UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MATHEUS SYDOR

OWNEDBOX: UMA PLATAFORMA DE ENSINO PARA SEGURANÇA WEB OFENSIVA E DEFENSIVA

GUARAPUAVA

2025

MATHEUS SYDOR

OWNEDBOX: UMA PLATAFORMA DE ENSINO PARA SEGURANÇA WEB OFENSIVA E DEFENSIVA

OwnedBox: A Teaching Platform for Offensive and Defensive Web Security

Projeto de Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Técnologo em Tecnologia em Sistemas para Internet do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Roni Fabio Banaszewski

GUARAPUAVA

2025

RESUMO

A crescente complexidade dos sistemas digitais e o aumento de ataques cibernéticos tornam essencial o desenvolvimento de ferramentas educacionais que auxiliem estudantes e profissionais a compreender vulnerabilidades e práticas de segurança. Este trabalho apresenta o projeto e desenvolvimento da plataforma **OwnedBox**, uma aplicação web de código aberto voltada ao ensino de segurança ofensiva e defensiva por meio de laboratórios controlados. A solução organiza o conteúdo em módulos práticos que abordam vulnerabilidades recorrentes no OWASP Top 10, como *SQL Injection*, *Cross-Site Scripting* (XSS) e upload de arquivos maliciosos, permitindo ao usuário explorar cenários reais de pentest de forma segura. O sistema implementa autenticação, emissão e validação de tokens, além de um painel de progresso do usuário. O desenvolvimento seguiu metodologias ágeis (Scrum, Kanban e MoSCoW) e utilizou tecnologias como Laravel, Docker e MySQL. Os resultados demonstram que a plataforma oferece um ambiente didático eficaz para aprendizado prático em cibersegurança, contribuindo para a formação de profissionais qualificados na área.

Palavras-chave: cibersegurança; pentest; aplicaes web vulnereis; educao em segu- ran; owasp.

ABSTRACT

The increasing complexity of digital systems and the rise of cyberattacks highlight the need for educational tools that help students and professionals understand vulnerabilities and security practices. This work presents the design and development of **OwnedBox**, an open-source web platform aimed at teaching offensive and defensive web security through controlled laboratories. The solution organizes content into practical modules covering recurring OWASP Top 10 vulnerabilities, such as *SQL Injection*, *Cross-Site Scripting* (XSS), and malicious file uploads, enabling users to safely explore realistic pentesting scenarios. The system implements authentication, token issuance and validation, and a user progress panel. The development followed agile methodologies (Scrum, Kanban, and MoSCoW) and employed technologies such as Laravel, Docker, and MySQL. Results indicate that the platform provides an effective educational environment for practical cybersecurity training, contributing to the qualification of professionals in the field.
Keywords: cybersecurity; pentesting; vulnerable web applications; security education; owasp.

LISTA DE ABREVIATURAS E SIGLAS

Abreviaturas

pentest	Teste de Intrusão (*penetration test*)

Siglas

DOM	*Document Object Model* (Modelo de Objeto de Documento)
GDPR	*General Data Protection Regulation* (Regulamento Geral de Proteção de Dados da União Europeia)
HTB	*Hack The Box*
IoT	*Internet of Things* (Internet das Coisas)
LGPD	Lei Geral de Proteção de Dados
MVP	Produto Mínimo Viável, do inglês *Minimum Viable Product*
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
TCC	Trabalho de Conclusão de Curso
WIP	*Work in Progress* (Trabalho em Progresso)
WWW	*World Wide Web* (Rede Mundial de Computadores)
XSS	*Cross-Site Scripting* (Script entre Sites)

SUMÁRIO

1 INTRODUÇÃO

Este trabalho tem como objetivo principal apresentar o projeto de Trabalho de Conclusão de Curso (TCC) que pretende abordar as principais vulnerabilidades web e o desenvolvimento de um sistema web de aprendizado na área de cibersegurança ofensiva e defensiva.

O sistema proposto tem como objetivo criar um ambiente virtual para que estudantes e entusiastas de área da segurança digital possam aprender e testar suas habilidades.

1.1 Considerações iniciais

A segurança da informação tem se consolidado, nas últimas décadas, como um dos pilares das empresas, governos e sociedade em um mundo cada vez mais conectado com o meio digital. Quando surgiu a ARPANET, no final da década de 1960, inicialmente desenvolvida para fins militares, até a sua popularidade comercial na década de 1990, o meio digital trouxe inúmeros benefícios para a população, mas também abriu espaço para a proliferação de ameaças cibernéticas (CASTELLS, 2003). A popularização da *World Wide Web* (Rede Mundial de Computadores) (WWW), junto ao crescimento do comércio eletrônico e das redes sociais, ampliou consideravelmente os vetores de ataque, fazendo com que a cibersegurança tenha se tornado uma área estratégica para organizações e indivíduos (STALLINGS, 2019)

Por volta de 1980 já existiam estudos e a difusão dos primeiros vírus e softwares maliciosos, impulsionando o surgimento dos antivírus comerciais. Nos anos de 1990 e 2000, as ameaças começaram a evoluir, ataques direcionados a sistemas corporativos, bancos de dados sensíveis e infraestruturas críticas passaram a ser visados por cibercriminosos (KASPERSKY, 2020). Com isso, surgiram formas de prevenir e remediar essas vulnerabilidades, surgiram ferramentas como *firewalls*, sistemas de detecção e prevenção de intrusão, além de leis cada vez mais rígidas, como o *General Data Protection Regulation* (Regulamento Geral de Proteção de Dados da União Europeia) (GDPR) e, no Brasil, a Lei Geral de Proteção de Dados (LGPD) que reforçam a necessidade de proteger o sistema e seus usuários (União Europeia, 2016; BRASIL, 2018).

Nos dias de hoje, fatores como a popularização da *Internet of Things* (Internet das Coisas) (IoT) e o cenário da segurança da informação se tornaram cada vez mais importantes. Impulsionada pela pandemia de COVID-19, desencadeou-se a tendência do trabalho remoto e do comércio online. A proteção de dados tornou-se cada vez mais importante para indivíduos, empresas e governos. Relatórios como OWASP Top Ten (OWASP Foundation, 2021) evidenciam que vulnerabilidades como SQL Injection, *Cross-Site Scripting* (Script entre Sites) (XSS) e upload de arquivos maliciosos permanecem entre as mais exploradas em ataques reais, reforçando a necessidade de mecanismos eficazes de prevenção e mitigação.

Empresas recorrem ao Teste de Intrusão (*penetration test*) (pentest), um teste de segurança cibernética que tem como principal objetivo simular um ataque para encontrar vulne-

rabilidades em um sistema. Este teste é feito de maneira legal e com contrato, garantindo que nenhuma vulnerabilidade explorada causará danos para os clientes da empresa e garantindo que o sistema está seguro (IBM, 2025). Além disso, a prática possui caráter educacional, sendo replicada em laboratórios controlados e plataformas específicas que oferecem ambientes seguros e éticos para estudantes e profissionais desenvolverem suas habilidades. Dessa forma, é possível praticar técnicas ofensivas sem riscos jurídicos ou danos a sistemas de terceiros, em conformidade com o Marco Civil da Internet e a Lei Carolina Dieckmann (BRASIL, 2014; BRASIL, 2012).

Existem diversas iniciativas internacionais, como as plataformas *Hack The Box* (HTB) e TryHackMe, que têm desempenhado um papel importante no ensino de segurança ofensiva e defensiva. No entanto, essas ferramentas apresentam barreiras relacionadas ao idioma, custos de acesso e falta de contextualização com a realidade brasileira. Além disso, são direcionadas prioritariamente ao mercado global, o que reforça a carência, no Brasil, de soluções didáticas acessíveis, em português, capazes de facilitar o aprendizado individual e de atender tanto a ambientes acadêmicos quanto corporativos.

Devido a isso, é importante investir em metodologias inovadoras de ensino em cibersegurança. A segurança digital não é somente um diferencial competitivo entre as empresas, mas uma necessidade. Laboratórios controlados podem ajudar estudantes, entusiastas e profissionais a entenderem as técnicas ofensivas de exploração de vulnerabilidades, e ao mesmo tempo permiti-los a desenvolver habilidades defensivas para proteger sistemas. Tendo isso em vista, a OwnedBox será desenvolvida como uma solução gratuita e de código aberto, voltada ao aprendizado e à capacitação de profissionais em cibersegurança ofensiva. A plataforma permitirá que instituições de ensino e organizações adaptem os laboratórios às suas necessidades específicas e colaborem com a evolução da aplicação web.

1.2 Objetivos

Nesta seção será apresentado o objetivo geral deste trabalho.

1.2.1 Objetivo geral

Desenvolver uma aplicação web denominada **OwnedBox**, que funcione como um laboratório de ensino em cibersegurança ofensiva e defensiva, possibilitando a exploração prática de vulnerabilidades web comuns em um ambiente controlado, seguro e didático.

1.2.2 Objetivos Específicos

Os objetivos específicos do projeto *OwnedBox* foram definidos de forma a orientar o desenvolvimento das principais funcionalidades do sistema e garantir que o protótipo atenda aos propósitos educacionais e técnicos propostos. São eles:

- Implementar o sistema de autenticação e cadastro de usuários.
- Desenvolver módulos de aprendizado ofensivo e defensivo.
- Criar laboratórios interativos com vulnerabilidades simuladas.
- Gerar tokens de validação para comprovar a conclusão dos desafios.
- Construir uma página de perfil com estatísticas e progresso.
- Implementar contramedidas defensivas para cada vulnerabilidade.
- Disponibilizar um ambiente seguro, isolado e instrutivo para prática.

2 TRABALHOS RELACIONADOS

O ensino de segurança da informação tem avançado significativamente nos últimos anos, impulsionado pela disponibilidade de plataformas que simulam ambientes vulneráveis e permitem a prática de técnicas ofensivas e defensivas. Essas soluções têm contribuído para o aprendizado prático de estudantes, profissionais e entusiastas, mas muitas delas apresentam barreiras relacionadas ao idioma, aos custos de acesso ou à falta de contextualização com a realidade acadêmica brasileira. Nesse cenário, destaca-se a importância de analisar ferramentas amplamente utilizadas na área, especialmente aquelas que influenciaram a concepção da **OwnedBox**. Entre as principais iniciativas encontram-se o Hack The Box (HTB), o TryHackMe (THM) e o Damn Vulnerable Web Application (DVWA), cada uma com características próprias e relevância significativa no ecossistema de cibersegurança.

2.1 Hack The Box (HTB)

O *Hack The Box* é uma das plataformas mais reconhecidas internacionalmente para prática de *pentest* e *red teaming*. Lançado em 2017, o HTB oferece máquinas virtuais com diferentes níveis de vulnerabilidades, oferecendo aos alunos a oportunidade de praticar técnicas de exploração em um ambiente controlado e constantemente atualizado (Hack The Box Ltd., 2025)

Pontos Positivos:

- Grande quantidade de laboratórios e desafios, com atualização frequente.
- Comunidade ativa e fóruns de suporte.
- Reconhecimento no mercado, sendo referência em treinamentos de cibersegurança.

Pontos Negativos:

- Interface totalmente em inglês, dificultando o uso por iniciantes brasileiros.
- Curva de aprendizado elevada, exigindo conhecimentos prévios.
- Muitas funcionalidades só estão disponíveis mediante assinatura paga.

2.2 TryHackMe

O *TryHackMe* é uma plataforma de ensino de segurança cibernética com foco em *gamificação* e aprendizado gradual. Ela disponibiliza laboratórios com tutoriais guiados e desafios práticos. Ela se destaca pelo foco em atividades do tipo *Capture The Flag* (CTF), modalidade

competitiva em que os participantes exploram falhas em laboratórios virtuais para conquistar pontos e melhorar sua posição em rankings globais (TRYHACKME, 2025)

Pontos Positivos:

- Abordagem didática com tutoriais passo a passo.
- Estrutura de aprendizado organizada em trilhas temáticas.
- Requer apenas um navegador para começar a usar (sem necessidade de configuração avançada).

Pontos Negativos:

- Versão gratuita limitada, com acesso restrito a laboratórios.
- Conteúdo em inglês, o que pode ser um obstáculo para estudantes brasileiros.

2.3 DVWA (Damn Vulnerable Web Application)

O *DVWA* é uma aplicação web propositalmente vulnerável, desenvolvida para fins educacionais. É uma ferramenta bastante utilizada em cursos de segurança, permitindo a exploração de falhas em diferentes níveis de dificuldade (RandomStorm, 2025).

Pontos Positivos:

- Gratuito e de código aberto.
- Permite testar vulnerabilidades clássicas da web (SQL Injection, XSS, File Upload, CSRF, etc.).
- Flexibilidade para instalação em ambiente local.

Pontos Negativos:

- Interface simples e pouco amigável.
- Ausência de trilhas pedagógicas organizadas, dificultando o uso em contexto acadêmico formal.

3 REFERENCIAL TEÓRICO

O desenvolvimento da plataforma **OwnedBox** requer a compreensão dos princípios, métodos e tecnologias que sustentam sua concepção e execução. Este capítulo apresenta os fundamentos teóricos e técnicos que orientam o projeto, abordando as metodologias adotadas no processo de desenvolvimento, as ferramentas de apoio à gestão e as tecnologias que compõem a base da aplicação. Entre os principais recursos empregados estão as metodologias **MoSCoW**, **Kanban** e **Gitflow**, responsáveis, respectivamente, pela priorização de requisitos, organização das atividades e controle de versionamento, além do uso das tecnologias **Laravel (Blade)**, **MySQL**, **Docker** e **Figma**, que estruturam o ambiente técnico da solução.

A criação de uma aplicação web funcional e segura demanda a escolha criteriosa de frameworks, linguagens e práticas de engenharia de software que garantam qualidade, estabilidade e flexibilidade para futuras expansões. O uso de contêineres e ambientes isolados com **Docker** assegura portabilidade e consistência nas execuções, enquanto o **Laravel** e o **MySQL** oferecem uma base sólida para o desenvolvimento da lógica de negócio e a gestão eficiente de dados.

Além disso, são considerados os processos e ferramentas que favorecem o planejamento e o acompanhamento contínuo do projeto, desde a modelagem das interfaces até a entrega de funcionalidades em ciclos curtos e iterativos. Essa abordagem integrada visa estabelecer uma estrutura organizada e sustentável, capaz de garantir a evolução constante e o aprimoramento contínuo da solução proposta.

3.1 Processos de desenvolvimento

3.1.1 Kanban

O Kanban é uma metodologia ágil voltada à gestão visual de processos, desenvolvida originalmente no contexto do sistema de produção da Toyota e posteriormente adaptada para o desenvolvimento de software. Seu princípio fundamental é o controle do fluxo de trabalho por meio da visualização das tarefas em um quadro dividido em colunas que representam as etapas do processo, como A Fazer, Em Progresso e Concluído. Essa abordagem permite acompanhar o andamento das atividades de forma intuitiva, identificar gargalos e otimizar a distribuição de tarefas dentro da equipe (ANDERSON, 2010).

A aplicação do Kanban tem como base três pilares: visualização do trabalho, limitação do *Work in Progress* (Trabalho em Progresso) (WIP) e melhoria contínua. A visualização oferece transparência e compreensão coletiva do status das tarefas; a limitação do WIP impede a sobrecarga de tarefas simultâneas, garantindo foco e fluidez; e a melhoria contínua incentiva ajustes incrementais no fluxo, buscando eficiência e qualidade nas entregas.

Diferentemente de metodologias mais prescritivas, o Kanban é flexível e se adapta ao contexto de cada equipe, permitindo implementar mudanças gradualmente sem interromper o processo em andamento. Essa característica o torna especialmente útil em projetos de desenvolvimento de software, onde demandas e prioridades podem variar de forma dinâmica ao longo do ciclo de vida do produto.

3.1.2 Scrum

O *Scrum* é um framework ágil voltado para o gerenciamento e desenvolvimento de projetos complexos, especialmente em contextos de software. Criado por Ken Schwaber e Jeff Sutherland na década de 1990, o Scrum baseia-se em ciclos iterativos e incrementais denominados *sprints*, que possuem duração fixa e têm como objetivo a entrega contínua de partes funcionais do produto.

O *framework* promove a adaptação rápida às mudanças e incentiva a comunicação constante entre os membros da equipe. O processo é guiado por três papéis principais: o *Product Owner*, responsável por definir e priorizar os requisitos do produto; o *Scrum Master*, que atua como facilitador e garante o cumprimento das práticas do *framework*; e o *Development Team*, equipe responsável pela implementação e entrega dos incrementos do produto.

3.1.3 MoSCow

O método MoSCoW é uma técnica de priorização utilizada no gerenciamento ágil de projetos, especialmente em contextos que exigem definição clara de prioridades entre funcionalidades e requisitos. O nome deriva das iniciais das categorias que compõem sua estrutura: Must Have (deve ter), Should Have (deveria ter), Could Have (poderia ter) e Won't Have (não terá, pelo menos neste momento). Essa classificação auxilia na tomada de decisões sobre o que deve ser implementado de forma imediata, o que pode ser adiado e o que deve ser excluído do escopo atual.

A principal finalidade do MoSCoW é equilibrar as expectativas entre a equipe de desenvolvimento e as partes interessadas, assegurando que os recursos mais críticos para o sucesso do projeto recebam prioridade. O método favorece uma comunicação mais clara sobre o escopo e as entregas, reduzindo ambiguidades e facilitando o planejamento de sprints e versões. Além disso, sua simplicidade torna-o compatível com diferentes metodologias ágeis, como Scrum e Kanban, podendo ser integrado facilmente ao ciclo de desenvolvimento (CLEGGE; MOSS, 2004).

No contexto de projetos de software, o MoSCoW contribui para a gestão eficiente do tempo e dos recursos, permitindo que o produto evolua de forma incremental, sem comprometer

sua funcionalidade essencial. Ao adotar esse método, é possível garantir que o foco permaneça nas entregas que agregam maior valor ao usuário e ao objetivo geral do sistema.

3.2 Ferramenta de desenvolvimento

3.2.1 Git e GitFlow

O Git é um sistema de controle de versão distribuído amplamente utilizado no desenvolvimento de software para registrar e gerenciar o histórico de alterações em projetos de código-fonte. Criado por Linus Torvalds em 2005, o Git permite que múltiplos desenvolvedores trabalhem simultaneamente em diferentes partes de um mesmo projeto, preservando a integridade do código e facilitando o rastreamento de mudanças. Por meio de sua estrutura descentralizada, cada colaborador mantém uma cópia completa do repositório, o que proporciona maior segurança, flexibilidade e independência no processo de desenvolvimento.

Para aprimorar a organização do fluxo de trabalho dentro do Git, foi proposto o modelo Gitflow, desenvolvido por Vincent Driessen em 2010. Esse modelo define uma convenção de ramificações (branches) que orienta o ciclo de vida do software, separando claramente as fases de desenvolvimento, teste e entrega. No Gitflow, as principais ramificações são *main* (ou *master*), responsável pela versão estável do sistema, e *develop*, que concentra as funcionalidades em desenvolvimento. Além dessas, o modelo utiliza ramificações auxiliares, como *feature* (para novas funcionalidades), *release* (para preparação de versões) e *hotfix* (para correção de erros em produção).

3.2.2 GitHub

O GitHub é uma plataforma online de hospedagem e colaboração de código-fonte baseada no sistema de controle de versão Git. Criada em 2008, a ferramenta permite que desenvolvedores armazenem, compartilhem e gerenciem projetos de software de forma integrada, oferecendo recursos como controle de versões, revisão de código, acompanhamento de problemas (issues) e automação de fluxos de trabalho. Além de facilitar o trabalho colaborativo, o GitHub serve como repositório central para projetos de código aberto e privados, promovendo integração contínua e rastreabilidade no desenvolvimento de software (GitHub, 2025a).

3.2.3 GitHubProjects

O GitHub Projects é uma ferramenta integrada à plataforma GitHub que permite organizar e gerenciar tarefas de forma visual por meio de quadros (*boards*), listas e cartões. Seu objetivo é apoiar equipes no acompanhamento de atividades, planejamento de sprints e priori-

zação de funcionalidades, facilitando a colaboração e o controle do progresso dos projetos de software dentro do ambiente do repositório (GitHub, 2025b).

3.3 Tecnologias do lado do cliente

3.3.1 JavaScript

O JavaScript é uma linguagem de programação de alto nível, interpretada e orientada a eventos, amplamente utilizada no desenvolvimento web para tornar páginas e aplicações interativas. Criada em 1995, permite a manipulação de elementos do *Document Object Model* (Modelo de Objeto de Documento) (DOM), controle de eventos e comunicação assíncrona com servidores, sendo compatível com todos os principais navegadores (FLANAGAN, 2020).

3.4 Tecnologias do lado do servidor

3.4.1 Laravel

O Laravel é um *framework* PHP de código aberto voltado para o desenvolvimento de aplicações web, que oferece uma arquitetura organizada e recursos avançados para a criação de sistemas robustos, seguros e escaláveis. Sua proposta é simplificar tarefas comuns no desenvolvimento web, como roteamento, autenticação, acesso a banco de dados, envio de e-mails e gerenciamento de sessões, proporcionando maior produtividade ao desenvolvedor.

Entre suas principais funcionalidades, destaca-se o *Blade*, mecanismo de templates nativo do Laravel, responsável por separar a camada de apresentação da lógica de negócio. O Blade opera no lado do servidor, processando diretivas, variáveis e estruturas de controle antes de gerar o HTML final enviado ao cliente. Essa abordagem permite a criação de páginas dinâmicas de forma eficiente, com suporte a layouts reutilizáveis, seções e componentes, promovendo organização, consistência visual e facilidade de manutenção das interfaces (Laravel, 2025).

O Laravel também conta com uma série de ferramentas oficiais, como o *Laravel Breeze*, que fornece uma implementação minimalista de autenticação. Essas soluções integradas reforçam a filosofia do framework de oferecer um ecossistema completo, modular e adequado tanto para iniciantes quanto para aplicações de maior complexidade.

3.4.2 MySQL

O MySQL é um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) de código aberto, amplamente utilizado no desenvolvimento de aplicações web. Ele permite o armazenamento, consulta e manipulação de dados estruturados por meio da linguagem *Structured Query*

Language (Linguagem de Consulta Estruturada) (SQL). O MySQL oferece confiabilidade, desempenho e suporte a transações, além de possibilitar o gerenciamento de múltiplos usuários e a integração com diversos *frameworks* e linguagens de programação. Sua popularidade se deve à facilidade de uso, escalabilidade e compatibilidade com diferentes ambientes de desenvolvimento.

3.4.3 Docker

O Docker é uma plataforma de código aberto voltada para a containerização de aplicações, permitindo que software e suas dependências sejam empacotados em contêineres isolados. Cada contêiner é executado de forma consistente em diferentes ambientes, garantindo portabilidade, reprodutibilidade e isolamento do sistema operacional subjacente. No desenvolvimento de software, o Docker facilita a configuração de ambientes, reduz conflitos de dependências e possibilita a integração contínua e o gerenciamento eficiente de serviços e aplicações (MERKEL, 2014).

4 MATERIAIS E MÉTODOS

4.0.1 Materiais

O projeto **OwnedBox** será desenvolvido utilizando o **framework Laravel**, que oferece uma estrutura robusta e segura para a criação de aplicações web modernas. O **Blade**, motor de templates nativo do Laravel, será empregado para a construção das interfaces dinâmicas e responsivas, permitindo a integração entre o front-end e o back-end de forma eficiente.

O banco de dados **MySQL** será adotado para o armazenamento das informações, devido à sua confiabilidade, desempenho e ampla compatibilidade com o Laravel. Para o controle de versão do código, será utilizado o **Git**, com o repositório hospedado na plataforma **GitHub**, possibilitando o acompanhamento das alterações e a colaboração entre desenvolvedores.

O **Docker** será implementado para a criação de ambientes de desenvolvimento padronizados, assegurando que a aplicação funcione de maneira consistente em diferentes sistemas operacionais. Além disso, o método **Kanban** no **Github Projects** será utilizado para o gerenciamento das tarefas e acompanhamento do progresso do projeto, promovendo uma organização visual e clara das etapas de desenvolvimento.

4.0.2 Métodos

O desenvolvimento do sistema **OwnedBox** será conduzido de forma iterativa e incremental, seguindo princípios das metodologias ágeis.

Inicialmente, será realizada uma etapa de levantamento e refinamento dos requisitos, permitindo identificar as funcionalidades essenciais do sistema e compreender as necessidades dos usuários. Em seguida, será feita a modelagem preliminar do banco de dados, assim como o planejamento das telas, rotas, e componentes que serão implementados no *framework* Laravel. A camada de apresentação será estruturada com Blade, priorizando critérios de usabilidade, clareza e experiência do usuário.

A gestão das tarefas será realizada por meio do método Kanban, utilizando quadros no *GitHub Projects*. As atividades serão distribuídas em colunas representando seu estado durante cada *sprint*, como **To Do**, **In Progress**, **Review** e **Done**. Esse modelo de acompanhamento permitirá visualizar o progresso e priorizar demandas conforme o avanço do desenvolvimento.

A condução do projeto seguirá uma adaptação da metodologia *Scrum*, adequada ao contexto de desenvolvimento individual. As *sprints* terão duração de duas semanas, e o orientador desempenhará os papéis de *Product Owner* e *Scrum Master*, enquanto o aluno atuará como *Developer*, responsável pela implementação das funcionalidades e acompanhamento do andamento das atividades.Com essa combinação de métodos, espera-se conduzir o desenvolvimento do sistema de forma organizada e incremental.

5 ANÁLISE DO SISTEMA

Este trabalho propõe o desenvolvimento da aplicação web **OwnedBox**, um ambiente de ensino gratuito de código aberto voltado para a prática de segurança web ofensiva e defensiva. O sistema será disponibilizado como um **protótipo funcional Produto Mínimo Viável, do inglês *Minimum Viable Product* (MVP)**, contemplando três vulnerabilidades web entre as mais recorrentes segundo o OWASP Top 10 (OWASP Foundation, 2021): *SQL Injection*, *Cross-Site Scripting* e *Upload de Arquivos Maliciosos*. Cada vulnerabilidade será apresentada em um módulo específico, permitindo que o estudante explore a falha ofensivamente, receba um **token de validação** ao concluir o desafio e, em seguida, visualize contramedidas defensivas que demonstrem como corrigir o problema.

A aplicação será distribuída como **código aberto no GitHub**, acompanhada de *snapshots* das máquinas de laboratório, possibilitando que qualquer usuário instale o sistema em seu próprio ambiente local ou servidor. Para viabilizar essa instalação, será incluída uma **página de instruções com os requisitos necessários**, de modo a facilitar a configuração mesmo por usuários com pouca experiência em implantação de aplicações web. Essa escolha elimina a necessidade de manter a plataforma publicada em um domínio público, reduz riscos de segurança e reforça o caráter acadêmico e didático do projeto.

Por se tratar de um projeto de **código aberto**, os utilizadores poderão contribuir ativamente com a evolução da aplicação, propondo melhorias, corrigindo falhas e desenvolvendo **laboratórios personalizados** que atendam a demandas específicas. Essa abertura à colaboração amplia o potencial educacional da plataforma, tornando-a um recurso dinâmico e em constante aperfeiçoamento pela comunidade.

Além disso, a aplicação contará com uma **página de estatísticas do usuário**, no qual será possível acompanhar o progresso individual, visualizar os tokens coletados e obter uma visão clara do desempenho durante os estudos.

5.1 Perfis de usuários

A aplicação web contempla, nesta etapa de desenvolvimento, apenas um perfil de usuário: o **Estudante** (ou Usuário Comum). Esse perfil é responsável por realizar o cadastro e autenticação na plataforma, acessar os módulos de aprendizado, explorar as vulnerabilidades simuladas e visualizar os tokens de validação obtidos ao concluir os desafios propostos.

5.2 Funcionalidades essenciais

Nesta seção são apresentadas as histórias de usuário classificadas de acordo com o método MoSCoW. Para o escopo deste projeto, foram priorizados apenas os requisitos *Must*

Have, considerados essenciais para o funcionamento mínimo viável do sistema. Os demais requisitos — *Should Have*, *Could Have* e *Won't Have* — encontram-se detalhados no Apêndice A.

Cada funcionalidade está apresentada no formato de História de Usuário, identificada por um código no padrão HUxx (História de Usuário), como HU01, HU02, HU03, e assim por diante, o que possibilita uma melhor organização e referência ao longo do trabalho.

- **HU01**: **Como** um usuário, **quero** realizar meu cadastro informando dados básicos (nome, e-mail e senha), **para** que eu possa acessar a aplicação web e participar dos módulos de aprendizado.
- **HU02**: **Como** usuário cadastrado, **quero** efetuar login com minhas credenciais, **para** acessar meu perfil e os módulos de aprendizado.
- **HU03**: **Como** usuário autenticado, **quero** acessar meu perfil com informações de desempenho, **para** acompanhar meu progresso na aplicação web.
- **HU04**: **Como** estudante de cibersegurança, **quero** acessar laboratórios com vulnerabilidades simuladas, **para** praticar técnicas ofensivas em ambiente controlado.
- **HU05**: **Como** usuário da aplicação web, **quero** visualizar contramedidas para cada vulnerabilidade, **para** aprender como corrigir falhas e aplicar boas práticas defensivas.
- **HU06**: **Como** usuário que conclui um desafio, **quero** receber um token de validação, **para** comprovar que atingi o objetivo proposto.
- **HU07**: **Como** usuário da plataforma, **quero** acessar relatórios de desempenho, **para** acompanhar minha evolução ao longo do tempo.
- **HU08**: **Como** usuário da OwnedBox, **quero** ter garantia de que meus dados estão protegidos, **para** utilizar a aplicação web sem riscos.
- **HU09**: **Como** usuário, **quero** ter acesso a uma página com instruções de instalação e requisitos do sistema, **para** configurar a aplicação web corretamente em meu ambiente local.

6 PROJETO DO SISTEMA

Este capítulo apresenta o projeto do **OwnedBox**, descrevendo como o sistema será organizado e preparado para sua implementação. São detalhadas as decisões iniciais de arquitetura, o planejamento das primeiras sprints, os protótipos de interface e a modelagem do banco de dados, estabelecendo a estrutura necessária para o desenvolvimento da aplicação. O objetivo é definir, de forma clara e objetiva, os principais componentes do sistema e o fluxo de trabalho que orientará sua construção.

6.1 Identidade Visual

A identidade visual do *OwnedBox* foi concebida para refletir o propósito central da plataforma: oferecer um ambiente seguro, intuitivo e imersivo para o estudo de vulnerabilidades e práticas de pentest. Assim como apresentado na análise anterior, o sistema prioriza uma estética moderna alinhada a plataformas de cibersegurança, garantindo clareza na navegação, coerência visual e foco no conteúdo técnico. O design adotado também aproxima o usuário da experiência típica de laboratórios profissionais, como os utilizados em treinamentos de red team e ambientes práticos de segurança ofensiva.

6.1.1 Paleta de Cores

A paleta de cores do *OwnedBox* foi definida para transmitir profissionalismo, tecnologia e concentração. Inspirada em ambientes comparáveis ao *Hack The Box*, a interface utiliza predominantemente tons escuros, permitindo ao usuário explorar os laboratórios por longos períodos sem desconforto visual. As principais escolhas são:

- **Background:** tons de cinza-escuro e preto suave, fornecendo contraste adequado e criando um ambiente imersivo semelhante a terminais e interfaces técnicas.
- **Texto e ícones:** branco e cinza claro, garantindo legibilidade em ambiente dark.
- **botões e barras de progresso:** azul claro e verde limão, destacando visualmente o avanço do usuário nos módulos.

A Tabela 1 apresenta a paleta de cores utilizada, com seus respectivos códigos e representações visuais.

Tabela 1 – Paleta de cores do OwnedBox

Elemento	Cor	Hexadecimal
Background principal		#121c21
Cinza secundário		#192b33
Texto branco		#FFFFFF
Texto cinza claro		#D1D5DB
Azul (progresso e botões)		#1294D4
verde limão		#56D412

Essa paleta equilibra estética e funcionalidade. Os tons escuros favorecem o foco e reduzem distrações, enquanto as cores destacadas reforçam a percepção de ação, progresso e segurança dentro da plataforma.

6.1.2 Tipografia

A tipografia utilizada no *OwnedBox* prioriza legibilidade e modernidade, características essenciais em sistemas de estudo técnico. A fonte adotada segue o padrão de interfaces minimalistas amplamente utilizadas em dashboards e plataformas educacionais de segurança da informação. A escolha reforça a clareza e a precisão esperada em aplicações voltadas para profissionais e estudantes da área.

6.1.3 Logotipo

O logotipo do *OwnedBox* é apresentado na Figura 1. Ele foi desenvolvido com o objetivo de refletir simultaneamente segurança, investigação e exploração, sendo esses os pilares fundamentais do sistema. O cubo presente no símbolo contém elementos como um cadeado, um escudo e uma lupa, representando análise e busca por vulnerabilidades, respectivamente. Sua composição geométrica e as cores utilizadas alinham-se ao tema geral do sistema, reforçando a sensação de ambiente técnico e confiável.

Figura 1 – Logotipo do sistema OwnedBox.

6.2 Planejamento das primeiras sprints

O planejamento das primeiras sprints do projeto **OwnedBox** estabelece a organização inicial das atividades que serão realizadas no início do desenvolvimento. Nesta seção, são apresentadas as sprints fundamentais que estruturam o início do trabalho, detalhando as entregas previstas, as prioridades e os objetivos de cada ciclo.

6.2.1 Preparação do ambiente de desenvolvimento (Sprint 0)

A *Sprint 0*, com duração estimada de 5 dias, tem como objetivo configurar e padronizar o ambiente de desenvolvimento, garantindo que todos os recursos necessários estejam operacionais e integrados. Embora não gere uma entrega direta ao usuário final, essa etapa é indispensável para assegurar a estabilidade e produtividade das próximas sprints. As principais atividades planejadas incluem:

- Instalação e configuração do *framework* Laravel, incluindo dependências e bibliotecas essenciais para o funcionamento da aplicação.
- Configuração do ambiente de containerização utilizando o Docker, assegurando consistência entre os ambientes de desenvolvimento e produção.
- Criação e modelagem inicial do banco de dados em MySQL, definindo tabelas e relacionamentos principais.
- Integração com o sistema de controle de versão Git e configuração de repositório remoto no GitHub.

Ao final da *Sprint 0*, o ambiente estará devidamente preparado para suportar o desenvolvimento ágil e colaborativo do projeto.

6.2.2 Funcionalidades iniciais (Sprint 1)

A *Sprint 1*, com duração de duas semanas, será responsável pela primeira entrega funcional ao usuário. Nesta etapa, o foco será o desenvolvimento das telas de **login**, **cadastro** e **módulos de aprendizado**, fundamentais para o acesso e navegação dentro da plataforma OwnedBox. As principais atividades previstas são:

- Criação das interfaces de login e cadastro utilizando o mecanismo de templates *Blade*.
- Estruturação da tela de módulos, onde o usuário poderá visualizar os temas disponíveis em segurança ofensiva e defensiva.

Ao término da *Sprint 1*, espera-se que o sistema **OwnedBox** esteja apto a permitir o registro, autenticação e navegação inicial do usuário, estabelecendo a base necessária para o desenvolvimento dos módulos práticos e laboratórios nas próximas sprints.

6.2.3 Criação dos laboratórios práticos (Sprint 2)

A *Sprint 2*, com duração de duas semanas, será focada exclusivamente na implementação dos laboratórios práticos utilizados para o ensino de cibersegurança ofensiva e defensiva. Nesta etapa, o objetivo principal será estruturar os ambientes isolados em *containers Docker*, que servirão como base para os desafios propostos aos usuários dentro da plataforma OwnedBox.

As principais atividades previstas são:

- Configuração dos laboratórios práticos utilizando *Docker*, assegurando ambientes isolados, controlados e reproduzíveis.
- Preparação de cenários de vulnerabilidades comuns, tanto ofensivas quanto defensivas, alinhados ao conteúdo dos módulos de aprendizado.
- Integração inicial dos laboratórios com o painel do usuário, permitindo que os ambientes possam ser iniciados a partir da plataforma.
- Definição dos requisitos técnicos e operacionais necessários para o funcionamento adequado de cada laboratório.

Ao término da *Sprint 2*, espera-se que os laboratórios estejam funcionalmente configurados e acessíveis, estabelecendo a base prática essencial para a validação de conhecimento dos usuários nas próximas sprints.

6.2.4 Implementação do sistema de tokens e validação de progresso (Sprint 3)

A *Sprint 3*, também planejada para duas semanas, será dedicada ao desenvolvimento de toda a lógica responsável pela emissão, gerenciamento e validação de **tokens** associados à conclusão dos laboratórios. Esses tokens permitirão que o sistema reconheça automaticamente o progresso do usuário e registre sua evolução dentro da plataforma OwnedBox.

As principais atividades previstas são:

- Implementação da lógica de geração de tokens únicos ao término de cada laboratório.
- Desenvolvimento do mecanismo de validação de tokens na plataforma, garantindo autenticidade e integridade das informações.

- Integração do sistema de tokens com o painel de progresso do usuário, atualizando automaticamente o status dos módulos concluídos.
- Testes de segurança e consistência para assegurar que tokens inválidos ou repetidos não sejam aceitos.

Ao término da *Sprint 3*, a plataforma OwnedBox será capaz de acompanhar o desempenho dos usuários por meio de tokens validados, possibilitando a construção de uma trilha de aprendizado consistente e segura.

6.3 Prototipação de telas

Com base nos requisitos levantados e priorizados pela técnica *MoSCoW*, foram desenvolvidos os protótipos de interface do sistema utilizando a ferramenta *Figma*.

A prototipação teve como objetivo representar, de forma visual e interativa, os principais fluxos de navegação da aplicação, dando uma visão da experiência do usuário antes da implementação definitiva.

Nesta seção, são apresentados os protótipos das telas fundamentais do sistema, incluindo: criação de conta, autenticação, módulos disponíveis, perfil do usuário e sessões de vulnerabilidade. Todas seguem o padrão visual definido para o projeto, priorizando simplicidade, coerência visual e clareza nos fluxos de interação.

6.3.1 Tela de criação de conta

A tela de criação de conta foi projetada para permitir o registro de novos usuários, solicitando informações essenciais como nome, e-mail e senha. O layout prioriza a clareza dos campos e o fluxo natural de preenchimento, além de incluir mensagens de validação para garantir a segurança das informações inseridas.

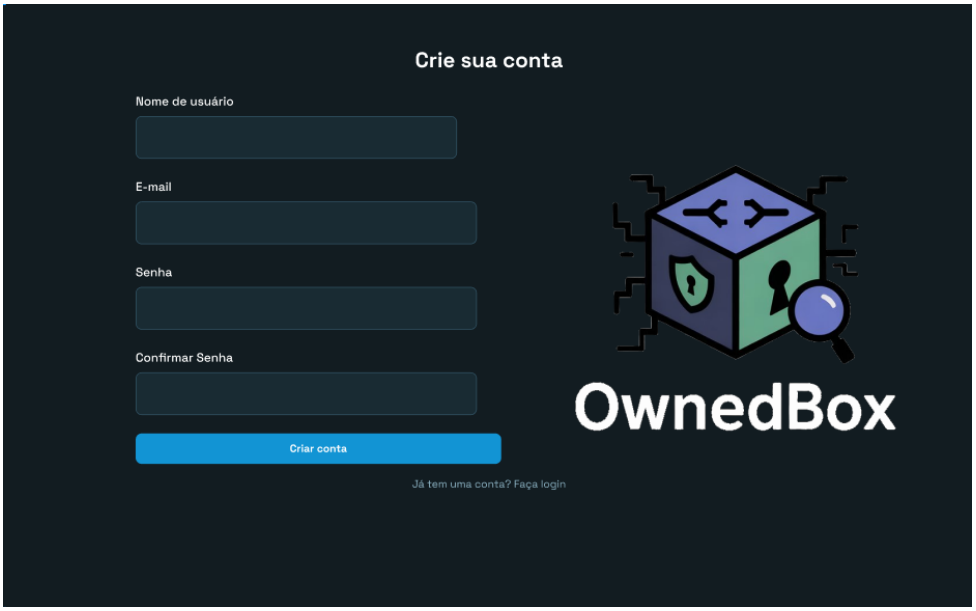

Figura 2 – Protótipo da tela de criação de conta desenvolvido no Figma.

6.3.2 Tela de login

A tela de login foi desenhada para oferecer uma autenticação simples e direta, mantendo consistência visual com a tela de registro. Além dos campos de e-mail e senha, foi previsto um sistema de aviso para tentativas inválidas, conforme o requisito de segurança definido na etapa de levantamento.

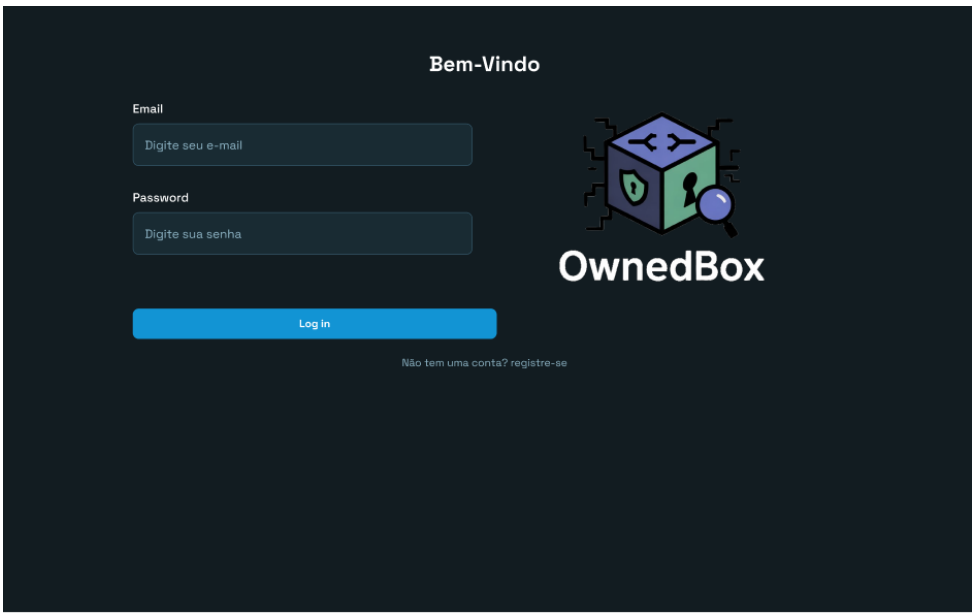

Figura 3 – Protótipo da tela de login desenvolvido no Figma.

6.3.3 Tela de módulos

A tela de módulos apresenta os desafios de aprendizado disponíveis para o usuário, relacionados a diferentes tipos de vulnerabilidades de segurança. Cada módulo inclui uma breve descrição da vulnerabilidade e uma indicação visual de progresso, permitindo que o usuário acompanhe seu desempenho ao longo do curso. O design segue o padrão de cores, tipografia e navegação já estabelecido nas telas de cadastro e login, mantendo a consistência visual do sistema.

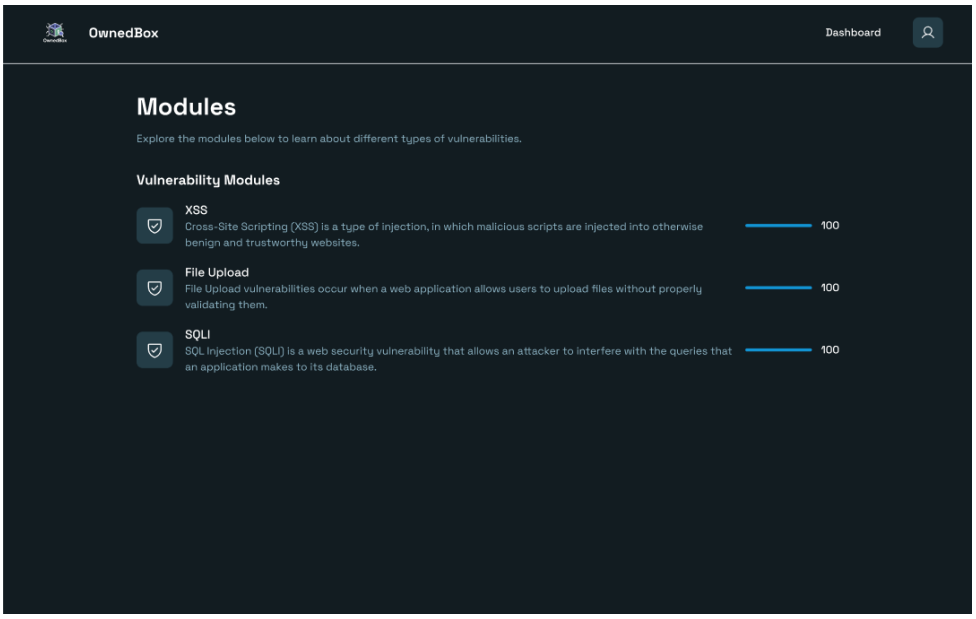

Figura 4 – Protótipo da tela de módulos desenvolvido no Figma.

6.3.4 Tela de perfil

A tela de perfil permite que o usuário visualize e edite suas informações pessoais, como nome, e-mail e senha. Além disso, a interface inclui uma barra de progresso que indica a porcentagem total de módulos concluídos pelo usuário, fornecendo uma visão rápida do seu desempenho no curso. O design mantém a consistência visual com as telas anteriores, utilizando o mesmo esquema de cores, tipografia e elementos de navegação.

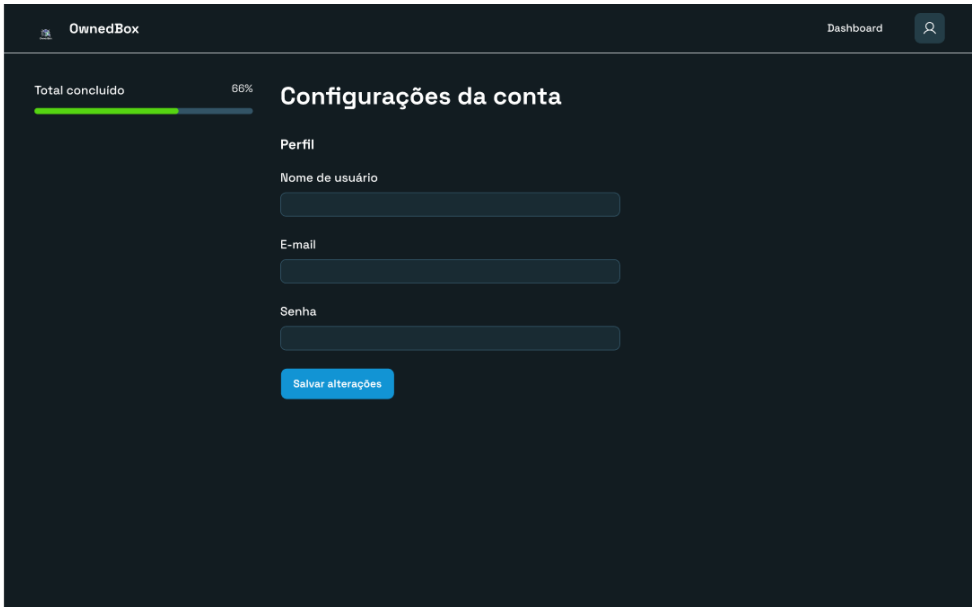

Figura 5 – Protótipo da tela de perfil desenvolvido no Figma.

6.3.5 Tela de sessão de vulnerabilidade

A tela de sessão de vulnerabilidade apresenta ao usuário informações detalhadas sobre a vulnerabilidade abordada, incluindo uma descrição teórica e instruções práticas para o laboratório. O layout inclui campos interativos, como o botão para gerar instâncias de teste e o campo para submissão de tokens, permitindo que o aluno pratique os conceitos aprendidos de forma segura. O design mantém consistência visual com as outras telas do sistema, utilizando o mesmo esquema de cores, tipografia e elementos de navegação.

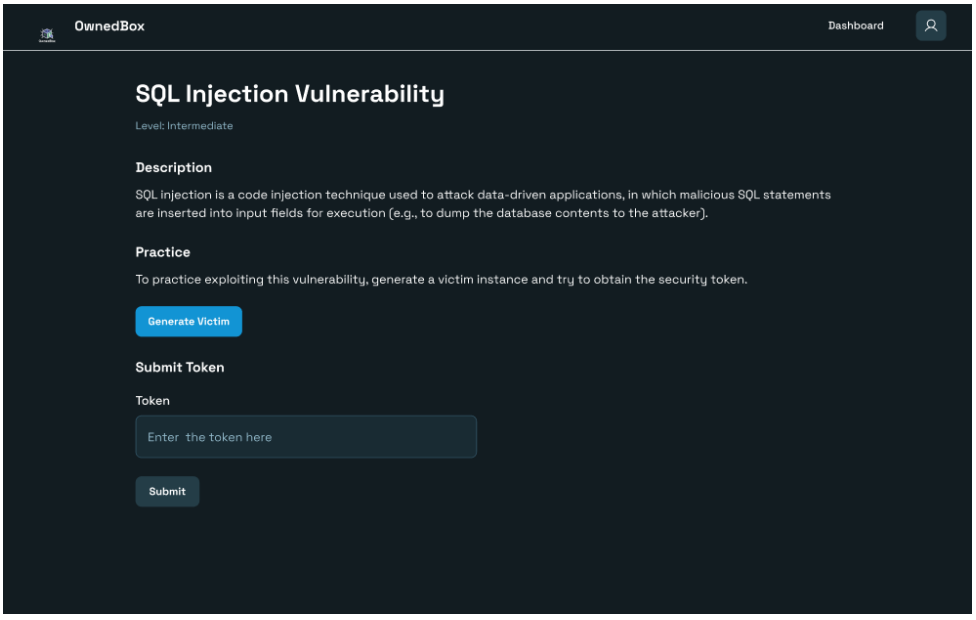

Figura 6 – Protótipo da tela de sessão de vulnerabilidade desenvolvido no Figma.

6.4 Modelagem do Banco de Dados

Para o desenvolvimento do sistema, foi elaborada a modelagem do banco de dados responsável por armazenar as informações de usuários, módulos e os módulos vinculados a cada um deles. A modelagem priorizou a simplicidade e a integridade dos dados, garantindo compatibilidade com o sistema de autenticação já implementado no *Laravel Breeze*.

A estrutura proposta é composta por três entidades principais: **users**, **module** e **owned_modules**. A tabela **users** foi aproveitada diretamente do Breeze, sendo responsável por gerenciar o cadastro e a autenticação dos usuários. Essa decisão reduz a duplicidade de informações e aproveita a infraestrutura nativa de autenticação segura já existente no *framework*.

A tabela **module** armazena os módulos disponíveis no sistema, contendo um identificador, o nome e uma descrição para cada módulo. Essa entidade tem como objetivo centralizar as informações sobre os conteúdos ou atividades que o usuário poderá acessar.

A tabela **owned_modules** tem papel intermediário, representando o relacionamento entre as entidades **users** e **module**. Cada registro nessa tabela indica que um usuário possui determinado módulo, armazenando a data de aquisição e um valor booleano que confirma a posse do módulo. Essa estrutura permite identificar quais módulos cada usuário possui e serve como base para o controle de progresso e funcionalidades futuras.

Optou-se por utilizar uma relação do tipo um-para-muitos (1:N) entre as tabelas **users** e **owned_modules**, e também entre **module** e **owned_modules**. Essa abordagem reflete corretamente o comportamento esperado, visto que um usuário pode possuir diversos módulos, e cada módulo pode estar vinculado a múltiplos usuários.

A escolha dessa modelagem se mostrou vantajosa por oferecer um equilíbrio entre simplicidade e escalabilidade. Além de reduzir a complexidade da estrutura relacional, ela possibilita futuras expansões como o registro de progresso, notas ou tempo de conclusão sem a necessidade de grandes alterações no modelo existente.

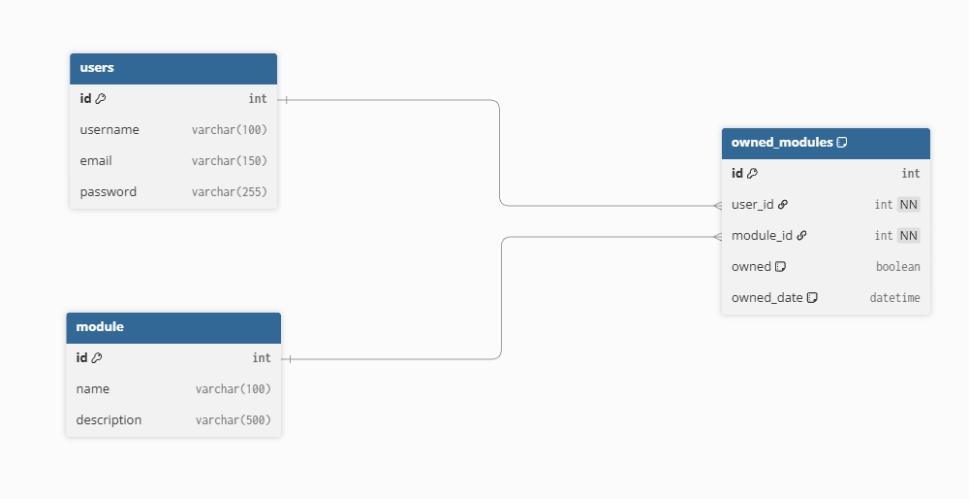

Figura 7 – Modelagem do banco de dados, entidade e relacionamento

De forma geral, a modelagem de dados proposta pretende-se ser adequada aos objetivos do projeto, podendo garantir integridade referencial, compatibilidade com o sistema de autenticação e flexibilidade para futuras evoluções da aplicação.

7 CONSIDERAÇÕES FINAIS

A crescente complexidade dos sistemas digitais e a intensificação dos ataques cibernéticos tornam indispensável a adoção de práticas avançadas de segurança da informação. Nesse contexto, o pentest consolida-se como uma estratégia essencial para identificar vulnerabilidades e fortalecer defesas, desde que conduzido de forma ética, legal e controlada.

A disponibilidade de laboratórios virtuais e plataformas específicas, como ambientes de treinamento e aplicações vulneráveis intencionalmente, possibilita que estudantes, pesquisadores e profissionais desenvolvam suas habilidades sem infringir a legislação vigente. Dessa maneira, a formação prática em cibersegurança não apenas contribui para a capacitação técnica, mas também promove uma cultura de responsabilidade e conformidade legal.

Neste contexto, foram apresentadas a proposta conceitual e o projeto inicial da plataforma OwnedBox, que busca oferecer um ambiente controlado para o aprendizado prático em segurança web. A solução organiza o conteúdo em módulos que simulam vulnerabilidades amplamente recorrentes como SQL Injection, Cross-Site Scripting (XSS) e Upload de Arquivos Maliciosos permitindo que os usuários pratiquem técnicas ofensivas, obtenham tokens de validação como evidência de suas conquistas e, posteriormente, compreendam as contramedidas defensivas correspondentes. Dessa forma, a plataforma OwnedBox configura-se como uma alternativa didática promissora, capaz de apoiar a formação de profissionais mais preparados para lidar com os desafios de um cenário digital em constante transformação.

REFERÊNCIAS

ANDERSON, D. J. **Kanban: Successful Evolutionary Change for Your Technology Business**. Sequim: Blue Hole Press, 2010.

BRASIL. **Lei n.º 12.737, de 30 de novembro de 2012 - Lei Carolina Dieckmann**. 2012. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12737.htm. Acesso em: 08 set. 2025.

BRASIL. **Lei n.º 12.965, de 23 de abril de 2014 - Marco Civil da Internet**. 2014. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l12965.htm. Acesso em: 08 set. 2025.

BRASIL. **Lei n.º 13.709, de 14 de agosto de 2018 - Lei Geral de Proteção de Dados (LGPD)**. 2018. http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 08 set. 2025.

CASTELLS, M. **A Galáxia da Internet: Reflexões sobre a Internet, os Negócios e a Sociedade**. Rio de Janeiro: Jorge Zahar Editor, 2003. Tradução de *La Galaxia Internet* (2001). ISBN 9788571107403.

CLEGGE, D.; MOSS, T. **DSDM: Business Focused Development**. Reading: Addison-Wesley, 2004.

FLANAGAN, D. **JavaScript: The Definitive Guide**. 7. ed. Sebastopol: O'Reilly Media, 2020.

GitHub. **About GitHub**. 2025. Acesso em: 19 out. 2025. Disponível em: https://docs.github.com/en/get-started/using-github/about-github.

GitHub. **About GitHub Projects**. 2025. Acesso em: 19 out. 2025. Disponível em: https://docs.github.com/en/issues/organizing-your-work-with-projects/about-projects.

Hack The Box Ltd. **Hack The Box**. 2025. https://www.hackthebox.com/. [Acesso em: 2025-09-09].

IBM. **O que é um teste de penetração (pentest)?** 2025. Acesso em: 08 set. 2025. Disponível em: https://www.ibm.com/br-pt/think/topics/penetration-testing.

KASPERSKY. **A Brief History of Computer Viruses What the Future Holds**. 2020. Acesso em: 08 set. 2025. Disponível em: https://www.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds.

Laravel. **Blade Templates**. 2025. Acesso em: 19 out. 2025. Disponível em: https://laravel.com/docs/10.x/blade.

MERKEL, D. Docker: Lightweight linux containers for consistent development and deployment. **Linux Journal**, 2014.

OWASP Foundation. **OWASP Top 10: The Ten Most Critical Web Application Security Risks**. 2021. https://owasp.org/Top10/. Acesso em: 11 set. 2025.

RandomStorm. **Damn Vulnerable Web Application (DVWA)**. 2025. https://github.com/digininja/DVWA. [Acesso em: 2025-09-09].

STALLINGS, W. **Cryptography and Network Security: Principles and Practice, 8th edition**. [*S.l.*]: Pearson, 2019. 832 p.

TRYHACKME. **TryHackMe - Plataforma de Aprendizado em Segurança Cibernética**. 2025. Acesso em: 08 set. 2025. Disponível em: https://tryhackme.com/.

União Europeia. **Regulamento (UE) 2016/679 do Parlamento Europeu e do Conselho, de 27 de abril de 2016 - Regulamento Geral de Proteção de Dados (GDPR)**. 2016. https://eur-lex.europa.eu/legal-content/PT/TXT/?uri=CELEX%3A32016R0679. Acesso em: 08 set. 2025.

APÊNDICE A – Funcionalidades Classificadas como Should Have, Could Have e Won't Have (MoSCoW)

Este apêndice apresenta as histórias de usuário classificadas como *Should Have*, *Could Have* e *Won't Have*, que representam funcionalidades desejáveis ou futuras expansões do sistema.

A.1 Should Have (Desejável, mas não obrigatório)

Estes requisitos representam melhorias importantes para a experiência do usuário e a análise de desempenho, porém não são imprescindíveis para a entrega mínima do projeto.

- Relatórios visuais de desempenho com gráficos interativos.
- Estatísticas detalhadas, incluindo tempo gasto, número de tentativas e nível de dificuldade dos módulos.

A.2 Could Have (Poderia ter em versões futuras)

Os itens abaixo são considerados opcionais e podem ser incorporados em versões futuras do sistema, ampliando suas funcionalidades e escopo de aplicação.

- Painel administrativo para gestão de usuários e módulos.
- Dashboard para professores acompanharem o desempenho de turmas.
- Ranking gamificado de usuários com base em tokens e desafios concluídos.

A.3 Won't Have

Estes requisitos foram identificados como fora do escopo atual devido à limitação de tempo e recursos disponíveis, podendo ser considerados apenas em futuras evoluções do projeto.

- Aplicativo mobile nativo.
- Integração com outras plataformas de ensino (por exemplo, Moodle).
- Suporte multilíngue.
- Implementação de todas as vulnerabilidades do OWASP Top 10.