Documentação de Software

Hackers School

Autores:

Vitor Felizatti

Pedro Otavio

Thiago Barros Gomes

Documento de especificação de requisitos do projeto "H.S."

Versão: 1.3

Projeto Interdisciplinar do 4º semestre do curso de Desenvolvimento de Software Em Multiplataforma desenvolvido na Faculdade de Tecnologia de Araras (FATEC), apresentado como Trabalho de Conclusão.

Faculdade de Tecnologia de Araras

DESENVOLVIMENTO DE SOFTWARE EM MULTIPLATAFORMA

TRABALHO DE CONCLUSÃO DO 4ºSEMESTRE.

Orientadores:

_____

Prof. Fernando Bryan Frizzarin

Laboratório de Desenvolvimento Web

_____

Prof. Orlando Saraiva

Desenvolvimento Web II

SUMÁRIO

1.0 INTRODUÇÃO

O projeto Hackers School é a Rede Social do Hackers do Bem, uma plataforma exclusiva para hackers, entusiastas de segurança cibernética e programadores compartilharem conhecimento, colaborarem e participarem de desafios.

1.1 OBJETIVO

O principal objetivo do projeto Hackers School é criar uma plataforma inovadora que funcione como uma rede social dedicada a hackers, entusiastas de segurança cibernética ou grupos ligados as áreas em questão.

1.2 ESCOPO

Executável:

Perfil de membro, cada membro tem um perfil personalizado, adicione suas habilidades, interesses e experiência e inclua links para blogs pessoais e redes sociais, para participa de discussões em fóruns temáticos e criar grupos para explorar tópicos específicos, temos a biblioteca de recursos para acessar artigos, tutorias e vídeos relevantes e também pode contribuir com seus próprios materiais. Pode testa suas habilidades em desafios regulares e explore categorias como criptografia e exploração de software.

Não executável:

Benefícios: Ajuda a conectar pessoas que tem a mesma área de interesse e estão interessadas em apreender e compartilhar suas experiencias.

Objetivos: Ensina pessoas a reconhecer tipos diferentes de golpes apreender mais sobre área de desenvolvimentos.

Meta: é promover a educação, colaboração, compartilhamento de conhecimento e desenvolvimento de habilidades entre os membros da comunidade de segurança cibernética e programação, com o objetivo final de contribuir para um ambiente online mais seguro e informado.

1.3 ACRÔNIMOS

RSHB: Rede Social Hacker do Bem

RF: Requisitos Funcionais

RNF: Requisitos Não Funcionais

UML: Linguagem de Modelagem Unificada

R: Restrições

SP: Suposições e Pendências

FER: Ferramentas

AQ: Atributos de Qualidade

RI: Requisitos de Interface

ML: Modelo Lógico

1.5 VISÃO GERAL

Este projeto uma é a ideia de uma plataforma colaborativa projetada para unir hackers, entusiastas de segurança cibernética e programadores. Nosso objetivo é fornecer um espaço seguro e estimulante onde os membros possam criar perfis personalizados, participar de fóruns e grupos temáticos, acessar e contribuir com uma biblioteca de recursos, enfrentar desafios e para aprimorar suas habilidades, se mantendo atualizado com eventos e webinars relevantes para a comunidade.

A plataforma será desenvolvida utilizando Django ou Flask e um banco de dados MySQL, com um forte enfoque em medidas robustas de segurança para proteger os dados dos usuários.

2.0 DESCRIÇÃO GERAL

O projeto "Rede Social do Hackers do bem" é uma iniciativa para criar uma plataforma interativa e educativa voltada para a comunidade de entusiastas da segurança cibernética, hackers éticos e programadores. O objetivo é proporcionar um ambiente onde os usuários possam se conectar, compartilhar conhecimentos e crescer profissionalmente. A rede social contará com funcionalidades como perfis personalizáveis, fóruns de discussão, grupos temáticos, uma biblioteca de recursos educacionais, desafios de segurança cibernética e eventos virtuais.

As principais características do projeto incluem:

- **Perfis de Membros**: Os usuários poderão criar perfis detalhados, destacando suas habilidades, interesses e experiências, além de vincular seus blogs pessoais e outras redes sociais.
- **Fóruns e Grupos**: Espaços para discussões em grupo e fóruns temáticos permitirão aos membros explorar e debater tópicos específicos de interesse.
- **Biblioteca de Recursos**: A plataforma oferecerá acesso a uma variedade de materiais educativos, como artigos, tutoriais e vídeos, e incentivará os membros a contribuir com seus próprios conteúdo.

2.1 REQUISITOS FUNCIONAIS

Os requisitos funcionais definem as funcionalidades principais que o sistema Hackers School deve oferecer para proporcionar uma experiência completa e satisfatória aos usuários:

RF01: Cadastrar usuário - obrigatório

RF02: Compartilhar informações e conhecimentos que tem na área - desejáveis

RF03: Receber atividades para ambrear os conhecimentos – obrigatório;

RF04: Acrescentar informações - obrigatório;

RF05: Receber notificações de respostas – desejável;

2.2 REQUISITOS NÃO FUNCIONAIS

RNF01: tempo de resposta não deve exceder 5 segundos;

RNF02: Deletar Prontuário;

RNF03: O sistema deverá manter o usuário conectado a sua conta;

RNF04: Editar informações que outros usuários acrescentaram;

2.2 REQUISITOS DE INTERFACE

RI01: Layout Intuitivo e fácil de navegar. Com elementos claros e organizados, permitindo que os usuários encontrem rapidamente as informações necessárias.

RI02: Design Responsivo: Acessível em diferentes dispositivos, como desktops, tablets e smartphones.

RI03: Segurança e Privacidade: A segurança dos dados de saúde é fundamental. A interface deve implementar medidas robustas de segurança, como criptografia e autenticação.

RI04: Registro e Pesquisa Eficiente: A interface deve permitir o registro eficiente de informações de sobre o assunto requerido, fácil entra em atividades.

RI05: Visualização de Dados: Os dados de desempenhos e chats papos devem ser apresentados de forma clara e compreensível.

RI06: Comunicação Integrada: A interface deve permitir a comunicação entre os usuários de forma segura e eficiente. Recursos como mensagens seguras, solicitação de atividades para melhorar eu desempenho e rendimentos e o engajamento.

2.3 ATRIBUTOS DE QUALIDADE

2.3.1 Desempenho

O sistema deve garantir um desempenho eficiente e ágil em todas as suas operações.

RNF01: O tempo de resposta das principais funcionalidades do sistema não deve exceder 2 segundos.

RNF02: O sistema é escalável e suportar até 10.000 usuários simultâneos sem degradação perceptível no desempenho.

2.3.2 Segurança

O sistema protege os dados dos usuários e garantir que apenas usuários autorizados possam acessar informações sensíveis.

RNF03: Feito a implementação de autenticação senhas hash no banco de dados para acesso ao sistema.

RNF04: Dados pessoais dos usuários devem ser criptografados em trânsito e em repouso.

2.3.3 Usabilidade

O sistema proporciona uma experiência de usuário agradável e ser acessível a todos os tipos de usuários.

RNF05: O sistema é compatível com as diretrizes de acessibilidade WCAG 2.1.

RNF06: A interface é intuitiva, com navegação clara e consistente.

2.3.4 Confiabilidade

O sistema deve ser confiável, garantindo alta disponibilidade e recuperação rápida em caso de falhas.

RNF07: O sistema deve garantir uma disponibilidade de 99,9% ao longo do ano.

RNF08: Implementação de backups diários e mecanismos de recuperação automática de falhas.

2.3.5 Manutenção

O sistema deve ser de fácil manutenção e permitir atualizações sem grandes interrupções.

RNF09: O código deve ser modular e bem documentado para facilitar a manutenção.

RNF10: As atualizações do sistema devem ser possíveis sem interromper o serviço por mais de 5 minutos.

2.4 Características dos Usuários

2.4.1 Administradores

CU01 - Descrição: Usuários responsáveis pela gestão e manutenção do sistema, incluindo a criação e gerenciamento de contas de outros usuários, monitoramento do sistema, gerenciamento das competições e manutenção dos rankings.

CU02 – Necessidades:

- Acesso a todas as funcionalidades do sistema.
- Ferramentas para monitoramento e relatórios.
- Controle sobre permissões de usuários e gerenciamento de competições.

CU03 - Como o sistema atende

- Interface de administração dedicada.
- Painéis de controle e relatórios de atividades.
- Ferramentas de gerenciamento de usuários, competições e permissões.

2.4.2 Usuários do Site

CU04 - Descrição: Usuários que utilizam o sistema para participar de competições, receber feedback sobre suas respostas, e acompanhar seu progresso nos rankings.

CU05 - Necessidades:

- Participar de competições e submeter respostas.
- Receber feedback e resultados das competições por email.
- Visualizar e acompanhar seu progresso nos rankings.

CU06: Como o sistema atende

Como o sistema atende:

- Interface amigável e intuitiva para participar de competições.
- Funcionalidade para submissão de respostas e recebimento de feedback por email.
- Painéis de controle para visualizar rankings e progresso pessoal.

2.5 Restrições

R01: Restrições de Plataforma

- Descrição: O sistema deve ser compatível com os principais navegadores modernos (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).
- Detalhes: Deve garantir funcionalidade completa e aparência consistente nos navegadores mencionados, sem a necessidade de plugins adicionais.

R02: Restrições de Tecnologia

- Descrição: O sistema deve ser desenvolvido utilizando as tecnologias e frameworks específicos definidos pelo projeto.
- Detalhes: Deve utilizar, por exemplo, React.js para o frontend e Node.js para o backend, conforme especificado no plano do projeto.

R03: Restrições de Desempenho

- Descrição: O sistema deve atender aos requisitos de desempenho definidos, incluindo tempos de resposta e suporte a um número máximo de usuários simultâneos.
- Detalhes: O tempo de resposta para as principais funcionalidades não deve exceder 2 segundos, e o sistema deve suportar até 10.000 usuários simultâneos sem degradação perceptível no desempenho.

R04: Restrições de Segurança

- Descrição: O sistema deve implementar medidas de segurança rigorosas para proteger os dados dos usuários.
- Detalhes: Deve incluir criptografia de dados em trânsito e em repouso, autenticação segura, e conformidade com as normas de segurança relevantes (como GDPR ou LGPD).

R05: Restrições Orçamentárias

- Descrição: O desenvolvimento e manutenção do sistema devem estar dentro do orçamento alocado.
- Detalhes: O orçamento deve cobrir todos os custos de desenvolvimento, hospedagem, manutenção e suporte, sem exceder o valor definido pelo projeto.

R06: Restrições de Tempo

- Descrição: O sistema deve ser desenvolvido e implementado dentro do cronograma estabelecido.
- Detalhes: O projeto deve ser concluído dentro do prazo de 6 meses, com marcos importantes entregues em datas específicas conforme o plano do projeto.

R07: Restrições de Conformidade

- Descrição: O sistema deve estar em conformidade com todas as regulamentações e padrões aplicáveis.
- Detalhes: Deve atender às normas de acessibilidade (WCAG 2.1), proteção de dados (GDPR, LGPD), e quaisquer outros requisitos legais e regulamentares aplicáveis ao contexto do projeto.

2.6 Suposições e Pendências

SP01: Suposições

- SP01.1: Disponibilidade de Recursos

Descrição: Supõe-se que todos os recursos necessários para o desenvolvimento do projeto (desenvolvedores, ferramentas, infraestrutura, etc.) estarão disponíveis conforme planejado.

- SP01.2: Participação dos Usuários

Descrição: Supõe-se que os usuários finais (administradores e usuários do site) estarão disponíveis para participar das fases de testes e fornecer feedback contínuo.

- SP01.3: Conectividade

Descrição: Supõe-se que os usuários terão acesso a uma conexão de internet estável e rápida para usar o sistema sem problemas de desempenho.

- SP01.4: Conformidade com Requisitos Legais

Descrição: Supõe-se que todas as regulamentações legais e padrões aplicáveis serão cumpridos durante o desenvolvimento e operação do sistema.

SP02: Pendências

- SP02.1: Definição Completa dos Requisitos

Descrição: A definição completa e detalhada dos requisitos funcionais e não funcionais está pendente, aguardando a validação final dos stakeholders.

- SP02.2: Testes de Integração

Descrição: A execução completa dos testes de integração entre os diferentes módulos do sistema está pendente e será realizada após a conclusão do desenvolvimento inicial.

- SP02.3: Feedback dos Usuários

Descrição: A coleta de feedback dos usuários durante a fase de testes beta está pendente e será essencial para ajustar e aprimorar o sistema antes do lançamento final.

- SP02.4: Implementação de Funcionalidades Adicionais

Descrição: A implementação de algumas funcionalidades adicionais, identificadas como desejáveis mas não críticas, está pendente e será considerada se houver tempo e recursos disponíveis.

3.1 Métodos Utilizados

3.1.1 Metodologia Ágil

- Descrição: A metodologia ágil foi adotada para o desenvolvimento do projeto, permitindo flexibilidade e ajustes contínuos ao longo do processo.
- Detalhes:
    - Scrum: Utilizado para organizar o trabalho em sprints, permitindo entregas incrementais e feedback regular dos stakeholders.
    - Kanban: Utilizado para visualização do fluxo de trabalho e gestão das tarefas em andamento.

3.1.2 Desenvolvimento Orientado a Testes (TDD)

- Descrição: A prática de TDD foi implementada para garantir a qualidade do código e facilitar a detecção de bugs desde as fases iniciais do desenvolvimento.
- Detalhes:
    - Especificação de testes antes da implementação de funcionalidades.
    - Utilização de frameworks de teste como Jest para JavaScript/Node.js.

3.1.3 Design Centrado no Usuário (UCD)

- Descrição: O design centrado no usuário foi aplicado para garantir que o sistema atenda às necessidades e expectativas dos usuários finais.
- Detalhes:
    - Condução de pesquisas e entrevistas com usuários para coletar requisitos.
    - Criação de wireframes e protótipos para validação de design com usuários.

3.1.4 Integração Contínua (CI) e Entrega Contínua (CD)

- Descrição: Práticas de CI/CD foram adotadas para automatizar o processo de integração e entrega de código, garantindo um ciclo de desenvolvimento rápido e eficiente.
- Detalhes:
    - Utilização de ferramentas como Jenkins e GitHub Actions para automação de builds e deploys.
    - Testes automatizados executados em cada commit para garantir a estabilidade do sistema.

3.1.5 Controle de Versão

- Descrição: O controle de versão foi implementado utilizando Git para gerenciar mudanças no código-fonte de maneira eficiente.
- Detalhes:
    - Utilização de GitHub para hospedagem de repositórios e colaboração entre desenvolvedores.
    - Adoção de estratégias de branching como Git Flow para organização do trabalho.

3.1.6 Documentação Contínua

- Descrição: A documentação contínua foi mantida ao longo do projeto para assegurar que todas as partes interessadas tenham acesso às informações atualizadas.
- Detalhes:
    - Documentação do código com comentários e markdown.
    - Criação de documentação do sistema, incluindo manuais de usuário e guias de instalação.

4.0 Ferramentas

FER01: Flask

- Descrição: Flask é um micro framework para Python baseado em Werkzeug e Jinja2. Ele é leve e modular, facilitando a criação de aplicações web e APIs RESTful.
- Versão: 3.0.2

FER02: SQLAlchemy

- Descrição: SQLAlchemy é uma biblioteca de SQL para Python que fornece um conjunto completo de padrões de persistência de banco de dados e ferramentas de ORM (Object Relational Mapper).
- Versão: 2.0.28

FER03: Flask-SQLAlchemy

- Descrição: Extensão para Flask que adiciona suporte ao SQLAlchemy, permitindo a integração de um banco de dados relacional com aplicações Flask.
- Versão: 3.1.1

FER04: Flask-Migrate

- Descrição: Extensão para Flask que facilita o gerenciamento de migrações de banco de dados usando Alembic.
- Versão: 4.0.7

FER05: Alembic

- Descrição: Ferramenta de migração de banco de dados para SQLAlchemy. Permite a criação, manutenção e aplicação de migrações de esquema de banco de dados.
- Versão: 1.13.1

FER06: Flask-Login

- Descrição: Extensão para Flask que gerencia sessões de usuário, incluindo login e logout, garantindo que os usuários estejam autenticados para acessar determinadas rotas.
- Versão: 0.6.3

FER07: Flask-Bcrypt

- Descrição: Extensão para Flask que fornece suporte para hashing de senhas usando Bcrypt, aumentando a segurança do armazenamento de senhas.
- Versão: 1.0.1

FER08: PyMySQL

- Descrição: Biblioteca para conexão com bancos de dados MySQL em Python. Utilizada como driver para interagir com o banco de dados.
- Versão: 1.1.0

FER09: OpenAI

- Descrição: Biblioteca para acessar a API da OpenAI, permitindo a integração de funcionalidades de inteligência artificial e aprendizado de máquina em aplicações Python.
- Versão: 0.28.0

FER10: pytest

- Descrição: Ferramenta de teste para Python que facilita a escrita e execução de testes, garantindo a qualidade do código.
- Versão: 8.2.1

4.1 Linguagens

LING01: Python

- Descrição: Python é a principal linguagem de programação utilizada no projeto para o desenvolvimento do backend. É conhecida por sua simplicidade, legibilidade e vasta gama de bibliotecas e frameworks.
- Uso no Projeto: Desenvolvimento de APIs, lógica de negócios, integração com banco de dados e testes automatizados.

LING02: JavaScript

- Descrição: JavaScript é uma linguagem de programação amplamente utilizada para desenvolvimento frontend. Permite a criação de interfaces de usuário dinâmicas e interativas.
- Uso no Projeto: Desenvolvimento de interfaces de usuário usando React.js, manipulação do DOM e interações assíncronas com o backend.

5.0 Diagrama de Caso de Uso

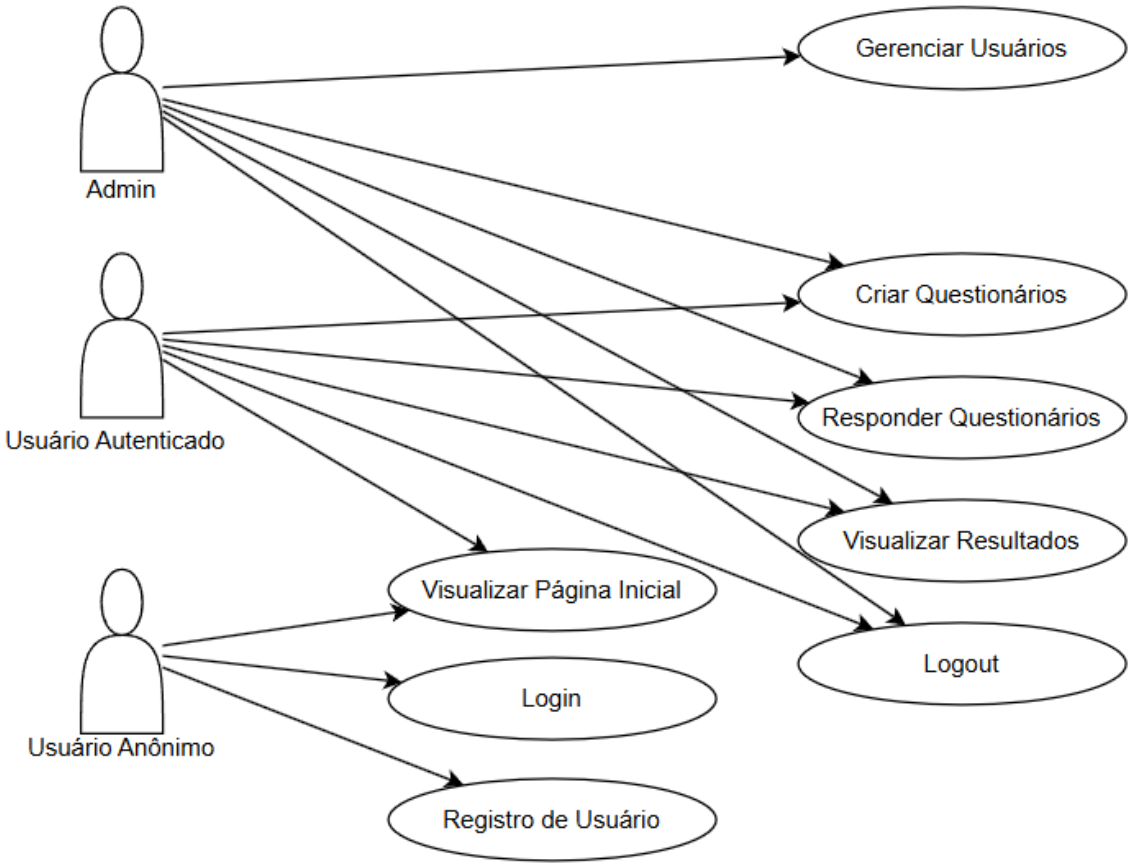

Figura 01 – Diagrama de Caso de Uso

6.0 DIAGRAMA DE SEQUÊNCIA

6.1 INSTITUIÇÃO

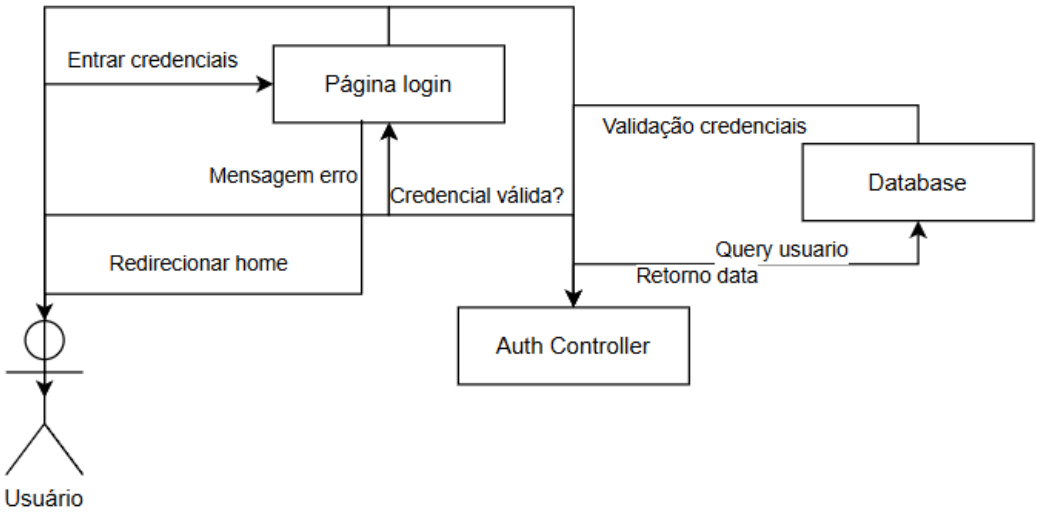

Figura 04 representa DS do momento do login até o final da autenticação e sua nuance-as.

7.0 BANCO DE DADOS

7.1 Modelo lógico

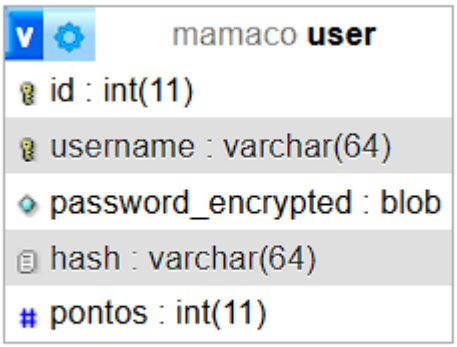

Figura 07 Modelo lógico do banco de dados representando os dos dados armazenados e manipulados.

8. Referências

GRINBERG, Miguel. **Flask Web Development: Developing Web Applications with Python**. 2ª edição. Sebastopol: O'Reilly Media, 2018.

TURNBULL, James. **The Docker Book: Containerization is the new virtualization**. 4ª edição. Melbourne: James Turnbull, 2018.

SKINNER, Joshua; RING, Cody. **Python Web Development with Flask**. 1ª edição. Birmingham: Packt Publishing, 2015.

WINTON, Nickolai. **Mastering Docker**. 3ª edição. Birmingham: Packt Publishing, 2020.

PYMES, Richard. **Flask Documentation**. Disponível em: https://flask.palletsprojects.com/en/2.0.x/. Acesso em: 12 maio 2024.

OPENAI. **OpenAI API Documentation**. Disponível em: https://beta.openai.com/docs/. Acesso em: 23 abril 2024.

DOCKER INC. **Docker Documentation**. Disponível em: https://docs.docker.com/. Acesso em: 03 junho 2024.

SMITH, John. **How to Build and Deploy a Flask Application with Docker**. Medium, 2023. Disponível em: https://medium.com/@johnsmith/how-to-build-and-deploy-a-flask-application-with-docker-1b9c4fbd70fa. Acesso em: 10 junho 2024.

DOE, Jane. **Understanding Flask's Application Context and Request Context**. Real Python, 2022. Disponível em: https://realpython.com/flask-app-context/. Acesso em: 01 junho 2024.

BROWN, Chris. **Creating RESTful Web Services with Flask**. Dev.to, 2023. Disponível em: https://dev.to/chrisbrown/creating-restful-web-services-with-flask-2a8k. Acesso em: 10 maio 2024.