma de hospedagem PythonAnywhere, onde o GitFólio foi implantado.

A Figura 1 apresenta o modelo de dados definido para o GitFólio, contemplando apenas as tabelas principais.

Figura 1 – Modelo de Dados do GitFólio



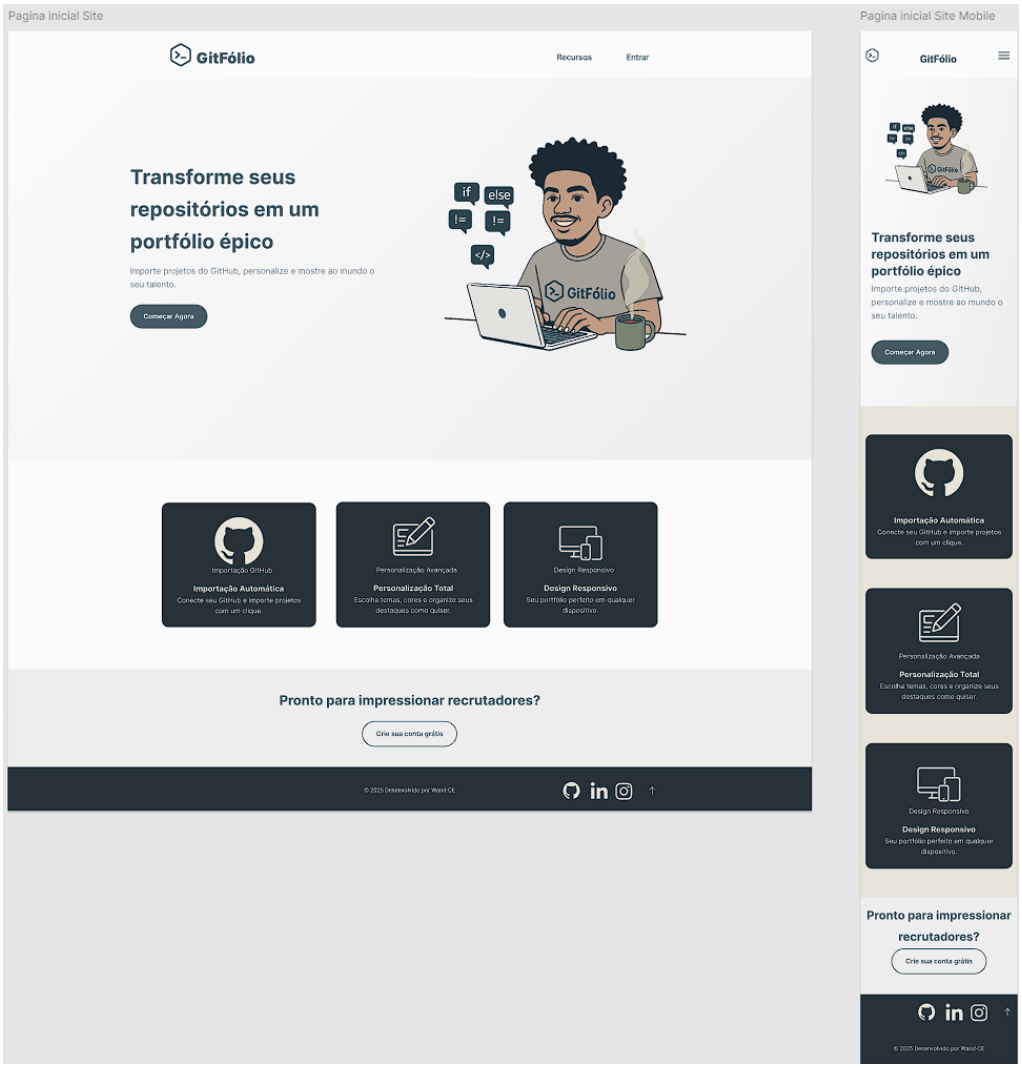
Fonte: Autoria própria (2025).

3.3.4 Figma

Para a fase de *design* de interfaces e prototipagem, o Figma foi a ferramenta utilizada. Com sua origem focada na colaboração em tempo real e na acessibilidade via navegador, a plataforma foi ideal para o esboço inicial do GitFólio. O Figma ofereceu a robustez necessária para a criação de interfaces e a facilidade de prototipagem integrada. Isso permitiu o desenvolvimento ágil de versões para temas claro e escuro, além de *layouts* responsivos para diferentes telas. A utilização de um sistema de componentes reutilizáveis no Figma foi fundamental para garantir a consistência visual e reduzir o retrabalho durante a fase de desenvolvimento, uma vez que o projeto foi conduzido por um único desenvolvedor.

Na Figura 2, podemos ver a tela definida inicialmente como a página de entrada do site, tanto no tamanho Desktop quanto no tamanho *mobile*. O protótipo completo pode ser visualizado no *link*: *Design* Inicial do Projeto

Figura 2 – Esboço da página de entrada do site



Fonte: Autoria própria (2025).

3.3.5 Bootstrap

O Bootstrap é um *framework frontend* (CSS e JavaScript) amplamente utilizado para o desenvolvimento de interfaces *web* responsivas. Ele adota a abordagem *mobile-first*, oferecendo um sistema de *grid*, componentes reutilizáveis e utilitários que facilitam a criação de layouts adaptáveis a diferentes dispositivos.

A escolha do Bootstrap neste projeto deve-se à sua praticidade na construção de interfaces e à facilidade de adaptação para diferentes tamanhos de tela, atendendo diretamente ao requisito de responsividade. Além disso, a disponibilidade de componentes prontos contribui para a padronização visual e reduz o esforço de desenvolvimento, especialmente em comparação com a implementação utilizando apenas CSS e JavaScript puro.

Como exemplo, destaca-se o uso do componente Modal, que permite a exibição

de conteúdos sobrepostos à interface principal. Na Figura 3, é apresentado o modal utilizado para autenticação do usuário por meio da conta do GitHub.

Figura 3 – Modal de login renderizado na página

Fonte: Autoria própria (2025).

3.3.6 API do GitHub

O GitFolio integra-se à API do GitHub para automatizar a busca e exibição dos projetos desenvolvidos pelo usuário. Essa integração foi implementada para facilitar o processo de criação do portfólio, facilitando para o usuário a importação de dados sobre determinado projeto de forma mais rápida e prática. Assim que o usuário está autenticado, na parte de gerenciamento de projetos do usuário, o sistema realiza requisições à API do GitHub utilizando um token. A busca é feita no *endpoint* `/users/{username}/repos`, retornando todos os repositórios públicos associados ao usuário logado, após a busca, os projetos recuperados da API são exibidos automaticamente na interface do portfólio, sem necessidade de intervenção manual. O usuário pode visualizar, selecionar e importar os projetos desejados para o seu portfólio, além de personalizar informações adicionais, como descrições e imagens.

Sempre que o usuário acessa a página de importação de projetos, o sistema realiza uma nova consulta à API do GitHub, garantindo que os projetos exibidos estejam atualizados conforme o perfil do usuário no GitHub. Uma das vantagens da utilização a API no projeto foi a praticidade, onde o usuário não precisa cadastrar manualmente

cada projeto, já que os mesmos já existem publicamente no próprio GitHub, tornando o processo mais ágil e confortável para o usuário.

3.3.7 Cloudinary

O Cloudinary é um serviço de gerenciamento de mídia que oferece uma plataforma completa baseada em nuvem para *uploads*, armazenamento, manipulação e entrega de imagens e vídeos, sendo projetado para otimizar o fluxo de trabalho de mídia e garantir a entrega rápida e otimizada de ativos digitais em qualquer dispositivo[1].

O GitFolio integra-se ao serviço Cloudinary para facilitar o gerenciamento e o armazenamento de imagens utilizadas nos projetos do portfólio. Essa integração foi implementada para permitir que o usuário faça *upload* de imagens de forma rápida, segura e sem a necessidade de armazenar arquivos localmente no servidor da aplicação. Ao cadastrar ou editar um projeto, o usuário pode selecionar uma imagem, que é enviada diretamente para a nuvem por meio da API do Cloudinary. O sistema utiliza as credenciais de acesso configuradas para autenticar as requisições e garantir a segurança do processo.

Após o upload, a API do Cloudinary retorna a URL pública da imagem, que é automaticamente vinculada ao projeto correspondente no portfólio do usuário. Dessa forma, as imagens ficam disponíveis para visualização imediata, sem sobrecarregar o servidor do GitFolio. Além disso, o Cloudinary oferece recursos de otimização e redimensionamento automático das imagens, garantindo melhor desempenho e qualidade visual na apresentação dos projetos.

A escolha pelo uso do Cloudinary se deu pela praticidade e escalabilidade, permitindo que o usuário gerencie imagens de maneira eficiente, sem se preocupar com limitações de armazenamento ou configurações complexas de infraestrutura. Isso torna o processo de personalização do portfólio mais simples, rápido e seguro para todos os usuários.

Para integrar o GitFólio com o Cloudinary, foi necessário configurar as credenciais de acesso fornecidas pela própria plataforma. Essas credenciais são inseridas no arquivo `.env` do projeto, garantindo segurança e evitando que dados sensíveis fiquem expostos no código-fonte. Em seguida, no arquivo `settings.py`, foi feita a configuração da biblioteca do Cloudinary utilizando as variáveis de ambiente previamente definidas. Isso permite que o Django envie diretamente os arquivos de mídia (imagens, vídeos ou documentos) para o armazenamento em nuvem, sem a necessidade de salvar localmente no servidor:

[1] Mais informações sobre as especificações do serviço estão disponíveis em: https://cloudinary.com/documentation/solution_overview

Listagem de Código 3.1 – Configuração do Cloudinary no settings.py

```
cloudinary.config(
    cloud_name=os.environ.get('CLOUDINARY_CLOUD_NAME'),
    api_key=os.environ.get('CLOUDINARY_API_KEY'),
    api_secret=os.environ.get('CLOUDINARY_API_SECRET'),
    api_proxy=os.getenv('CLOUDINARY_API_PROXY', ''),
)
```

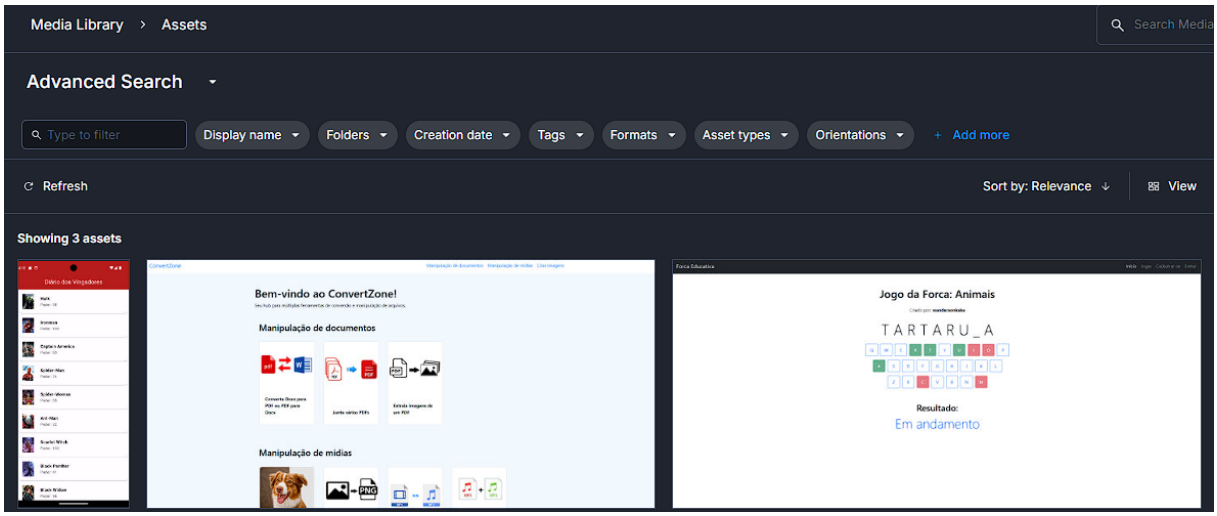
Após a configuração, foi necessário ajustar a variável `STORAGES` do Django. Nela, definimos que os arquivos de mídia seriam tratados pelo `MediaCloudinaryStorage`, que direciona automaticamente os *uploads* para o Cloudinary. Já os arquivos estáticos (CSS, JavaScript e imagens do próprio sistema) permaneceram sendo gerenciados pelo método `StaticFilesStorage` padrão do Django:

Listagem de Código 3.2 – Configuração dos storages no settings.py

```
STORAGES = {
  "default": {
      "BACKEND": "cloudinary_storage.storage.MediaCloudinaryStorage",
  },
  "staticfiles": {
      "BACKEND": "django.contrib.staticfiles.storage.StaticFilesStorage",
  },
}
```

Dessa forma, os *uploads* no aplicativo são armazenados com segurança e escalabilidade na nuvem, utilizando a performance, a CDN (*Content Delivery Network*) e a otimização de mídia do Cloudinary. A Figura 4 apresenta imagens adicionadas aos projetos, as quais são refletidas na nuvem de forma instantânea.

Figura 4 – Imagens de um portfólio refletindo na nuvem do Cloudinary



Fonte: Autoria própria (2025).

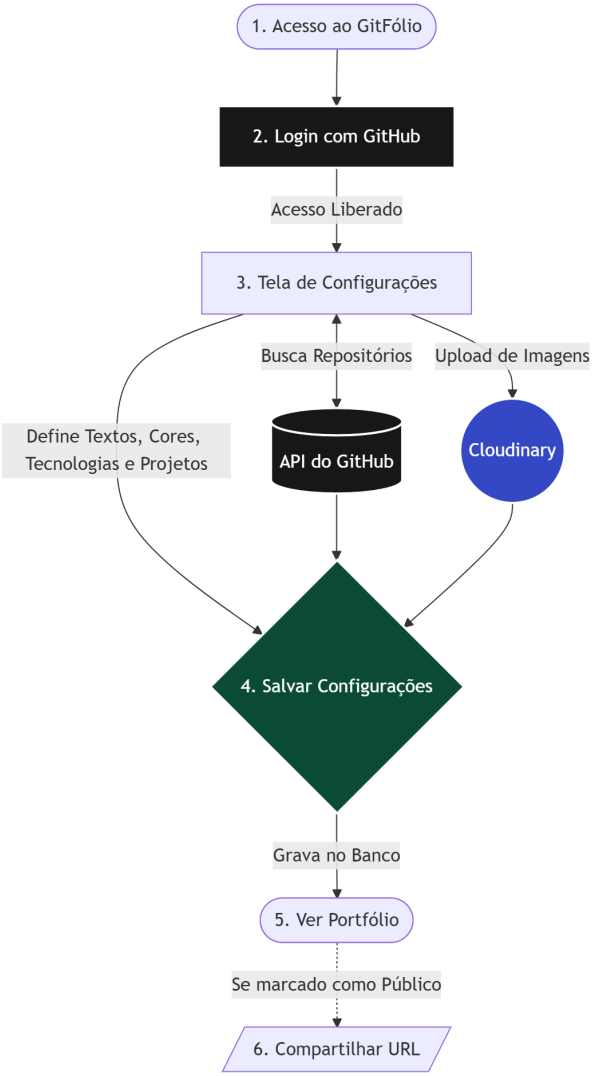
4 DESCRIÇÃO TÉCNICA DO WEBSITE

O GitFólio é uma aplicação *web* desenvolvida com o *framework* Django, cujo objetivo é permitir que desenvolvedores criem seu portfólio de forma rápida, usando os projetos do seu repositório do GitHub. O sistema oferece uma interface intuitiva para importar projetos, organizar informações profissionais e exibir habilidades técnicas de forma visualmente atraente e responsiva.

4.1 FLUXOGRAMA DE PROCESSOS E INTEGRAÇÕES DO GITFÓLIO

Para compreender a arquitetura e a jornada do usuário dentro da aplicação, a Figura 5 ilustra o diagrama do fluxo principal do GitFólio. O processo é estruturado em seis etapas que conectam o usuário no *frontend* com as integrações de *backend*.

Figura 5 – Fluxograma de processos e integrações do GitFólio



Fonte: Autoria própria (2025).

Conforme observado no diagrama, o processo é iniciado pelo acesso à plataforma e o *login* obrigatório e exclusivo via OAuth com o GitHub (Passos 1 e 2), o que resulta na liberação de acesso a página de configurações do portfólio do desenvolvedor. A partir da tela de configurações (Passo 3), o fluxo de processamento de dados se divide em três ações distintas que se complementam:

- Comunicação direta com a API do GitHub para a busca e importação automatizada dos repositórios públicos do desenvolvedor.
- *Upload* de arquivos de mídia, que são enviados, processados e hospedados externamente por meio da integração com o Cloudinary.
- Definição manual de preferências do usuário, como textos descritivos, paleta de cores e seleção de tecnologias.

A junção dessas ramificações ocorre na etapa de salvamento (Passo 4), momento em que o *framework* Django consolida os dados vindos das diferentes fontes e realiza a gravação no banco de dados. Concluída a gravação, o sistema gera a visualização final que pode ser acessada pelo dono do portfólio (Passo 5) e, caso à escolha do usuário seja de manter o perfil público, o mesmo pode compartilhar a URL do portfólio com outras pessoas (Passo 6).

4.2 FUNCIONALIDADES PRINCIPAIS

- **Autenticação via GitHub (OAuth):** o acesso ao sistema é efetuado exclusivamente por meio de autenticação com a conta GitHub do usuário. Essa integração via OAuth (*Open Authorization*) simplifica o fluxo de cadastro (eliminação de formulários de criação de credenciais) e reforça a segurança, pois as credenciais são gerenciadas pelo próprio GitHub, reduzindo o risco de vazamento de senhas e simplificando as políticas de controle de acesso.
- **Importação de projetos públicos do GitHub:** o usuário pode importar rapidamente seus repositórios públicos diretamente de sua conta GitHub. Os repositórios importados ficam disponíveis para exibição no portfólio e podem ser reordenados conforme preferência do desenvolvedor. Cada item importado é editável — é possível acrescentar descrições, imagens ilustrativas e *links* externos — o que permite adaptar a apresentação de cada projeto ao contexto do portfólio.
- **Cadastro e organização de tecnologias:** o sistema possibilita cadastrar e organizar as tecnologias dominadas pelo usuário, agrupando-as por duas categorias (*FrontEnd* e *BackEnd*), permitindo a ordenação manual. Esse recurso facilita a visualização das competências técnicas e a priorização das tecnologias mais relevantes para o público-alvo.

- **Interface intuitiva e design responsivo:** o *website* apresenta uma interface moderna e de fácil navegação, com recursos visuais que auxiliam a organização e a apresentação dos projetos e das tecnologias do usuário. As páginas foram criadas para serem responsivas, garantindo usabilidade adequada em diferentes tamanhos de tela (*smartphones*, *tablets* e *desktops*).

5 RESULTADOS E DEMONSTRAÇÃO DO SISTEMA

5.1 DESCRIÇÃO DAS INTERFACES

As interfaces do Gitfólio foram imaginadas para serem responsivas e de fácil navegação desde o início do esboço das mesmas, após a implementação com o bootstrap, foi possível verificar que o *design* desenvolvido inicialmente foi seguido à risca, houveram, durante o desenvolvimento, algumas mudanças na forma que seriam apresentadas determinadas funcionalidades do site, mas nada que não seguisse o padrão de *design* e cores definidas no esboço inicial. Nesta seção mostrarei as telas na ordem que o usuário encontraria caso estivesse se conectando no GitFólio pela primeira vez, também mostrarei em alguns casos, tanto a versão *Desktop* quanto a versão *mobile* das mesmas.

A Figura 6 representa a tela de apresentação do GitFólio em diferentes dispositivos, a partir dela podemos ver a responsividade do Bootstrap em ação. Nessa página tem uma seção de apresentação sobre o site, e botões como o “começar agora” e “crie sua conta grátis” que abrem o modal da Figura 3 para realizar *login* no site com a conta do GitHub

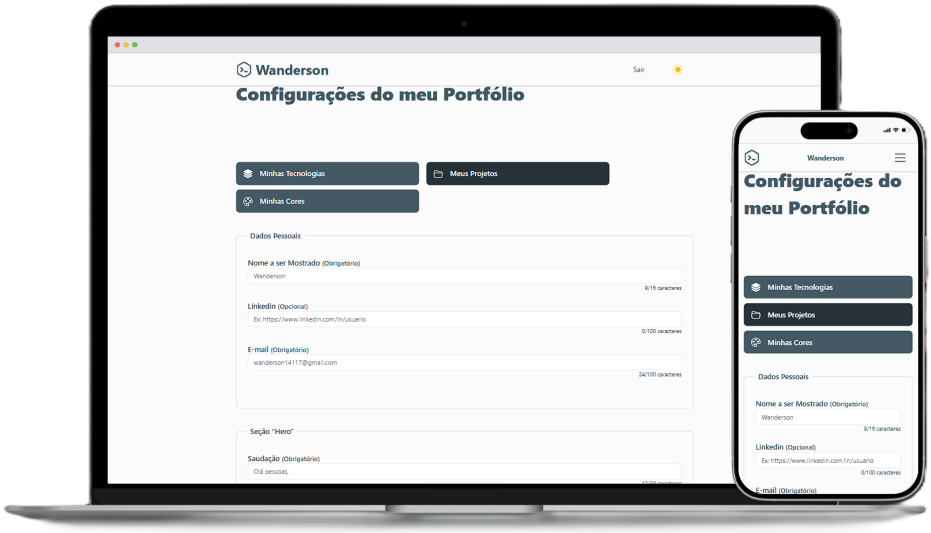
Figura 6 – Tela inicial do site responsiva

Fonte: Autoria própria (2025).

Após o *login* com sucesso do usuário, a tela que o mesmo é redirecionado é a de configurações do portfólio. A tela de configurações se divide em seções separadas, abaixo mostrarei e detalharei cada uma das seções.

Na Figura 7 podemos ver o *header* da página que mostra o nome do usuário, a opção de usar modo escuro e o botão de sair da conta em diferentes dispositivos. No centro da imagem existem três botões de ação, cada um deles abre um Modal diferente na página sem precisar redirecionar a mesma, eles serão mostrados mais a frente.

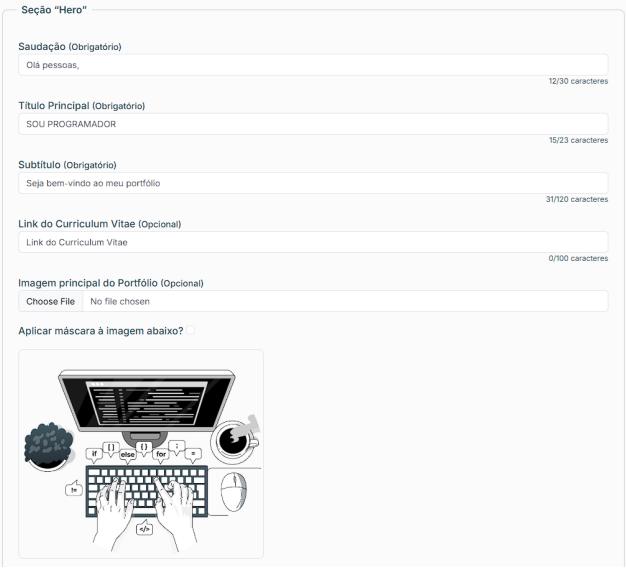
Figura 7 – Parte inicial da página de configurações do portfólio



Fonte: Autoria própria (2025).

A Figura 8 apresenta a configuração da seção *Hero*, que seria a primeira seção a ser vista por quem estiver visualizando o portfólio pronto, ela basicamente seria a apresentação do desenvolvedor dono do portfólio.

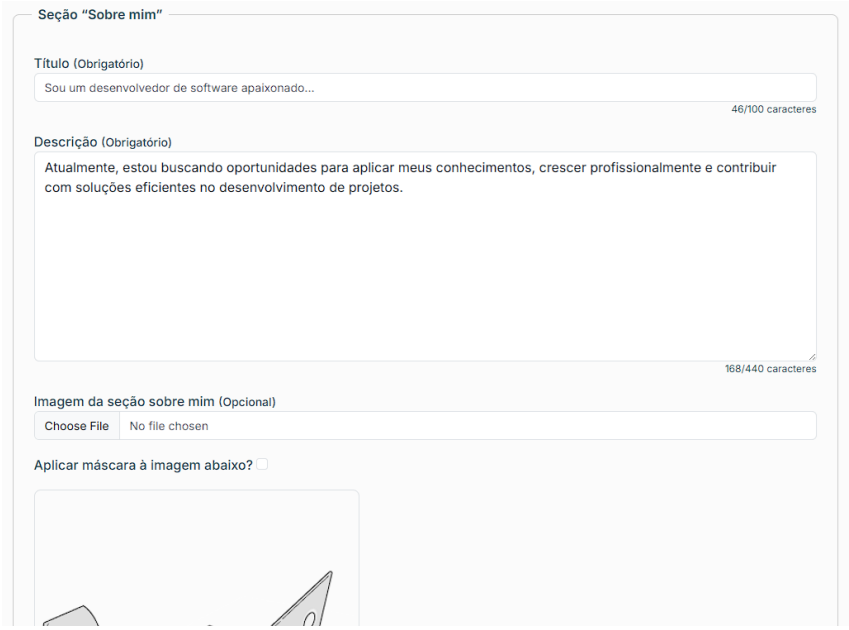
Figura 8 – Configurações da seção *Hero* do portfólio



Fonte: Autoria própria (2025).

A Figura 9 representa a configuração da seção "sobre mim", onde o desenvolvedor se apresenta para aqueles que estão visualizando o portfólio dele, ela basicamente seria a área onde o desenvolvedor diz quais são os seus objetivos atualmente na sua carreira.

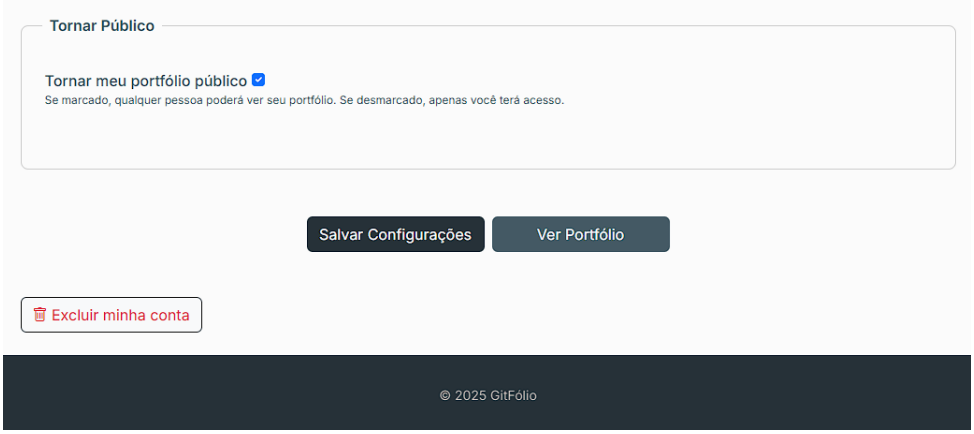
Figura 9 – Configurações da seção "sobre mim"do portfólio



Fonte: Autoria própria (2025).

A Figura 10 exibe a parte final do formulário de configurações. Nela, constam a opção de tornar o portfólio público com um *checkbox*, os botões para salvar as alterações e visualizar o portfólio, e a opção para exclusão da conta de usuário.

Figura 10 – Parte final da página de configurações

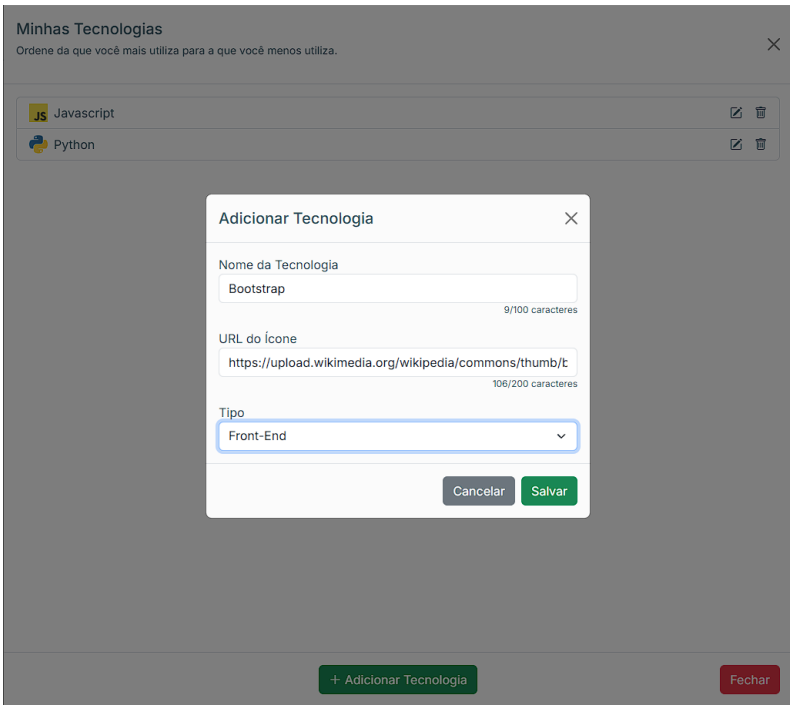


Fonte: Autoria própria (2025).

A Figura 11 mostra o modal que aparece ao usuário clicar no botão “Minhas tecnologias” no ínicio da página de configurações, ele é basicamente onde o usuário

consegue cadastrar tecnologias, adicionar uma URL para o ícone da mesma e o tipo dela, se é *Frontend* ou *Backend*. Quando o desenvolvedor possui tecnologias cadastradas no portfólio, é possível ordenar as mesmas por preferência.

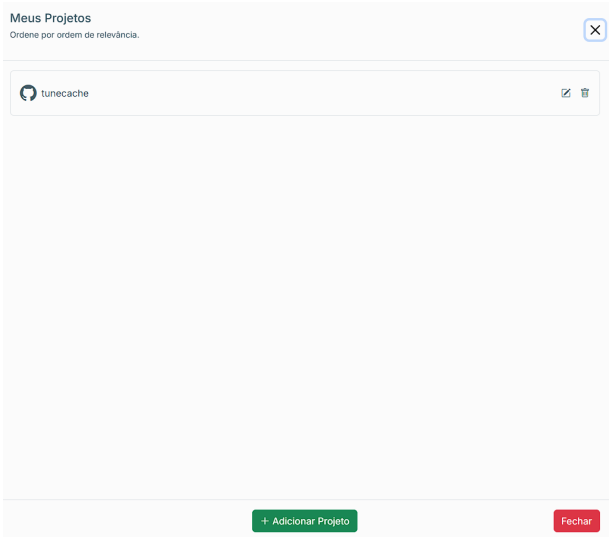
Figura 11 – Modal de cadastro de tecnologias



Fonte: Autoria própria (2025).

A seguir estão os modais da página de configurações do GitFólio, será mostrado apenas a versão desktop, mas a versão *mobile* altera apenas o tamanho dos mesmos de forma responsiva. A Figura 12 apresenta o modal que aparece ao usuário clicar no botão “Meus Projetos” no início da página, ele é basicamente onde o usuário consegue listar seus projetos já adicionados ao portfólio e também pode adicionar mais apertando no botão “Adicionar Projeto”.

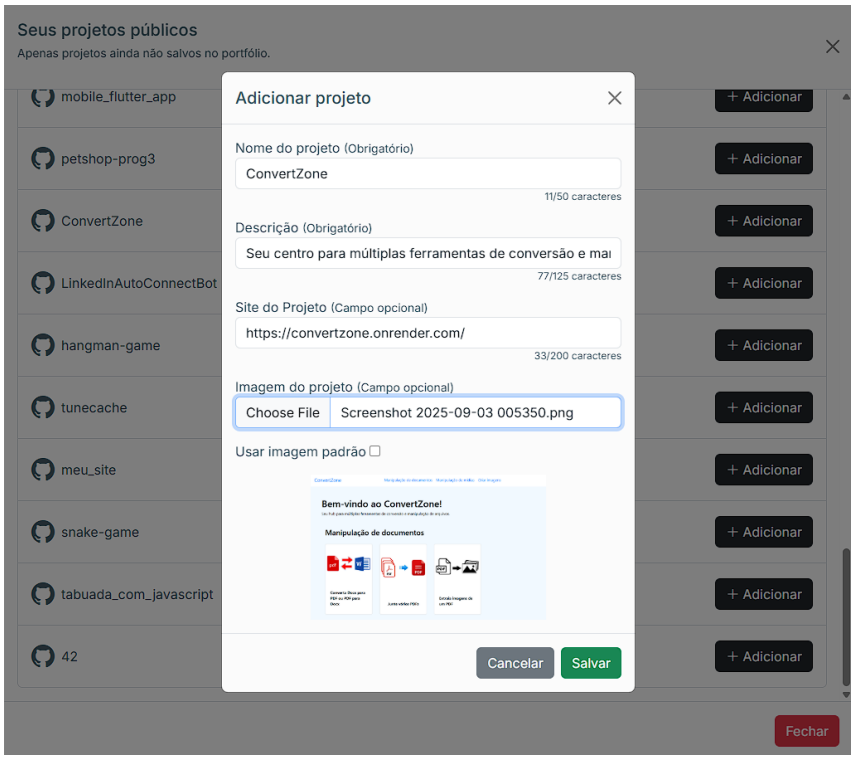
Figura 12 – Modal de cadastro de projetos



Fonte: Autoria própria (2025).

Na Figura 13 mostra o modal que aparece ao usuário clicar no botão “Adicionar projeto” da Figura 12 e após no botão adicionar no projeto desejado, nesta seção os projetos são puxados do GitHub público do usuário por meio da API (*Application Programming Interface*) do GitHub, eles são carregados em forma de paginação sob demanda conforme o usuário movimenta a barra lateral para baixo com o objetivo de evitar o *rate limit* da API.

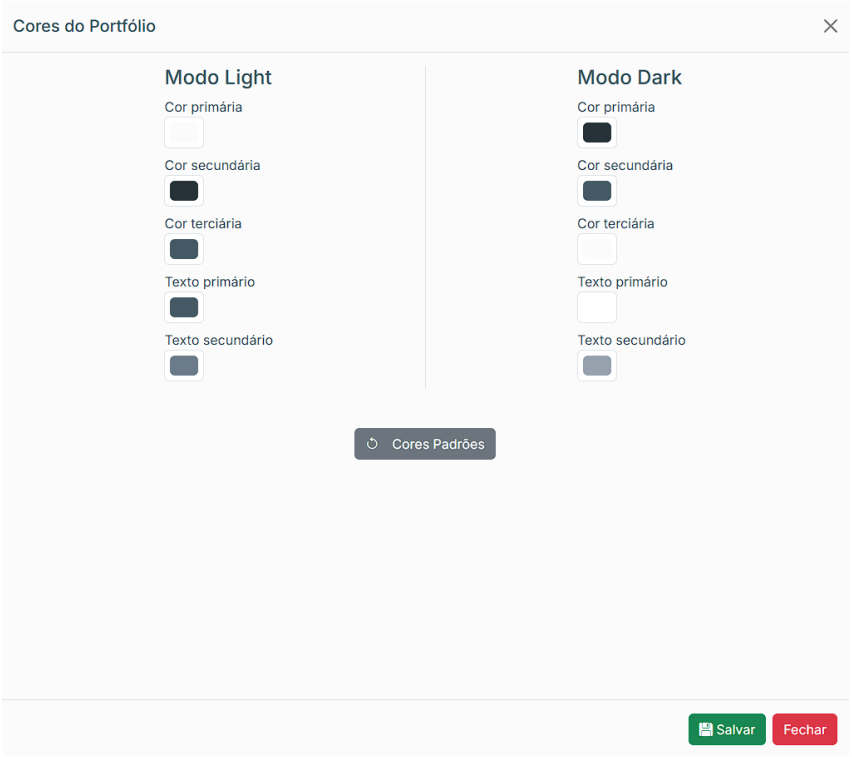
Figura 13 – Modal de seleção de projetos do GitHub



Fonte: Autoria própria (2025).

A Figura 14 representa o que aparece na tela ao usuário clicar no botão “Minhas Cores”, nela é possível o usuário definir as cores desejadas no seu portfólio para poder deixar da forma que mais lhe agrada, no centro da página tem um botão escrito “cores padrões”, onde ao apertar nele e em salvar, basicamente o portfólio do desenvolvedor vai voltar para as cores padrões do GitFólio.

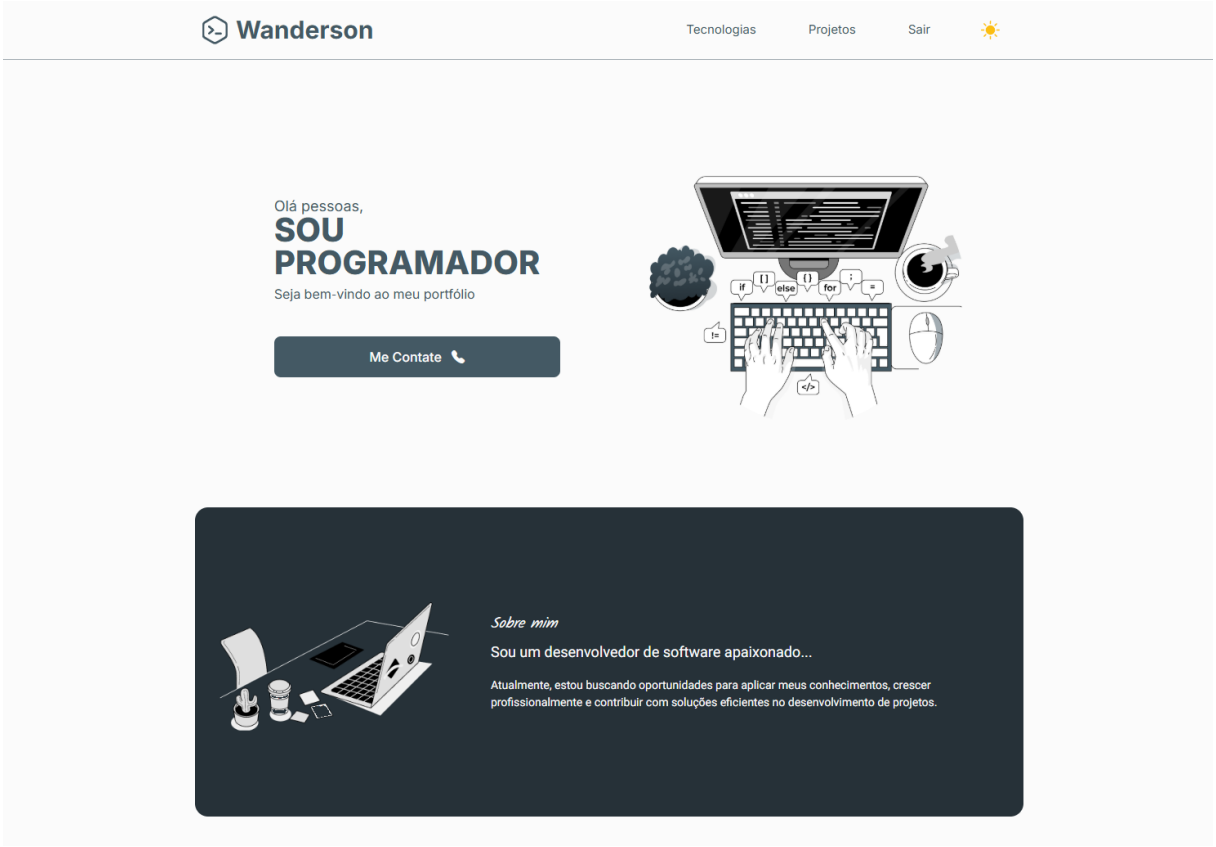
Figura 14 – Modal de seleção de cores do portfólio



Fonte: Autoria própria (2025).

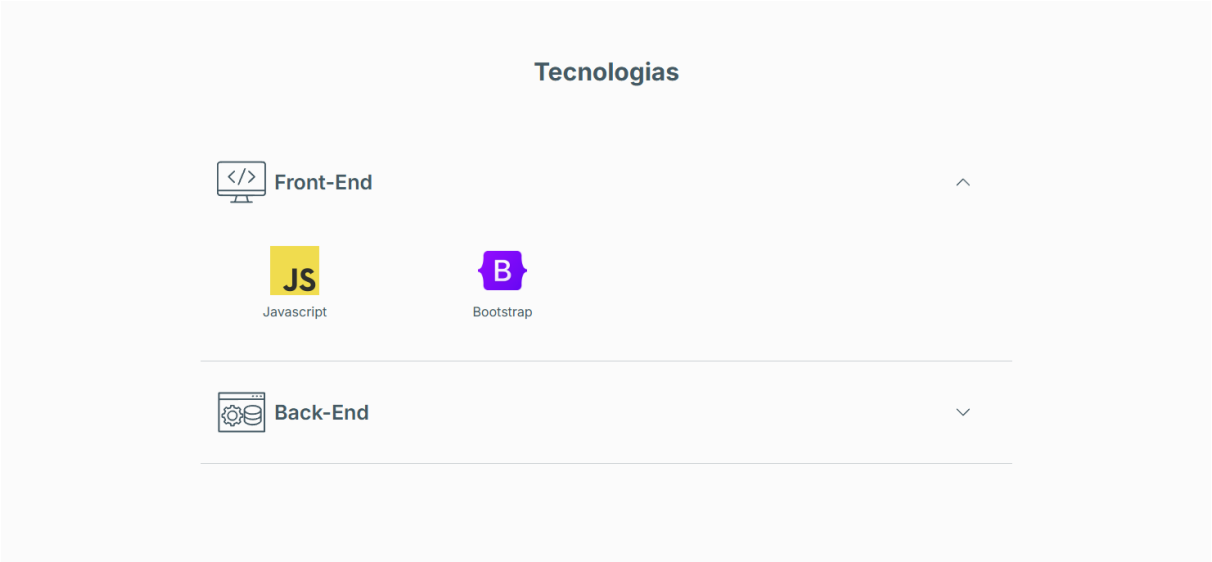
Após terminar de configurar e salvar o portfólio, ao usuário apertar no botão “Ver Portfólio” no fim da página de configurações, é possível ver como ficou o seu portfólio, e caso a opção de deixar público esteja marcado, já é possível compartilhar o *link* do portfólio com outras pessoas, para poder mostrar o seu trabalho. As figuras 15, 16 e 17 representam o portfólio final de um usuário na versão Desktop.

Figura 15 – Seção ’Hero’ e seção ’Sobre Mim’ do portfólio



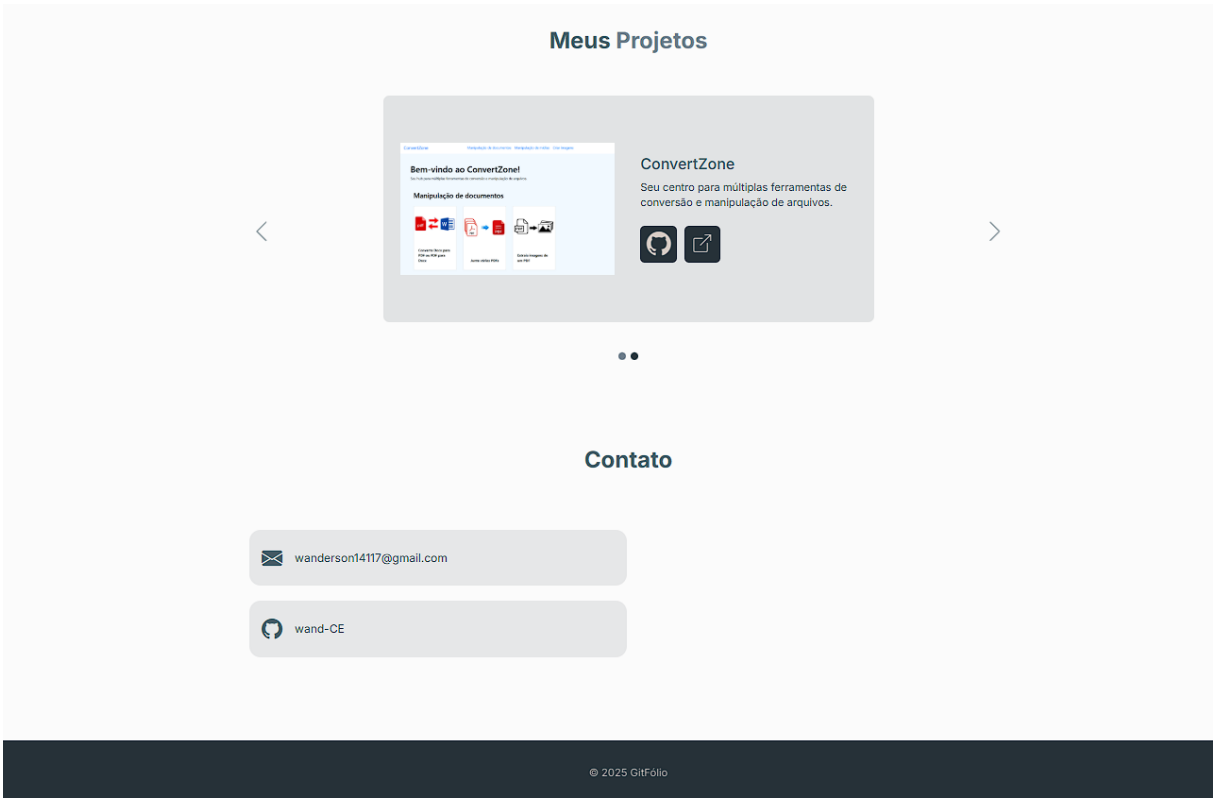
Fonte: Autoria própria (2025).

Figura 16 – Seção de tecnologias do Portfólio



Fonte: Autoria própria (2025).

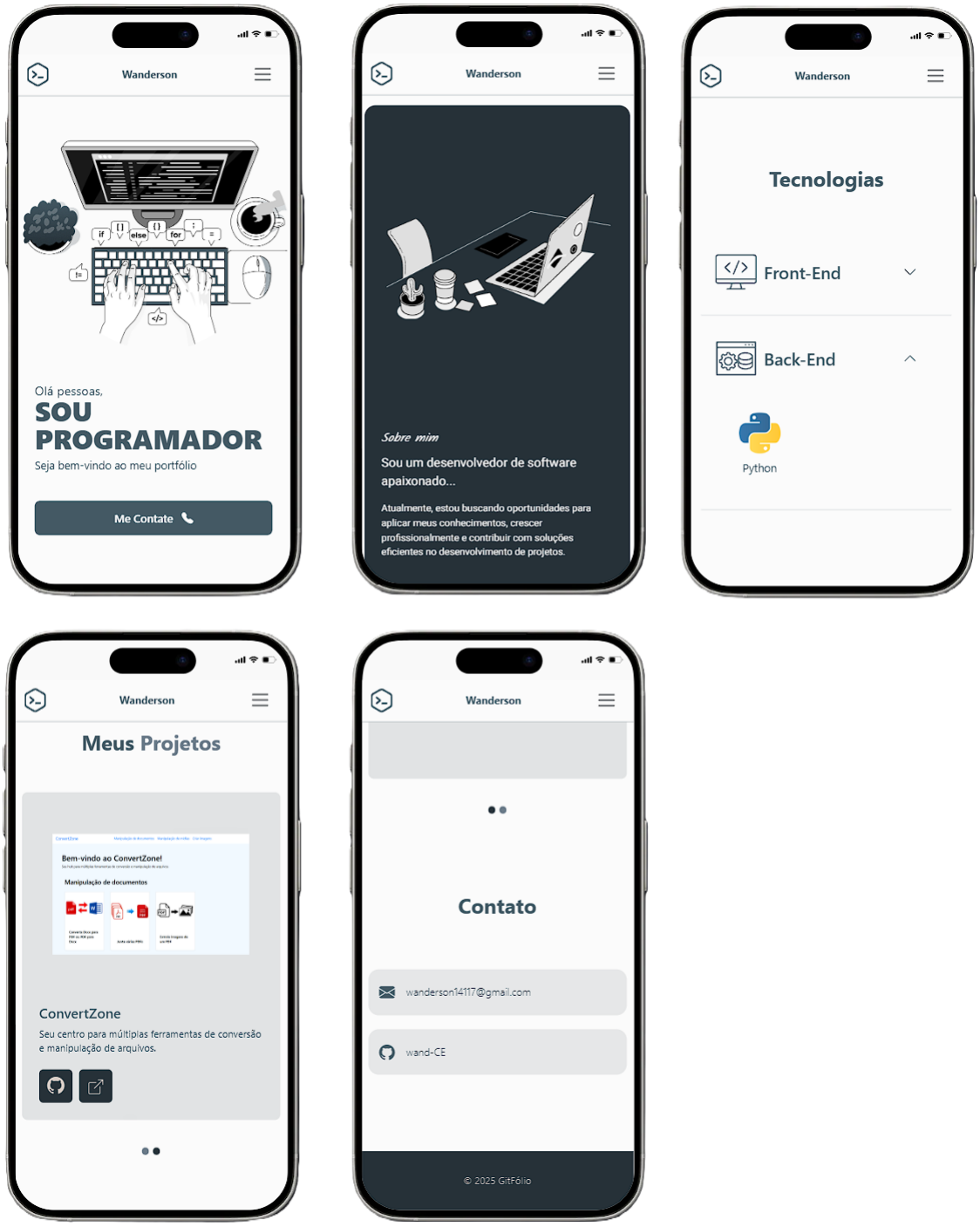
Figura 17 – Seção de Projetos e de contato do portfólio



Fonte: Autoria própria (2025).

Como a ideia do site era ser um website responsivo, abaixo estão as telas do portfólio em sua versão mobile, onde podemos verificar que a adaptação de um tamanho de tela maior para de tamanho menor foi feito de modo a respeitar conceitos de UX (*User Experience*) e UI (*User Interface*), sem perder funcionalidades ou prejudicar a experiência do usuário. As telas *mobile* são representadas pela Figura 18.

Figura 18 – Portfólio em versão mobile



Fonte: Autoria própria (2025).

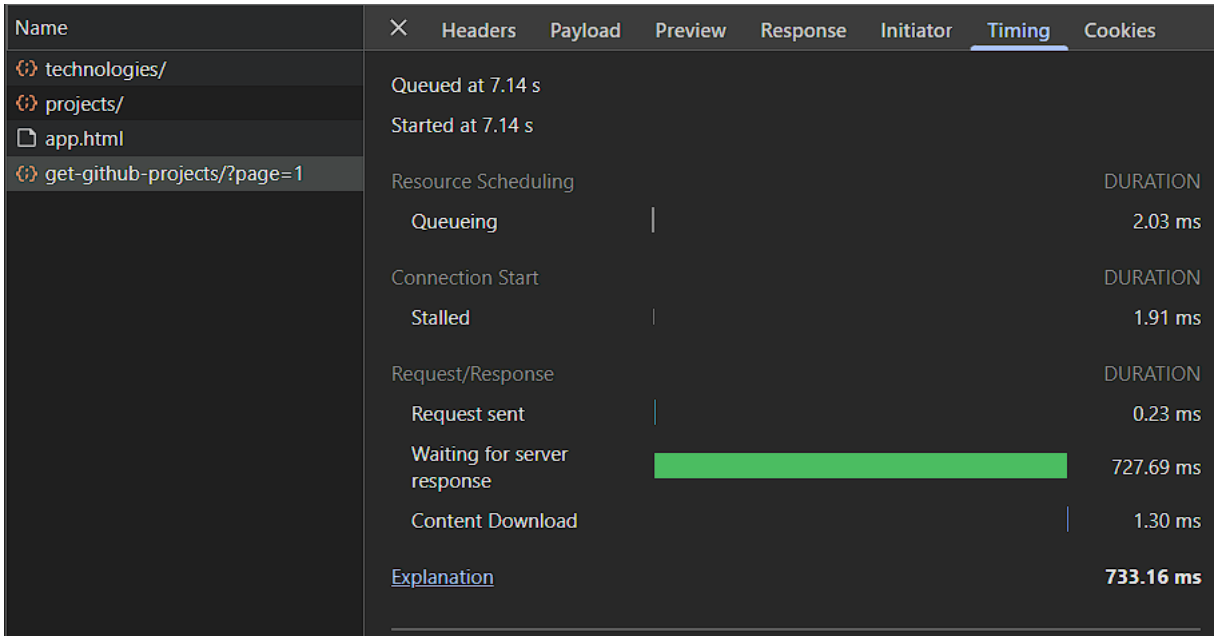
5.2 AVALIAÇÃO EXPERIMENTAL E DESEMPENHO

Para validar a confiabilidade e a eficiência do GitFólio, foram realizados testes focados no tempo de resposta e no desempenho dos *endpoints* da aplicação em produção. Conforme estabelecido na metodologia, a análise de métricas de performance é fundamental para assegurar a viabilidade técnica do sistema.

A métrica de tempo de resposta é crucial, visto que a aplicação depende da comunicação externa com a API do GitHub e do Cloudinary. Para esta avaliação, utilizou-se a ferramenta de desenvolvedor do navegador (*DevTools*), rastreando o *endpoint* responsável pela listagem de repositórios do desenvolvedor, simulando a jornada real de um usuário ao importar seus projetos.

A Figura 19 apresenta o detalhamento das etapas de latência e processamento da requisição (*Timing*).

Figura 19 – Métricas de tempo de resposta do endpoint de repositórios



Fonte: Autoria própria (2026).

Conforme observado nos resultados experimentais, a requisição foi processada com sucesso. O tempo total registrado para a operação foi de 733,16 ms para buscar 20 repositórios instantaneamente. Analisando o detalhamento técnico, observa-se que o tempo de espera pela resposta do servidor foi de 727,69 ms. Esse valor é tecnicamente significativo, pois engloba o tempo de ida da requisição ao servidor hospedado, o processamento da lógica que consome a API externa do GitHub e o início da devolução dos dados.

Este resultado demonstra um desempenho satisfatório para uma aplicação que depende de integrações de terceiros, mantendo o tempo de resposta abaixo de 1 segundo, o que é um indicador positivo de usabilidade. A fluidez observada no carrega-

mento dos dados (apenas 1,30 ms) confirma que a estrutura do JSON entregue pela aplicação é leve e otimizada. Paralelamente, testes manuais confirmaram que a interface responsiva e a estratégia de paginação sob demanda mantiveram a navegação estável e livre de latências perceptíveis em dispositivos móveis e *desktops*.

6 CONCLUSÃO

O presente Trabalho de Conclusão de Curso alcançou seu objetivo geral de desenvolver o GitFólio, uma aplicação *web* funcional para a criação de portfólios personalizados a partir da integração com a API (*Application Programming Interface*) do GitHub. O sistema provou ser uma solução prática e eficiente para o problema da baixa visibilidade e da dificuldade de organização de projetos por parte dos desenvolvedores, especialmente os iniciantes.

Os objetivos específicos foram integralmente cumpridos: a autenticação segura via OAuth (*Open Authorization*) com o GitHub foi implementada com sucesso, facilitando o acesso do usuário; a importação, seleção e organização dos repositórios públicos foram plenamente funcionais, permitindo a gestão eficaz dos projetos a serem exibidos. Além disso, foram entregues *templates* responsivos, opções de personalização visual (cores, foto e descrições) e uma arquitetura de dados eficiente, baseada no Django/Python e MySQL, que garantiu a escalabilidade e a manutenção do sistema. A adoção de um *design* intuitivo e responsivo, conforme demonstrado nas Figuras 17 a 19, assegura que os portfólios gerados sejam acessíveis e visualmente atraentes em qualquer dispositivo.

Em termos de aplicabilidade, o GitFólio oferece uma contribuição significativa ao mercado de TI, preenchendo uma lacuna ao fornecer uma ferramenta que transforma o repositório técnico do GitHub em um material de *marketing* pessoal e profissional coeso. A automação do processo de criação do portfólio permite que o desenvolvedor gaste menos tempo na apresentação e mais tempo no desenvolvimento de novos projetos.

Como trabalho futuro, sugerem-se algumas melhorias e expansões, como a integração com outras plataformas de controle de versão (ex: GitLab e Bitbucket), a implementação de um sistema de análise de métricas (ex: número de visualizações do portfólio), e o desenvolvimento de mais opções de *templates* de *design* para aumentar a diversidade visual dos portfólios.

Conclui-se que o GitFólio não é apenas um sistema que atende aos requisitos técnicos e funcionais propostos, mas uma ferramenta de valor real, capaz de potencializar a visibilidade e as oportunidades de carreira para programadores no cenário digital.

REFERÊNCIAS

CASCIARO, Mario; MAMMINO, Luciano. **Node.js Design Patterns**. 3. ed. [S. l.]: Packt Publishing, 2020. ISBN 978-1839214110.

CHACON, Scott; STRAUB, Ben. **Pro Git**. 2. ed. Nova Iorque: Apress, 2014. ISBN 978-1-4842-0077-3.

GOUSIOS, Georgios; PINZGER, Martin; DEURSEN, Arie van. **An exploratory study of the pull-based software development model**. In*:* PROCEEDINGS of the 36th International Conference on Software Engineering. [S. l.]: ACM, 2014. p. 345–355.

KRUG, Steve. **Não me faça pensar**: uma abordagem de bom senso à usabilidade na web e mobile. 3. ed. São Paulo: Alta Books, 2014. ISBN 978-0-321-96551-6.

MARLOW, Jennifer; DABBISH, Laura; HERBSLEB, Jim. **Impression formation in online peer production**: activity traces and personal profiles in GitHub. In*:* PROCEEDINGS of the 2013 conference on Computer supported cooperative work. [S. l.]: ACM, 2013. p. 117–128.

NORMAN, Donald A. **O design do dia a dia**. Rio de Janeiro: Rocco, 2013. ISBN 978-85-69474-42-5.

ROSSUM, Guido Van. **Computer Programming for Everybody**.: [S. l.: s. n.], 1999. Acessado em: 24-Out-2025. Disponível em: https://www.python.org/doc/essays/cp4e/.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Education do Brasil, 2011. ISBN 978-85-7936-108-1.

SONMEZ, John. **Soft Skills**: The software developer's life manual. 2. ed. Nova Iorque: Manning Publications, 2020. ISBN 9781617292392.

VINCENT, William S. **Django for Beginners**: Build websites with Python and Django. [S. l.]: Welsh & Vincent Publishing, 2020. ISBN 978-1735467207.