INSTITUTO FEDERAL DO PARANÁ

LEONARDO DA SILVA GOTARDO

MAINDUG: APLICAÇÃO *WEB* DE GERENCIAMENTO DE SENHAS

LONDRINA
2025

0

INSTITUTO FEDERAL DO PARANÁ

LEONARDO DA SILVA GOTARDO

MAINDUG: APLICAÇÃO *WEB* DE GERENCIAMENTO DE SENHAS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal do Paraná – Campus Londrina, como requisito parcial de avaliação.

Orientador: Augusto Luengo Pereira Nunes

LONDRINA
2025

RESUMO

O crescente volume de credenciais exigidas por plataformas digitais e serviços *online* tem imposto aos usuários o desafio de gerenciar, de forma segura e prática, um grande volume de informações de acesso. Essa necessidade torna-se ainda mais relevante diante da crescente preocupação com a privacidade e da dependência de soluções proprietárias, que limitam a autonomia dos indivíduos sobre seus próprios dados. Nesse contexto, foi desenvolvido o MainDug, uma aplicação *web* de código aberto voltada ao gerenciamento de credenciais com foco em transparência, privacidade, flexibilidade e segurança. Diferentemente de soluções tradicionais, o sistema permite que o usuário decida como e onde seus dados serão armazenados, podendo optar por hospedar a aplicação em servidores particulares, integrar a sistemas externos ou modificar os mecanismos de criptografia de acordo com suas necessidades. Entre os principais recursos, destacam-se o armazenamento seguro por meio de algoritmos de criptografia, a possibilidade de personalização da infraestrutura e uma interface intuitiva que favorece a adoção por diferentes perfis de usuários. Dessa forma, o *MainDug* busca oferecer uma alternativa transparente e confiável, reduzindo a dependência de serviços centralizados e promovendo o fortalecimento da autonomia digital e do direito à privacidade.

Palavras-chave: Gerenciador de senhas, privacidade digital, *open-source*, criptografia, segurança da informação.

Abstract

The growing volume of credentials required by digital platforms and online services has imposed on *users* the challenge of securely and practically managing a large volume of access information. This need becomes even more relevant given the growing concern about privacy and the dependence on proprietary solutions, which limit individuals' autonomy over their own data. In this context, MainDug was developed, an open-source *web* application focused on credential management with a focus on transparency, privacy, flexibility and security. Unlike traditional solutions, the system allows *users* to decide how and where their data will be stored, and they can choose to host the application on private servers, integrate it with external systems, or modify encryption mechanisms according to their needs. Key features include secure storage through encryption algorithms the possibility of personalizing the infrastructure and an intuitive interface that favors adoption by different *user* profiles. In this way, *MainDug* seeks to offer a transparent and reliable alternative, reducing dependence on centralized services and promoting the strengthening of digital autonomy and the right to privacy.

Keywords: *Password* manager, digital privacy, open-source, encryption, information security.

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

Desde o surgimento da Internet, na segunda metade do século XX, o mundo passou por uma importante e inesquecível transformação. Inicialmente criada para ser uma rede que interligava computadores em ambientes militares e acadêmicos, a Internet evoluiu rapidamente para se tornar um dos principais pilares da sociedade moderna. Com a criação do *World Wide Web* (W3C) na década de 1990, as possibilidades de uso se expandiram exponencialmente, permitindo a criação de plataformas e serviços que impactaram profundamente tanto a vida pessoal quanto ambientes corporativos dentro de empresas. Entre os avanços mais significativos, destaca-se o desenvolvimento de sistemas que exigem credenciais de acesso, tais como *e-mails*, redes sociais, serviços financeiros e aplicações empresariais. Essas plataformas, ao longo do tempo, não apenas se popularizaram, mas também se tornaram indispensáveis para as operações cotidianas de indivíduos e organizações.

Com o crescimento acelerado de tais ferramentas digitais, o número de senhas que um usuário é obrigado a manter consigo também aumentou drasticamente. Essa realidade, embora reflita o avanço tecnológico, trouxe consigo desafios críticos relacionados à segurança da informação e à usabilidade. Em contextos empresariais, a situação se torna ainda mais sensível, dado o esquecimento ou a perda de senhas por parte dos funcionários, que pode resultar em consequências graves, como a interrupção e perdas de processos e documentos internos importantes, comprometimento de dados sigilosos ou até mesmo a violação de sistemas essenciais para o negócio. Casos de grandes empresas que sofreram prejuízos expressivos devido a credenciais comprometidas ilustram a relevância deste problema, destacando a necessidade de soluções mais eficazes para a gestão de senhas.

1.1 Objetivo

Nesse contexto, o presente trabalho propõe o desenvolvimento do *MainDug*, uma aplicação de código aberto voltada para o gerenciamento seguro de credenciais, com ênfase na privacidade e na autonomia do usuário. Diferentemente de abordagens tradicionais, o projeto oferece maior liberdade de configuração, permitindo que cada indivíduo defina a forma de utilização de acordo com suas vontades e necessidades específicas. Entre as possibilidades, destacam-se a hospedagem em servidores particulares, a configuração de *Proxies* de segurança, a alteração dos mecanismos de criptografia e a integração com serviços complementares.

A proposta visa, portanto, disponibilizar uma alternativa transparente, segura e personalizável, que contribua para a redução da dependência de soluções centralizadas e para o fortalecimento do direito à privacidade digital. Para a efetivação do produto, serão consideradas as seguintes etapas:

- Levantamento e análise dos usuários com foco em privacidade;
- Modelagem do sistema e da aplicação *web*;
- Teste de viabilidade e análise de integração em diferentes cenários de uso.

2 Soluções Correlatas

Com o objetivo de compreender melhor o cenário atual e identificar soluções já consolidadas no mercado, foi realizada uma pesquisa exploratória sobre os *sites* e sistemas especializados em gerenciamento de senhas ativos atualmente. Para isso, utilizou-se a ferramenta de busca *Google*, com as seguintes palavras-chave: " gerenciador de senhas", *password* manager, aplicativos de segurança de credenciais e armazenamento seguro de senhas.

Os principais resultados retornados pela ferramenta foram analisados segundo critérios de proximidade com o tema deste trabalho, funcionalidades disponibilizadas, nível de segurança, modelo de distribuição gratuito (proprietário ou código aberto), experiência do usuário e relevância no mercado. A partir dessa análise, destacaram-se como soluções representativas: *LastPass*, *1Password* e *Bitwarden*.

Para avaliar aspectos técnicos e de usabilidade, os sistemas foram testados considerando métricas de desempenho, acessibilidade e segurança. Além disso, investigaram-se os recursos de personalização e o grau de autonomia oferecido ao usuário, fatores centrais para este estudo.

2.1 *LastPass*

O *LastPass* (LASTPASS:. . . , 2025) é um gerenciadore de senhas de modelo freemium desenvolvido pela LogMeIn, esse é um dos mais conhecidos e amplamente utilizados em escala global. Sua proposta é simplificar o armazenamento de credenciais por meio de sincronização entre múltiplos dispositivos, permitindo ao usuário acessar suas senhas em qualquer lugar. O aplicativo tem diversas ferramentas para gerenciamento e compartilhamento das credenciais. Com ele, pode-se criar senhas fortes, compartilhar credenciais, salvar formas de pagamento como cartões de crédito e débito. O aplicativo também verifica a complexidade das senhas e pode armazenar notas de texto criptografadas, documentos ou outros dados sensíveis. Também possui uma versão *mobile* que sincroniza todas as informações pela conta do usuário. A empresa também é conhecida por operar no modelo *zero-knowledge* (Modelo onde a fornecedora do serviço não tem acesso a informações do usuário), o que significa que nem mesmo os funcionários do *LastPass* podem acessar as informações armazenadas no aplicativo, uma vez que quem criptografa os dados é o próprio dispositivo do usuário, fazendo com que os dados já sejam armazenados criptografados. Para criptografia, o *LastPass* usa, em grande parte, o algoritmo AES-256. Um dos melhores algoritmos para criptografia atualmente. Tornando ataques como os de *brute-force* (força bruta) quase impossíveis.

Pontos positivos:

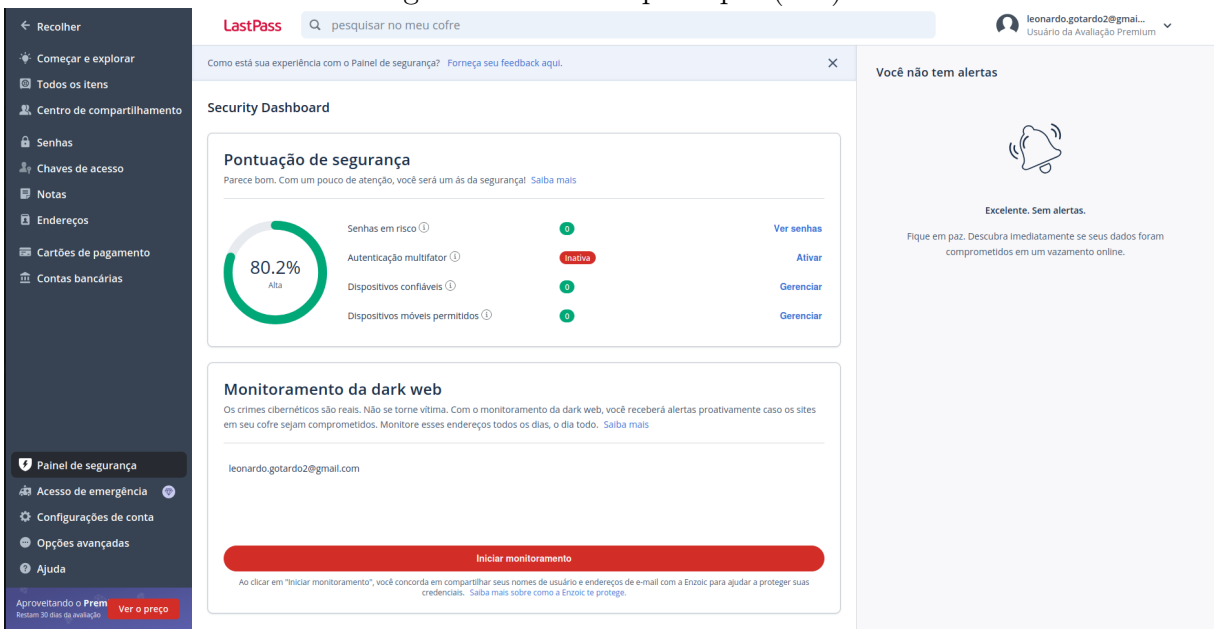
- Interface amigável e de fácil utilização, mesmo para usuários com pouca experiência técnica; (Figura 1)

- Disponibilidade em diversas plataformas, incluindo navegadores, sistemas móveis e *desktops*;
- Recursos adicionais, como geração de senhas fortes e armazenamento seguro de notas.

Pontos negativos:

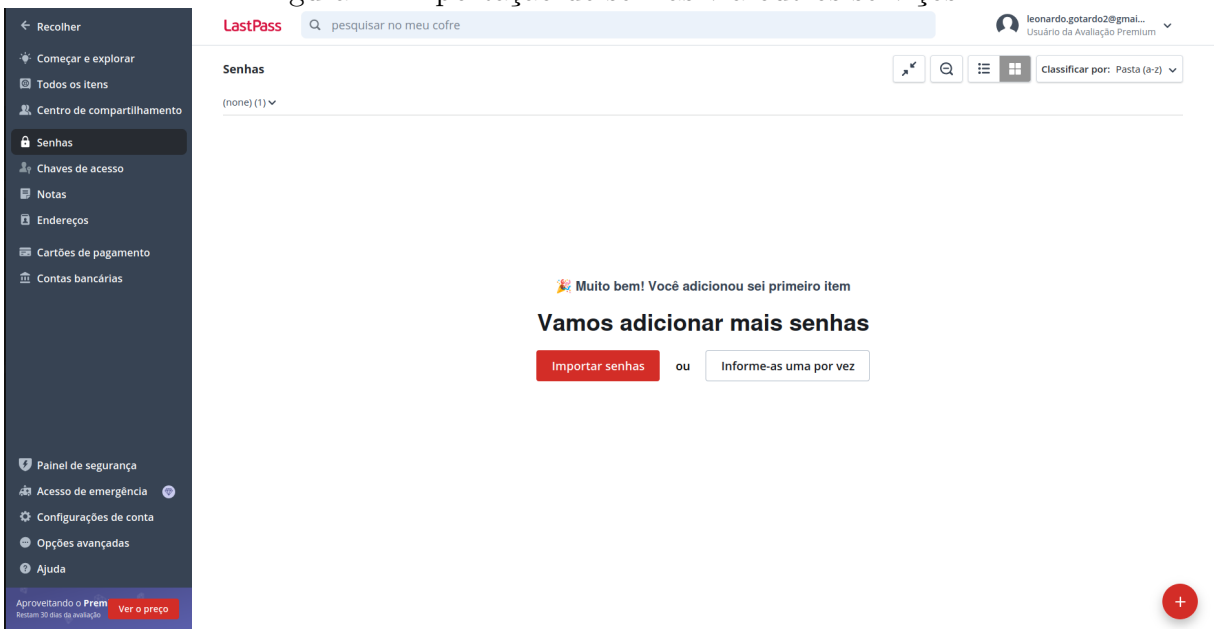
- Modelo de negócio baseado em assinatura, onde a assinatura gratuita comtempla apenas algumas funções basicas, o que pode ser um impeditivo para alguns usuários;
- Histórico de incidentes de segurança com relatos de vazamentos de dados, que comprometem a confiança de parte da comunidade;
- Estrutura fechada (proprietário), não permitindo personalizações ou auditorias independentes do código.

Figura 1: Interface principal (site)



Fonte: lastpass.com, 2025.

Figura 2: Importação de senhas via outros serviços



Fonte: lastpass.com, 2025.

2.2 *1Password*

O *1Password* (1PASSWORD:. . . , 2025) destaca-se pelo foco em usabilidade e experiência do usuário. É frequentemente recomendado em ambientes corporativos devido às suas funções de compartilhamento de credenciais e controle de permissões entre equipes. Assim como outros concorrentes, o *1Password* possui também cofres seguros para armazenamento de formas de pagamento, como cartões de crédito e débito, anotações de texto seguras, anotações também sobre os cartões salvos, como limite ou data de emissão e documentos como CPF ou RG. O *1Password* também possui a criptografia AES-256. Diferente de outros concorrentes, o *1Password* oferece a função de chave-mestra que consiste em uma chave secreta única de 128 *Bits* que é gerada no primeiro uso, fica armazenada no dispositivo e é necessária junto da senha mestra para abertura do aplicativo. Assim como a grande maioria dos seus concorrentes, o *1Password* também possui uma extensão para navegador e opera em modelo de *zero-knowledge*. Ele também conta com o *Whatchtower*, ferramenta responsável por monitorar vazamentos em *sites*, senhas fracas ou comprometidas, senhas repetidas e autenticação de dois fatores. Outra função que diferencia o *1Password* de outros concorrentes é o 'Modo Viagem', que, quando ativo, esconde temporariamente cofres tidos como não seguros em todos os dispositivos até a desativação.

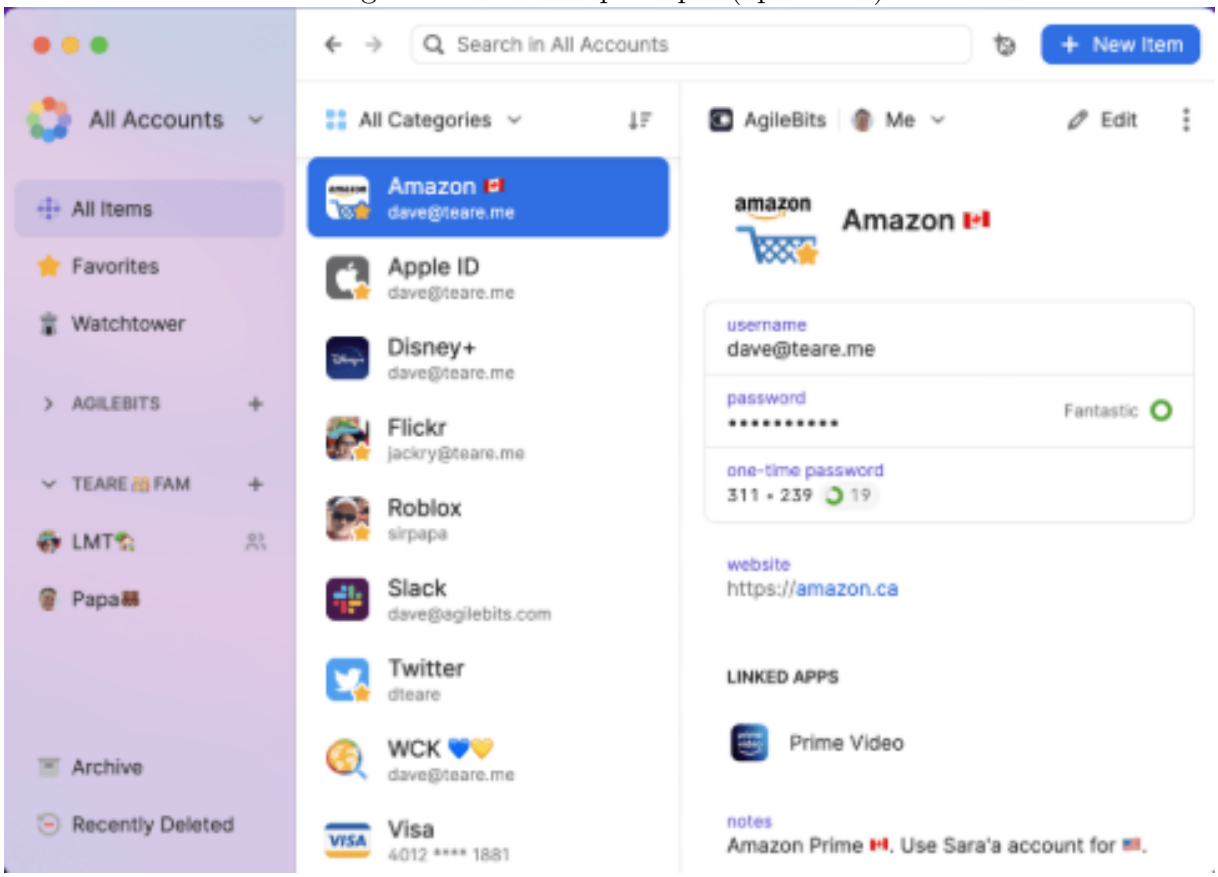
Pontos positivos:

- Experiência de uso intuitiva, com design bem estruturado (Figura 3);
- Funcionalidades voltadas para empresas, como cofre compartilhado e monitoramento de acesso (Figura 4);
- Reputação sólida no mercado em termos de segurança e confiabilidade.
- Observação de vazamentos (Figura 5);

Pontos negativos:

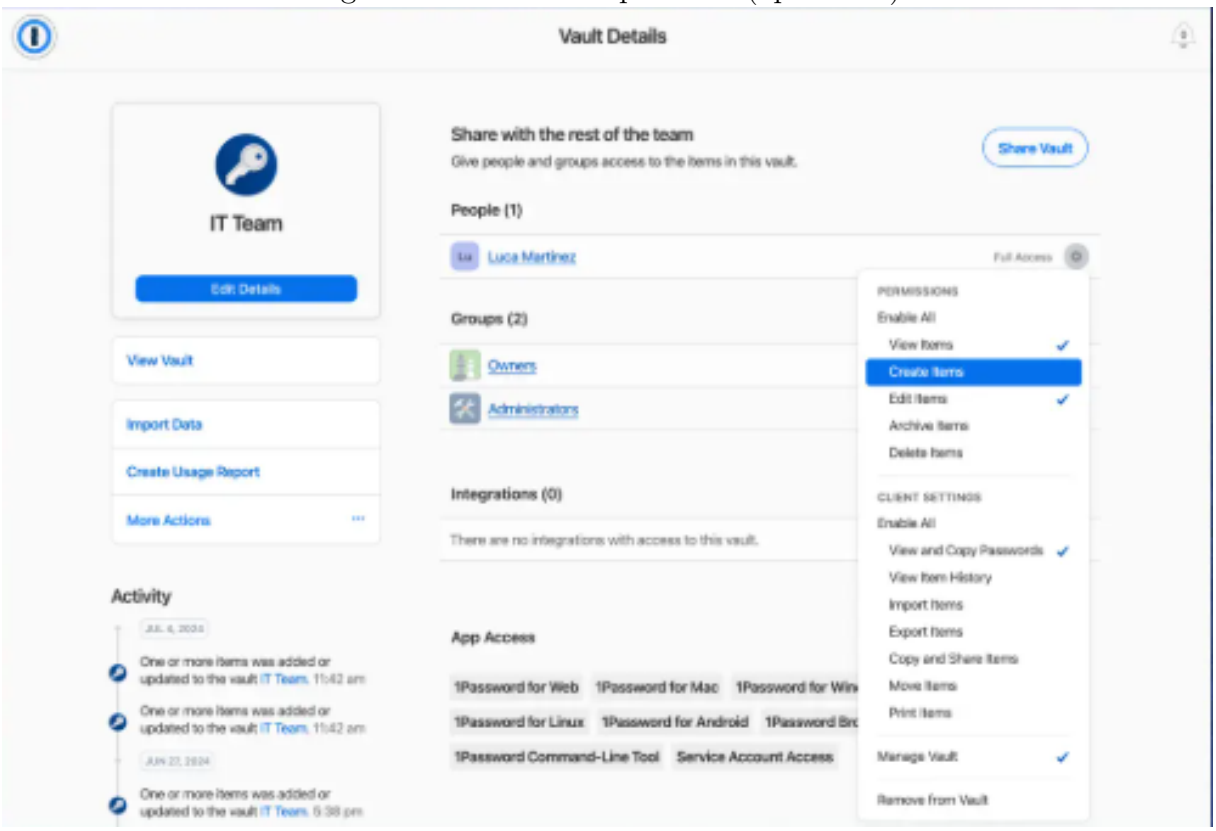
- Sistema proprietário, sem possibilidade de auditoria pública do código fonte;
- Dependência de assinatura paga, não oferecendo uma versão gratuita;
- Menor flexibilidade para usuários que buscam autonomia e personalização do armazenamento de dados.

Figura 3: Interface principal (aplicativo)



Fonte: *1password*.com, 2025

Figura 4: Interface empresarial (aplicativo)



Fonte: *1password*.com, 2025

Figura 5: *Whatchtower* (aplicativo)

Fonte: *1password*.com, 2025

2.3 *Google Senhas*

O *Google Senhas* (GERENCIADOR. . . , 2025) é o gerenciador de senhas integrado ao ecossistema *Google*, disponível nativamente nos dispositivos *Android* e no navegador *Chrome*. Esse é o gerenciador de senhas mais comumente usado. Ele conta com integração completa com os aplicativos do *Google* e *backup*/sincronização automática com todos os dispositivos conectados à mesma conta do *Google*. Assim como outros concorrentes, o *Google* Senhas também usa criptografia de ponta a ponta, políticas de *zero-knowladge* e conta com ferramentas de preenchimento automático de formulários, geração de senhas seguras, alertas de senhas fracas, comprometidas em vazamento de dados ou até senhas reutilizadas em mais de um *site*. Além de ferramentas básicas de gerenciamento de credenciais, o *Google* Senhas conta também com ferramentas para armazenamento seguro de anotações e oferece compartilhamento de credenciais via *Family Link*, possibilitando que membros de um mesmo grupo familiar compartilhem senhas selecionadas de forma segura.

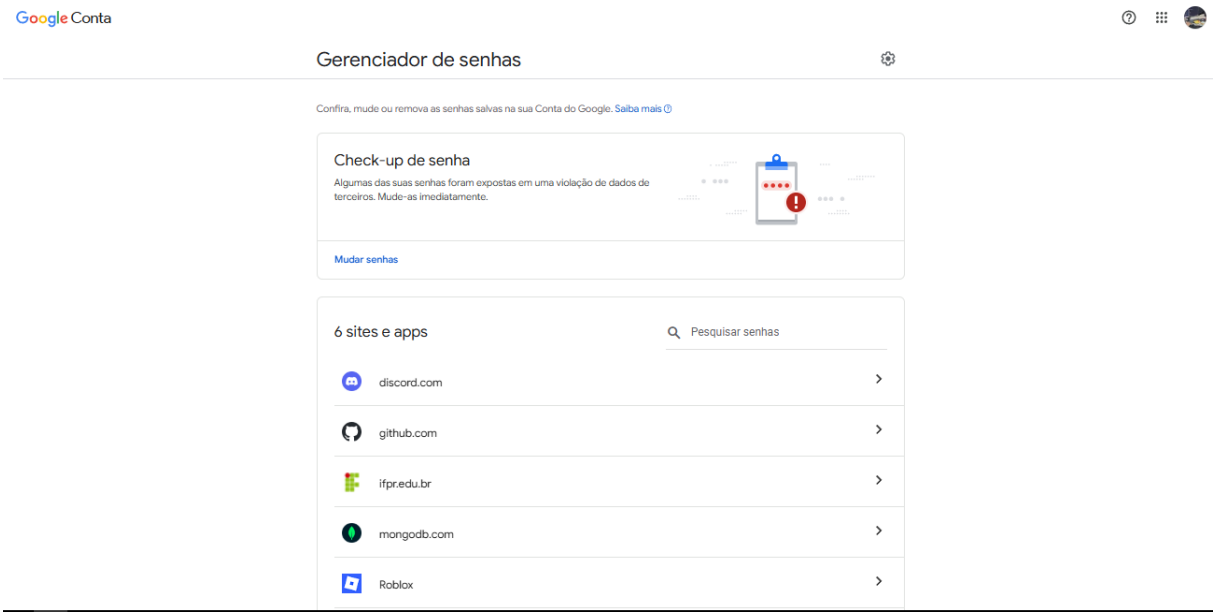
Pontos positivos:

- Vem disponível nativamente nos dispositivos *Android* e navegador *Chrome*;
- Tem compatibilidade com multiplas plataformas como celulares e *notebooks*;
- Sistema integrado de detecção de vazamentos de senhas e *dataleaks* (Figura 7);

Pontos negativos:

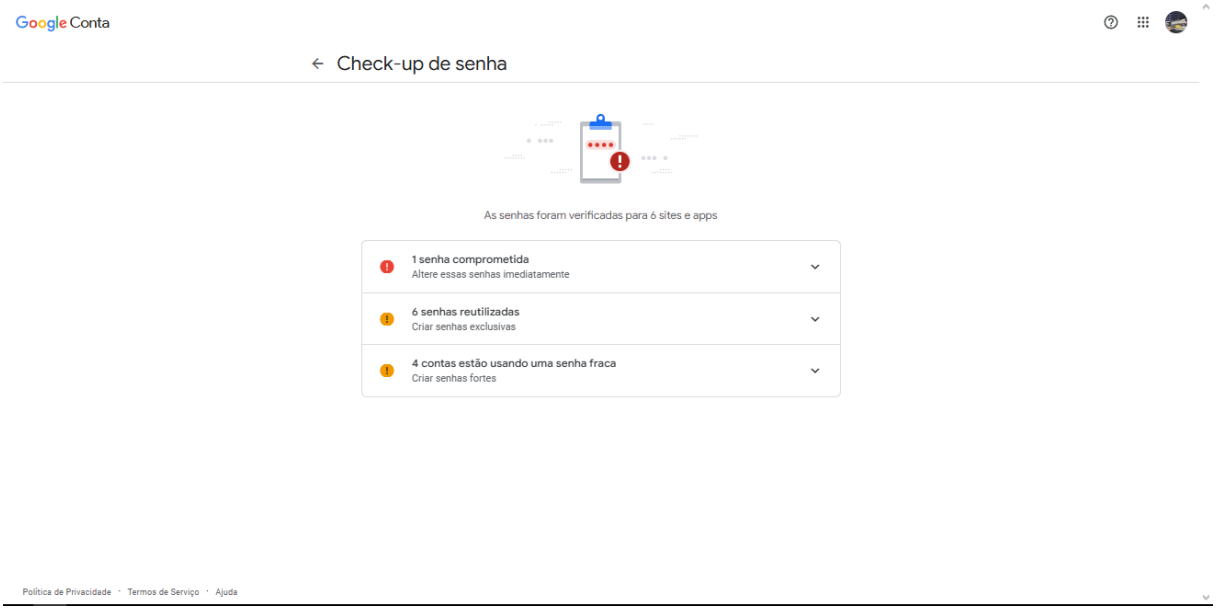
- Grande dependência do ecossistema *Google*;
- Problemas com privacidade, afinal a mesma depende exclusivamente das politicas de privacidade da *Google*;
- Poucas funcionalidades em comparação a outros concorrentes (Figura 6);

Figura 6: Menu principal



Fonte: *passwords.google*.com, 2025

Figura 7: Página de senhas comprometidas



Fonte: *passwords.google*.com, 2025

3 Metodologia

A metodologia adotada para o desenvolvimento do projeto *MainDug* foi estruturada em etapas, combinando uma abordagem de pesquisa exploratória para o levantamento de requisitos com um modelo de desenvolvimento de *software* iterativo e incremental, focado na prototipagem. Esta abordagem permitiu a análise contínua e o refinamento da aplicação ao longo do seu ciclo de vida.

O projeto foi dividido nas seguintes fases principais: Levantamento de Requisitos, Modelagem e Design do Sistema e Implementação.

3.1 Fase 1: Pesquisa e Levantamento de Requisitos

Esta fase inicial teve como objetivo compreender o domínio do problema e o cenário atual dos gerenciadores de senhas. Para isso, foi realizada uma pesquisa exploratória e bibliográfica sobre segurança da informação, privacidade digital e criptografia.

Conforme detalhado na Seção 2 (Soluções Correlatas), foi conduzida uma análise comparativa das principais soluções de mercado (LastPass, *1Password*, *Google* Senhas). Esta análise foi fundamental para:

- Identificar funcionalidades essenciais (ex: geração de senhas, armazenamento seguro, preenchimento automático).
- Compreender os pontos fortes e fracos das soluções existentes (ex: modelos de negócio, histórico de segurança, nível de personalização).
- Definir o diferencial do *MainDug*, com foco em código aberto, autonomia do usuário e privacidade.

Os dados coletados nesta análise serviram como base para a elicitação dos requisitos funcionais (RF) e não funcionais (RNF) do sistema, que estão detalhados na Seção 4 (Resultados).

3.2 Fase 2: Modelagem e Design do Sistema

Após a definição dos requisitos, iniciou-se a fase de modelagem, que traduziu as necessidades do usuário em uma arquitetura técnica. Esta etapa utilizou a *Unified Modeling Language* (UML) para a modelagem orientada a objetos e o Modelo de Entidade-Relacionamento para o banco de dados.

Foram desenvolvidos os seguintes artefatos (apresentados na Seção 4):

- **Diagramas de Caso de Uso:** Para descrever as interações dos atores (Usuário e Administrador) com o sistema.

- **Diagrama de Classes:** Para representar a estrutura estática do sistema, suas classes, atributos e relacionamentos.
- **Diagrama de Entidade-Relacionamento (DER):** Para projetar a estrutura lógica do banco de dados *PostgreSQL*, garantindo a integridade e o relacionamento correto entre as entidades (User, *Password*, Filter etc.).

Paralelamente, foi realizado o design da interface do usuário (UI) e da experiência do usuário (UX), resultando nos protótipos de tela que guiaram a implementação do *webservice*.

4 Resultados

4.1 Requisitos Funcionais (RF)

Tabela 1: Lista de Requisitos funcionais

ID	**Descrição**	**Prioridade**
RF-001	*Login*	Alta
RF-002	Cadastro	Alta
RF-003	Geração de senha segura	Alta
RF-004	Monitoramento de uso do *autocomplete* pela extensão	Média
RF-005	Monitoramento de vazamento de credenciais	Média
RF-006	Contagem de senhas reutilizadas	Baixa
RF-007	Edição de credenciais de acesso ao *webservice*	Média
RF-008	Verificação da força/complexidade das senhas cadastradas	Média
RF-09	Busca de credencial via filtro, como *site/login*	Alta
RF-010	Recuperação de senha via *e-mail*	Média
RF-011	Categorizar senhas em pastas ou filtros	Baixa
RF-012	Gerenciar credenciais	Alta

Tabela 2: Lista de Requisitos não funcionais

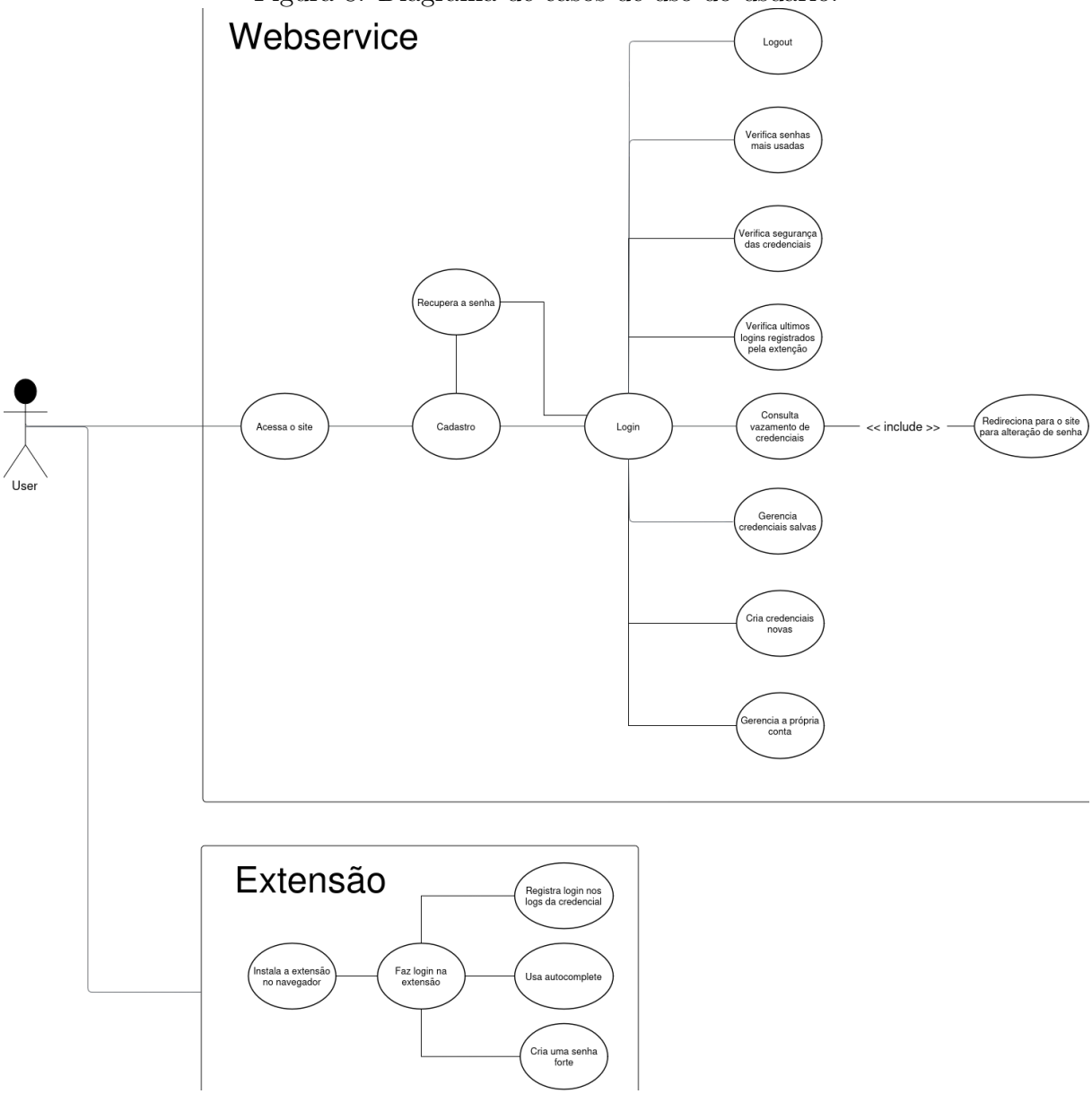
ID	**Descrição**	**Prioridade**
RNF-001	Criptografia *one-way* para as senhas	Alta
RNF-002	Criptografia *both-ways* para credenciais	Alta
RNF-003	Responsividade do *webservice*	Média
RNF-004	Sanitização de dados recebidos pela API	Alta
RNF-005	Disponibilidade mínima de 99% do sistema	Média
RNF-006	Tempo de resposta '¡' 500ms	Media
RNF-007	Compatibilidade com os principais navegadores (*Chrome*, *Firefox* e *Edge*)	Alta
RNF-008	Persistência dos cadastros salvos	Alta

4.2 Requisitos não Funcionais (RNF)

4.3 Diagrama de casos de uso

4.3.1 Usuário

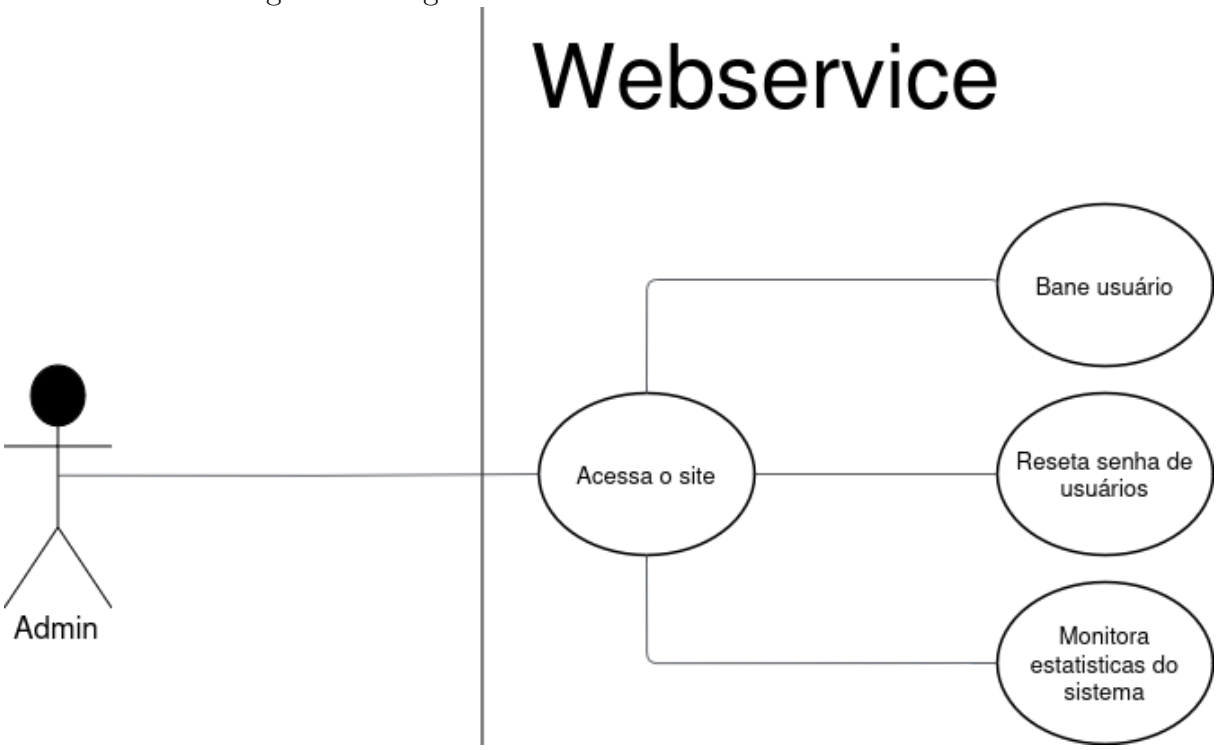
Figura 8: Diagrama de casos de uso do usuário.



O usuário tem acesso ao *Webservice* e à extensão do navegador. Na extensão, o usuário pode usar o *autocomplete*, cadastrar uma nova credencial e, independentemente do usuário, a ação de *login* é salva e enviada ao *Webservice*. Já no *Webservice*, o usuário pode consultar quais senhas são repetidas, quais foram vazadas, ser redirecionado ao *site* da senha repetida ou vazada, verificar *logs* ou alterações recentes nas credenciais, gerenciar sua conta ou recuperar a senha.

4.3.2 Administrador

Figura 9: Diagrama de casos de uso do administrador.



O administrador possui acesso apenas ao *Webservice*; lá, ele pode gerenciar usuários, banindo-os, enviar *e-mail* de recuperação de senha e monitorar as estatísticas dos usuários.

4.4 Estratégias de segurança

4.5 Mimissismo

da *database*: O mimissismo da *database* faz com que seja aplicada alguma criptografia ou ofuscação não só nos dados armazenados, mas também nos nomes dos atributos e tabelas. Isso faz com que mesmo com acesso ao banco de dados, o invasor não consiga determinar quais dados estão armazenados.

4.6 Criptografia da Senha-Mestra (*one-way*):

A senha-mestra do usuário, que dá acesso ao cofre, é protegida usando um algoritmo de *hash* adaptativo (como o *bcrypt* ou *Argon2*). Esta abordagem, recomendada pelo OWASP (PASSWORD. . . , 2024), torna a senha ilegível e computacionalmente inviável de ser revertida, protegendo-a contra ataques de força bruta mesmo em caso de vazamento do banco de dados.

4.7 Criptografia do Cofre (*both-ways*):

Para as credencias de *login* dentro do aplicativo como senha-mestra são protegidas por criptografia sem recuperação (como *bcrypt*). Seguindo as diretrizes do OWASP (CRYPTOGRAPHIC..., 2023) para armazenamento criptográfico, a chave de decodificação é derivada da senha-mestra do usuário. Isso garante que os dados no banco de dados permaneçam indecifráveis para qualquer um que não possua a senha-mestra, implementando um modelo *zero-knowledge*.

4.8 Tecnologias

Nesta sessão serão descritas as tecnologias escolhidas para o desenvolvimento do *MainDug* e a razão para essas escolhas.

4.8.1 *Python*

A escolha do *Python* como linguagem de programação principal para o back-end do *MainDug* se deve a múltiplos fatores. Primeiramente, sua sintaxe clara e legível (conhecida como " Pythonic") facilita a manutenção do código e, crucialmente para um projeto de código aberto, permite que a comunidade realize auditorias de segurança de forma mais eficaz.

Além disso, o *Python* possui um ecossistema robusto e maduro, com vastas bibliotecas de terceiros, especialmente no que tange à segurança e à criptografia (como *bcrypt*). Isso permitiu a implementação de algoritmos de criptografia fortes (RNF-001, RNF-002) sem a necessidade de reinventar soluções de segurança, garantindo o uso de padrões já testados e validados pela indústria.

4.8.2 *Flask*

Flask (FLASK..., 2025) foi selecionado como o *micro-framework web* para a construção da API e do *webservice* do *MainDug*. Diferente de frameworks monolíticos, o Flask adota uma abordagem minimalista e flexível, fornecendo as ferramentas essenciais para roteamento e gerenciamento de requisições sem impor uma estrutura rígida de projeto.

Essa flexibilidade foi um requisito fundamental para o *MainDug*, alinhando-se ao objetivo de autonomia e personalização. Ele permite um controle granular sobre os componentes da aplicação, facilitando a integração de bibliotecas específicas, como o *SQLAlchemy*, e a implementação de mecanismos de segurança personalizados (RNF-004), sendo ideal para construir uma API leve, de alto desempenho e focada em segurança.

4.8.3 *SQLAlchemy*

SQLAlchemy (SQLALCHEMY..., 2025) foi adotado como o Mapeador Objeto-Relacional (ORM) e *toolkit* SQL. A principal função do *SQLAlchemy* neste projeto é abstrair a comunicação com o banco de dados, permitindo que a lógica de negócios seja escrita em classes Python (como as classes *User* e *Password* no Diagrama de Classes) em vez de consultas SQL manuais.

A principal vantagem de segurança ao usar um ORM como o *SQLAlchemy* é a prevenção nativa contra ataques de Injeção de SQL (SQL Injection), que é consistentemente classificada como uma das vulnerabilidades mais críticas pela OWASP (OWASP..., 2021), pois ele parametriza automaticamente todas as consultas. Além disso, o *SQLAlchemy* é agnóstico em relação ao SGBD, o que significa que, embora o *PostgreSQL* tenha sido escolhido para este projeto, a aplicação pode ser facilmente adaptada por um usuário para rodar em outros bancos, como *MySQL* ou *SQLite*, reforçando o pilar da flexibilidade.

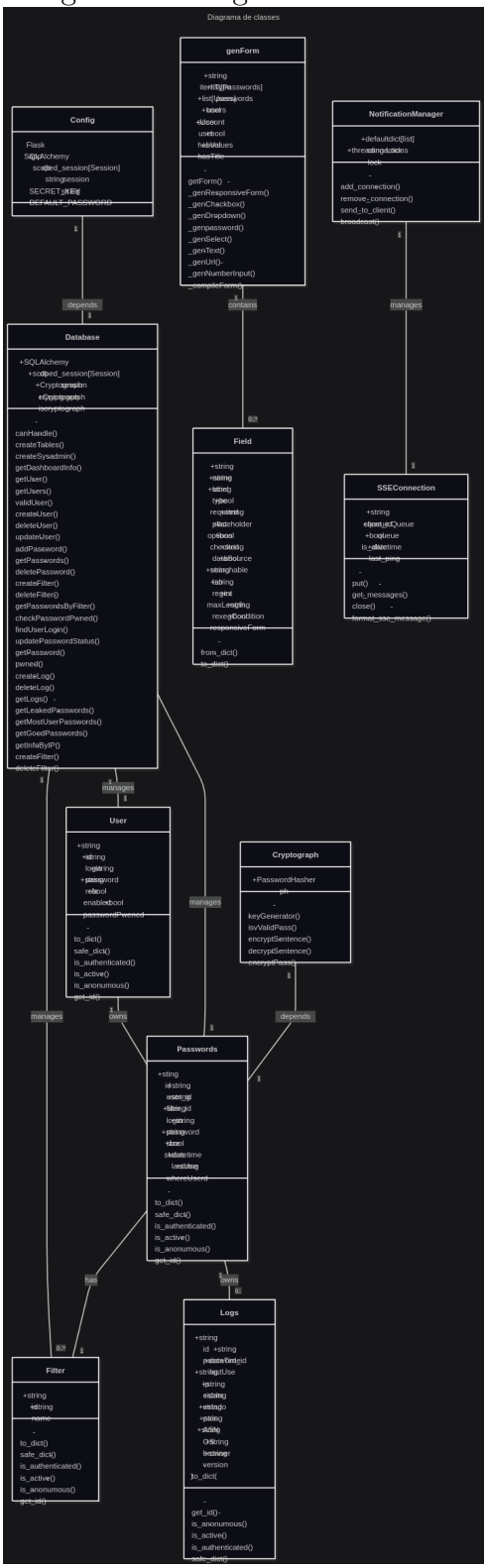
4.8.4 *PostgreSQL*

Para o Sistema de Gerenciamento de Banco de Dados (SGBD), optou-se pelo *PostgreSQL*. Esta escolha é justificada por ser um dos sistemas de banco de dados relacional de código aberto mais avançados e confiáveis do mundo, alinhando-se perfeitamente com a filosofia *open-source* do *MainDug*.

O *PostgreSQL* é renomado por sua robustez, integridade de dados e conformidade estrita com os padrões SQL. Para uma aplicação que gerencia dados sensíveis como credenciais, sua arquitetura madura e seus recursos avançados de segurança (como controle de acesso granular e extensibilidade) fornecem uma fundação sólida e confiável para o armazenamento persistente (RNF-005) e seguro dos dados criptografados dos usuários (POSTGRESQL:..., 2025).

4.9 Diagrama de classes

Figura 10: Diagrama de classes.



4.9.1 Classe *User*

Modelo da tabela de usuários para gerenciamento dentro do código. A classe também conta com um método extra para exportar todas as informações de um usuário via dicionário e outro método para exportar apenas dados não sensíveis. E age como uma referência da entidade que está dentro do banco de dados para orientação dentro do código. Também é responsável por definir os parâmetros e regras da tabela dentro do banco de dados.

4.9.2 Classe *Password*

Modelo da tabela de credenciais salvas para gerenciamento dentro do código. Dentro da classe também existe um método para extração dos dados completos do registro. Assim como a tabela de *User*, essa tabela age como um modelo de objeto para gerenciamento das entidades e definições de regras dentro do banco de dados.

4.9.3 Classe *Filter*

Modelo da tabela de filtros para gerenciamento dentro do código. Possui também um método para exportar as informações como dicionário. Também funciona como modelo de regras e gerenciamento para as entidades da tabela *Filter* dentro da *database*.

4.9.4 Classe *Database*

Classe principal de gerenciamento da *database*. Nela estão concentrados todos os métodos de gerenciamento de dados dentro do aplicativo. Também a partir dela é feita toda a recuperação de dados ao *front-end*.

4.9.5 Classe *Config*

Classe usada para definir variáveis globais e configurações do Flask e *SQLAlchemy*. Dentro dessa classe se encontram algumas das variáveis mais importantes do aplicativo, como variáveis de ambiente, sessão principal da *database*, nome do aplicativo ou a URL do banco de dados. Variáveis utilizadas dentro de todo o código.

4.9.6 Classe *Field*

Dataclass responsável por padronizar formulários recebidos via JSON. É a partir dela que a estrutura básica dos formulários é criada e os parâmetros sanitizados.

4.9.7 Classe *GenForm*

Classe com todos os templates de componentes HTML destinados à criação de formulários. Essa classe recebe os dados da classe *Field* e converte dicionários em for-

mulários prontos para serem exibidos ao usuário com o objetivo de facilitar o processo e padronizar a criação de formulários.

4.9.8 Classe *NotificationManager*

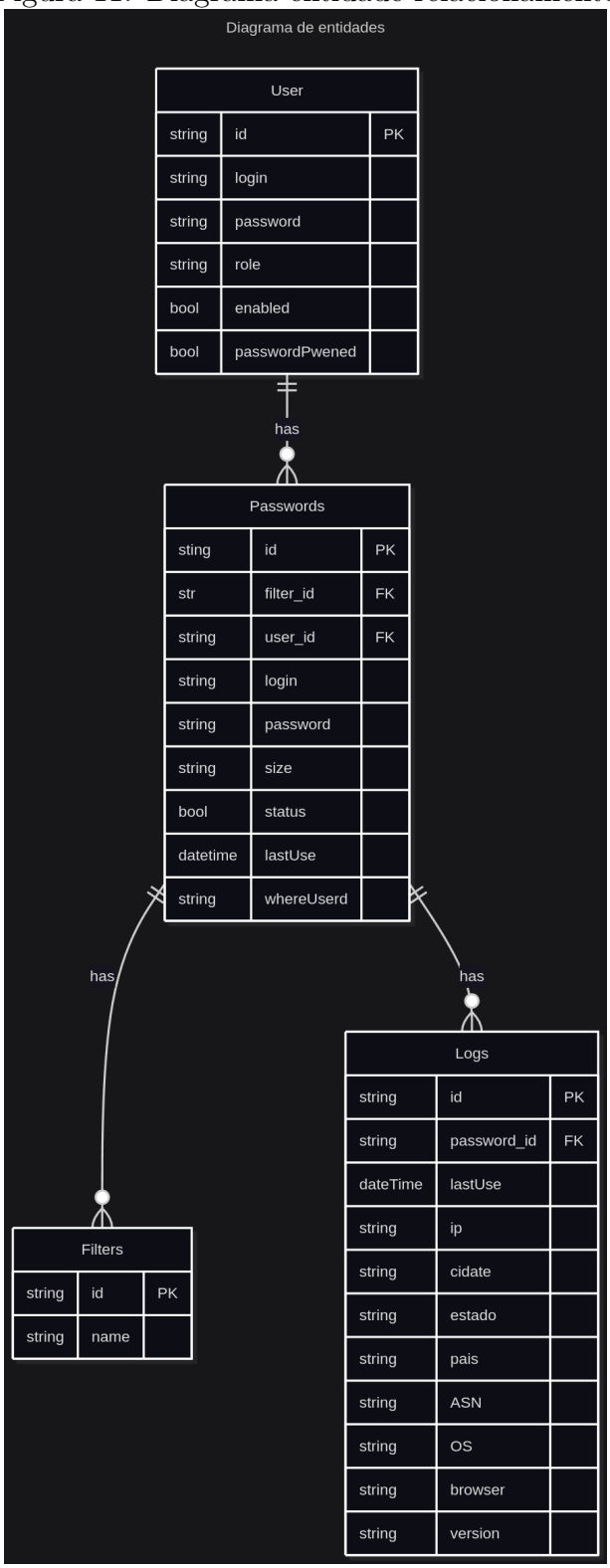
Classe responsável por gerenciar mensagens e notificações exibidas aos usuários dentro do *webservice*. Essa é a classe principal do sistema de notificação. Nela são armazenadas as listas de clientes e os estados de suas conexões. Por ela são enviadas as notificações para os clientes, de forma privada ou em *broadcast*.

4.9.9 Classe *SSEConnection*

Classe destinada ao gerenciamento das conexões entre o servidor e os usuários para envio de alertas e notificações dentro do *webservice*. A classe controla a conexão individual de cada cliente. Essa também monitora as mensagens pendentes para envio e o estado de cada mensagem já enviada.

4.10 Diagrama entidade-relacionamento (DER)

Figura 11: Diagrama entidade-relacionamento.



4.10.1 *User*

A tabela *User* gerencia as contas de acesso ao próprio sistema gerenciador. Ela armazena as credenciais de *login* (*login*, *password*) do usuário, seu nível de permissão (*role*) e se a conta está ativa (textitenabled) ou se sua senha foi comprometida (*passwordPwened*).

4.10.2 *Password*

Esta é a tabela principal do " cofre", onde são armazenadas as credenciais salvas pelo usuário. Cada registro representa um *login*/senha de um serviço externo (*login*, *password*, *whereUserd*), vinculando-se ao *User* (o dono) e a uma categoria.

4.10.3 *Filters*

Filters é uma tabela auxiliar simples usada para organizar as senhas. Ela armazena os nomes das categorias (como " Trabalho", " Pessoal"e " Redes Sociais") que são associadas aos registros na tabela *Passwords* para facilitar a busca e a organização.

4.10.4 *Logs*

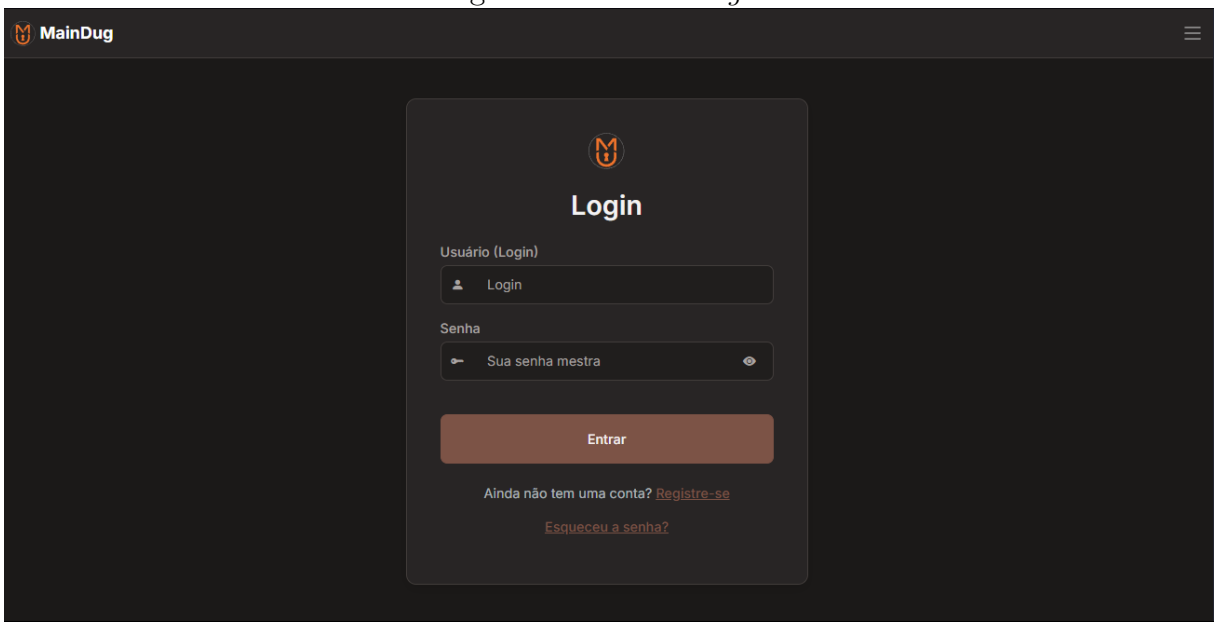
A tabela *Logs* é crucial para a auditoria de segurança. Ela registra um histórico detalhado de cada vez que uma credencial da tabela *Passwords* é utilizada, armazenando informações vitais como o IP, a geolocalização (cidade, país), o sistema operacional e o navegador de quem realizou o acesso.

5 Protótipos de tela

5.1 *Webservice*

5.1.1 Tela de *login*

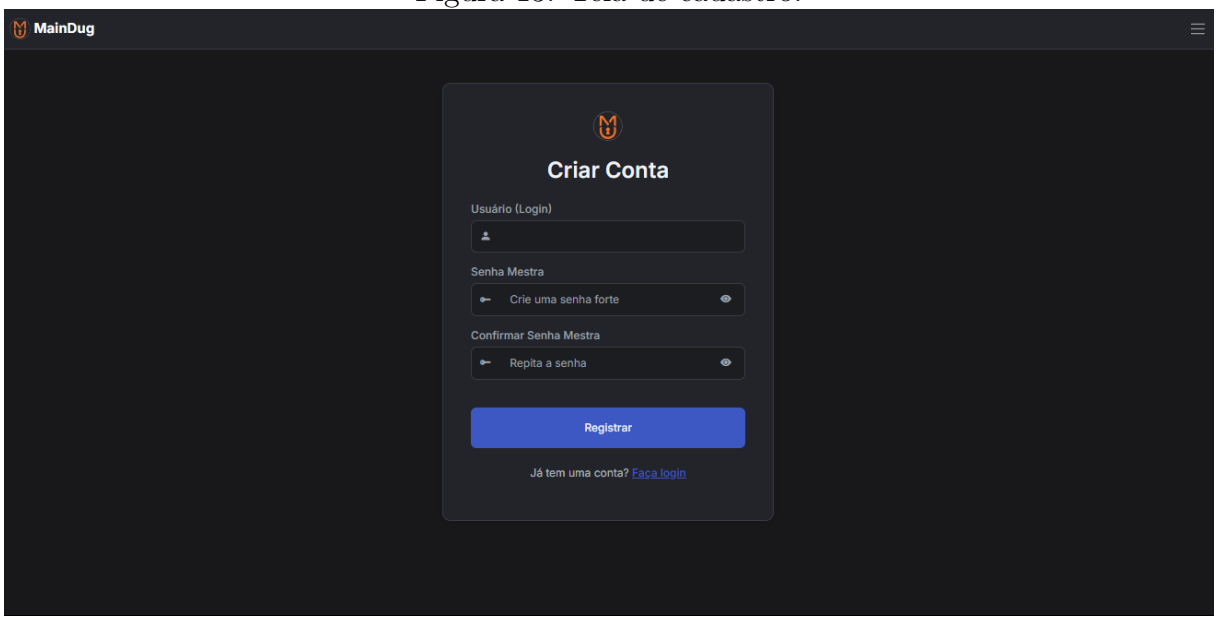
Figura 12: Tela de *login*.



Esta é responsável por fornecer o *login* ao usuário (RF-001). Ela contêm um campo para *login*/nome de usuário e outro para sua senha mestra. Também possui um botão destinado ao envio do formulário, *links* para a página de cadastro e para a página de recuperação de senha.

5.1.2 Tela de Cadastro

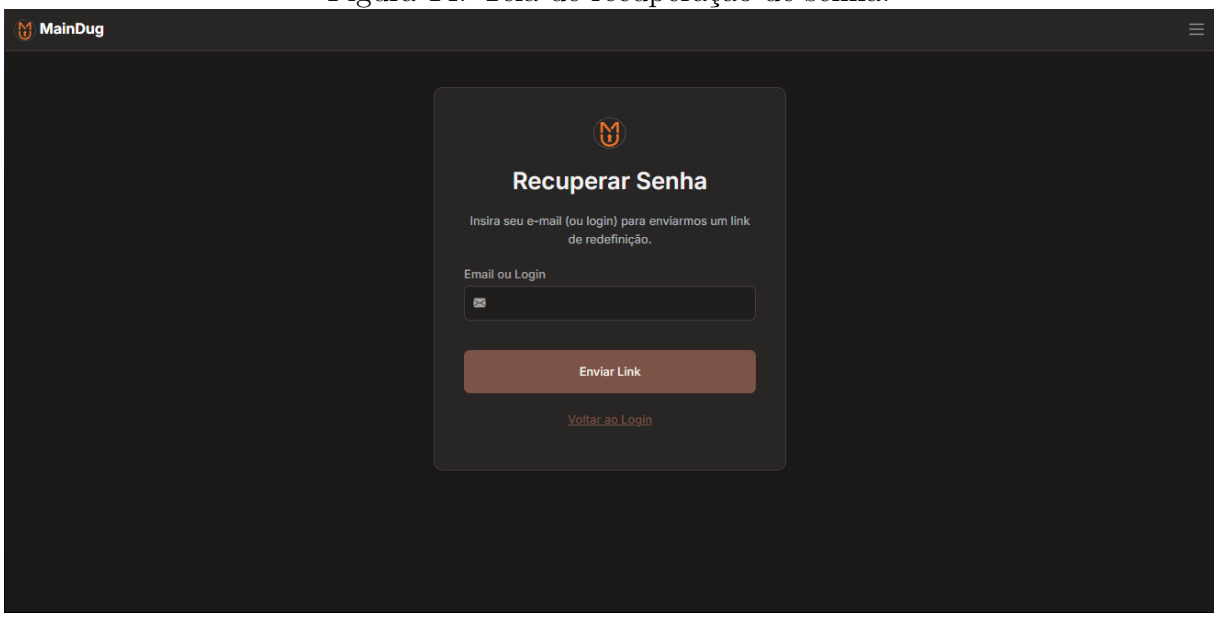
Figura 13: Tela de cadastro.



Responsável por permitir que o usuário crie um cadastro no aplicativo (RF-002). Ela possui três campos, respectivamente para *login*/*e-mail*, senha-mestra e confirmação da senha-mestra. Possui também um botão para envio de formulário e *links* para *login* e recuperação de senha.

5.1.3 Tela de recuperação de senha.

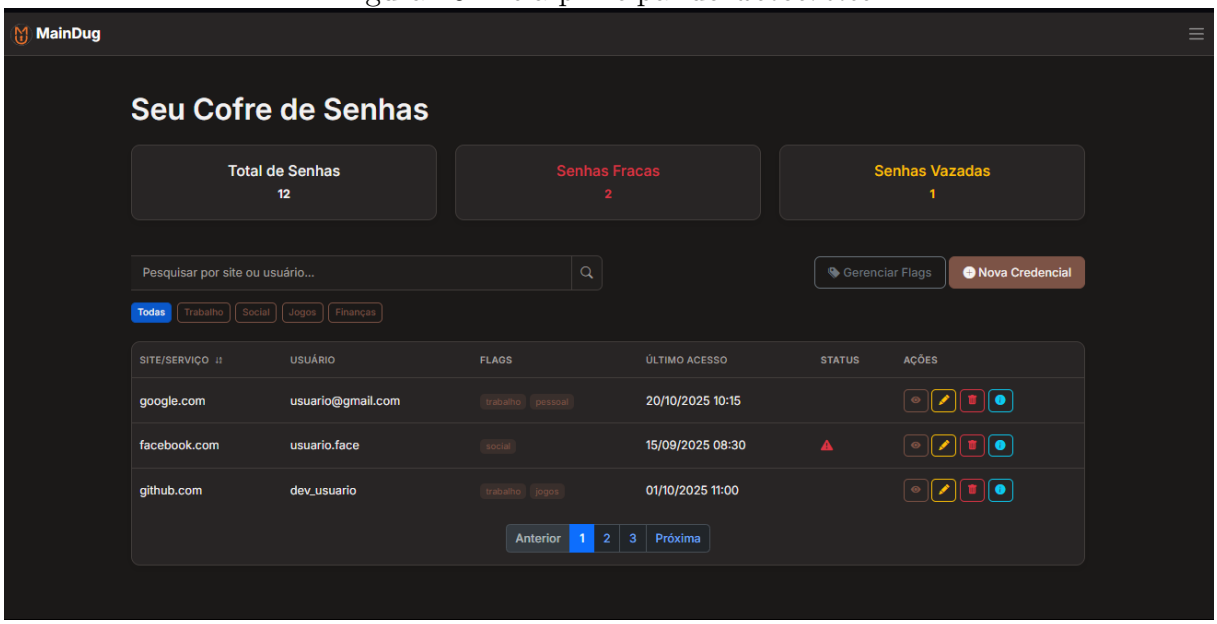
Figura 14: Tela de recuperação de senha.



Dentro desta página há apenas um campo para o usuário inserir o *login*/*e-mail*, um botão para enviar o formulário e *links* para as páginas de *login* e cadastro. Sua função é permitir que o usuario recupere o acesso a conta a partir do *e-mail*/*login* (RF-010).

5.1.4 Tela do *Dashboard*.

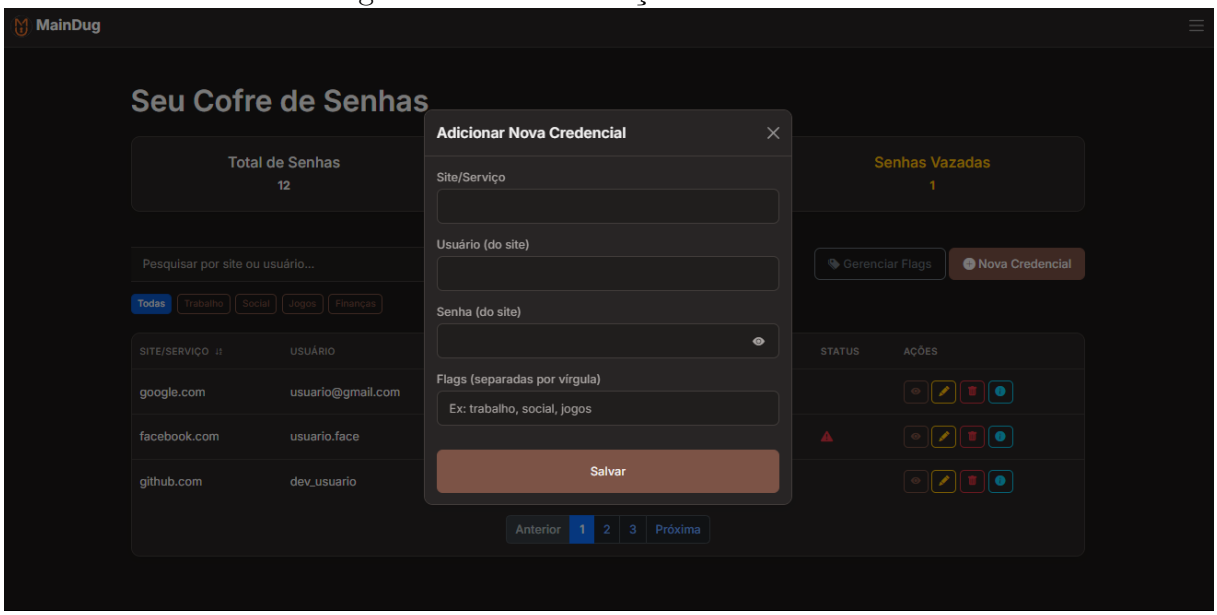
Figura 15: Tela principal do *webservice*.



Destinada a mostrar as informações principais do aplicativo ao usuário. Nela, encontra-se uma tabela com as credenciais salvas, opções para filtrar as credenciais na tabela (RF-009) e botões para editar, excluir (RF-012) e monitorar as credenciais salvas. Também há uma *navbar* com *links* para a página de edição da conta, saída da conta, mudança de tema entre claro/escuro e a página de estatísticas.

5.1.5 Tela de adição de credencial.

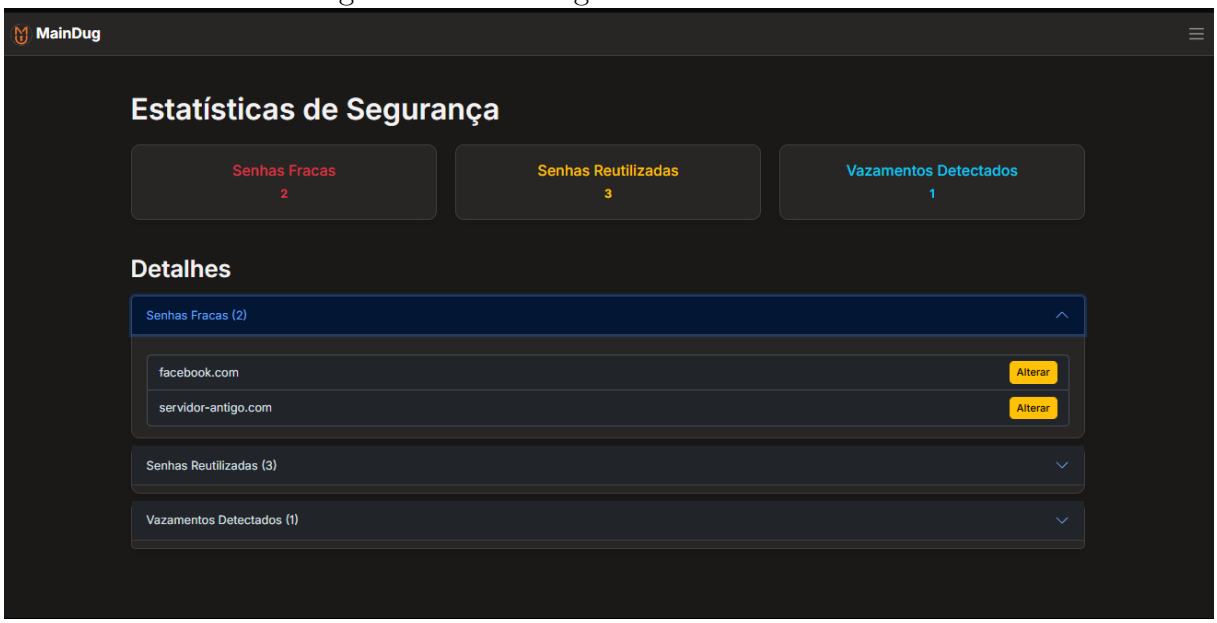
Figura 16: Tela de adição de credenciais.



Esta página aparece como um *popup* e é responsável por permitir a adição de credenciais dentro do sistema via *webservice* (RF-012). No formulário do *popup* existem quatro campos para o usuário preencher que são destinados, respectivamente, ao *login* da credencial dentro do *site* de terceiros, à senha para essa credencial, ao endereço *web* desse *site* e às *flags* ou filtros para esse cadastro e, por fim a página também possui um botão para salvar a credencial.

5.1.6 Tela de gerenciamento de filtros.

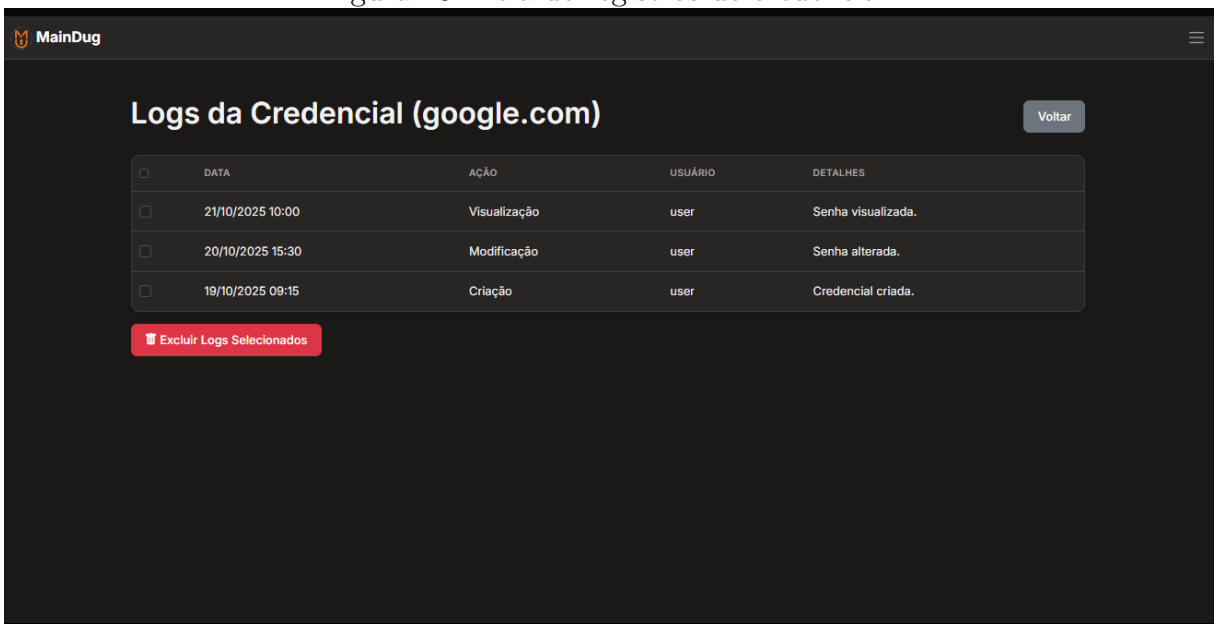
Figura 17: Tela de gerenciamento de filtros.



Página responsável por criar e deletar filtros para as credenciais. É composta por uma lista com todos os filtros, um botão para apagar cada um dos filtros e um botão para sair. As mudanças são salvas automaticamente no *back-end*.

5.1.7 Tela de registros de credencial.

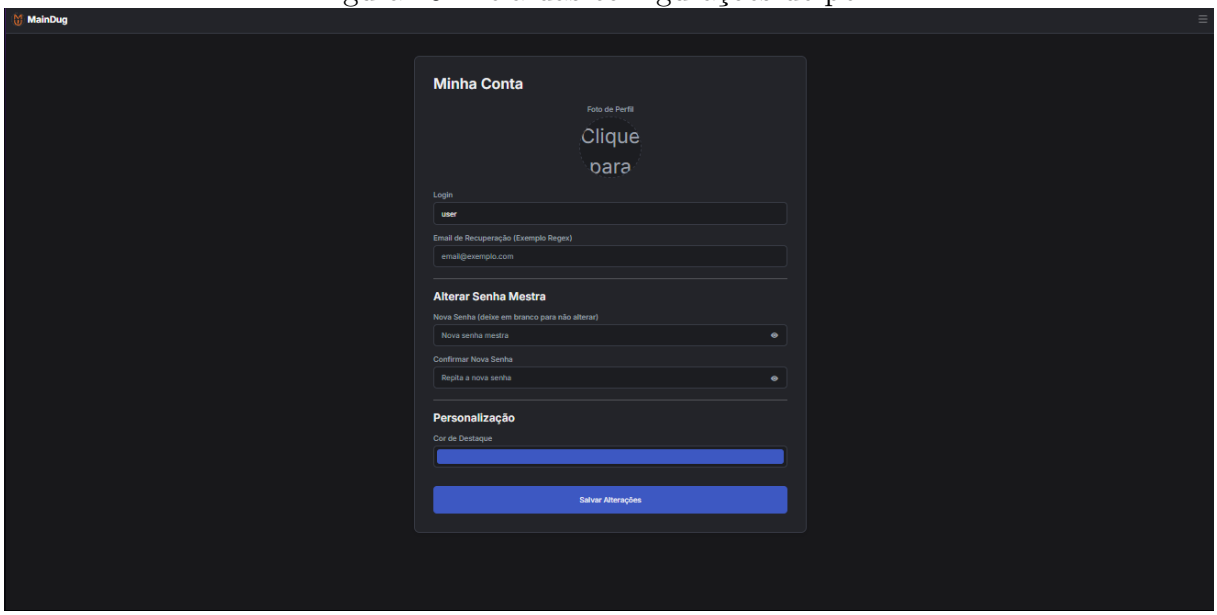
Figura 18: Tela de registros de credencial.



Página responsável por mostrar ao usuário todos os registros relacionados à essa credencial, como uso do *autocomplete*, vazamentos detectados e edição da credencial.

5.1.8 Tela de perfil.

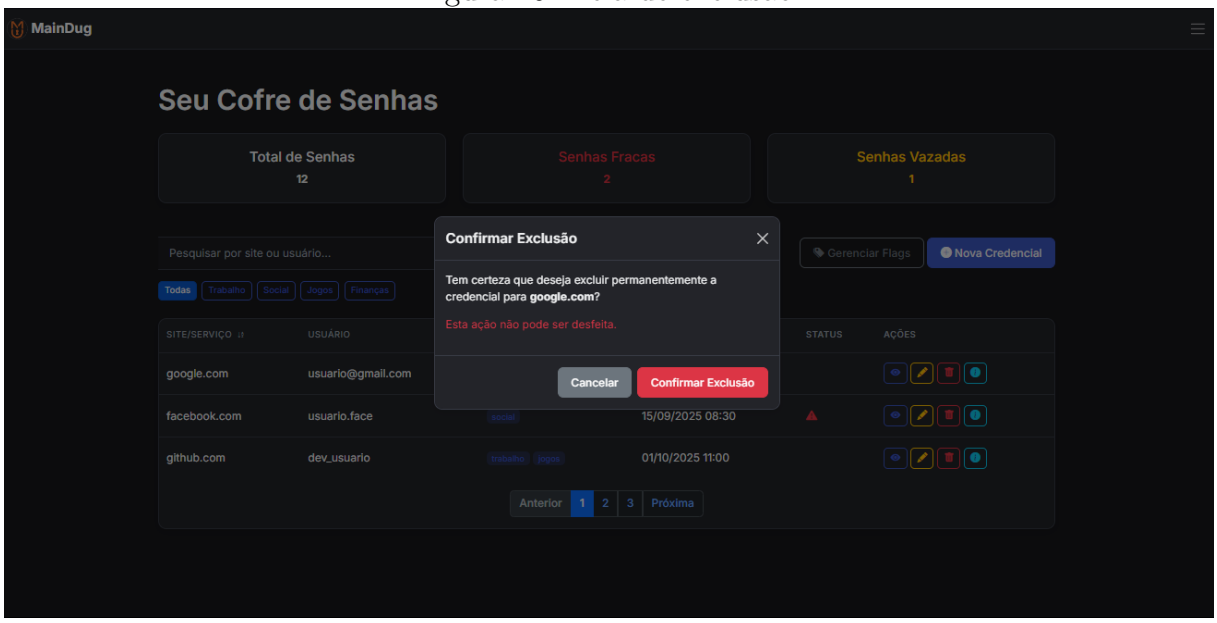
Figura 19: Tela das configurações de perfil.



Página destinada a todo o gerenciamento do perfil e conta do usuário (RF-007). Esta possui cinco campos para preenchimento, quatro deles destinados aos dados do usuário, respectivamente, para o *login*, senha, confirmação de senha e *e-mail* para recuperação da senha-mestra, já o último pode ser usado para personalizar a cor secundária do *site* e, assim como os outros formulários, esta possui um botão para salvar as alterações.

5.1.9 Tela de exclusão de credencial.

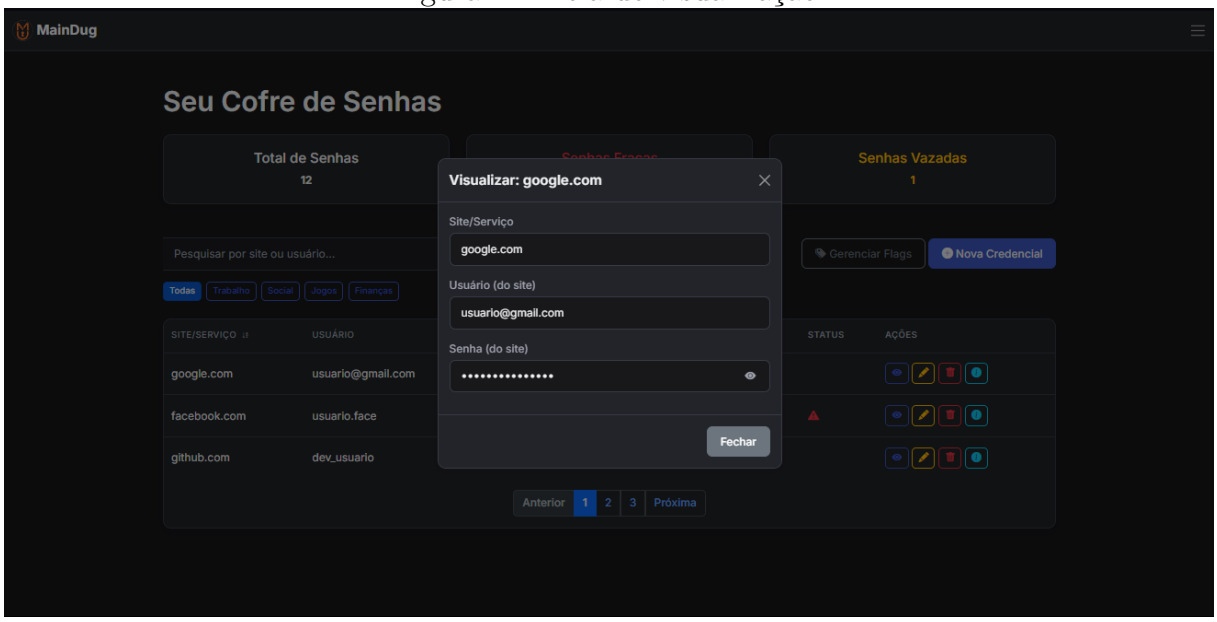
Figura 20: Tela de exclusão.



Página dedicada a confirmação de exclusão de uma credencial (RF-007). Ela aparece como *popup* e é composta por um botão para confirmar a exclusão e um botão para fechar a *popup*.

5.1.10 Tela de visualização das credencial.

Figura 21: Tela de visualização.



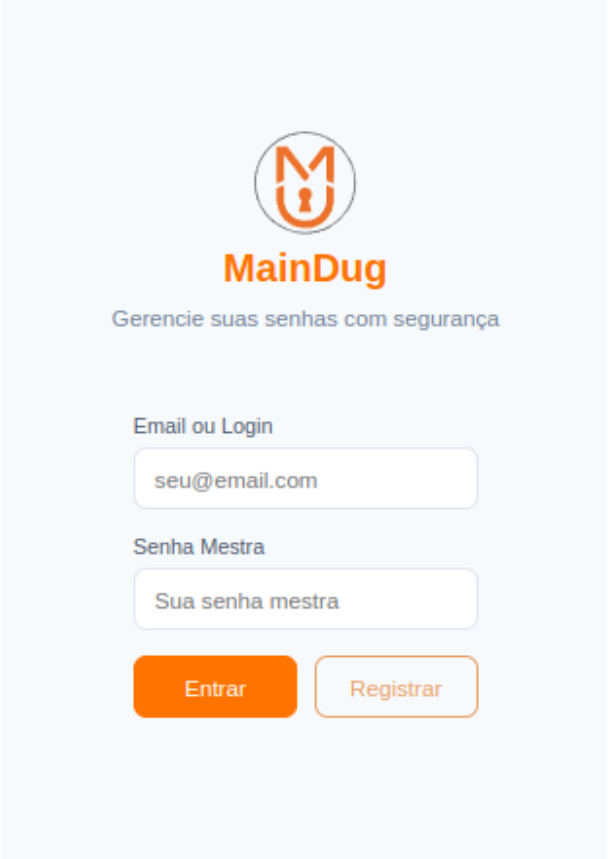
Página destinada a visualização de uma credencial. Ela aparece como *popup* e é composta por três campos, sendo eles respectivamente o endereço do *site*, o *login* e a senha. Nela também ha um botão para cancelar a ação.

5.2 Extensão

Para interagir com o navegador do usuário, foi desenvolvida uma extensão *web* compatível com a API de extensões do *Google Chrome* (CHROME. . . , 2025).

5.2.1 *Popup* de *login*

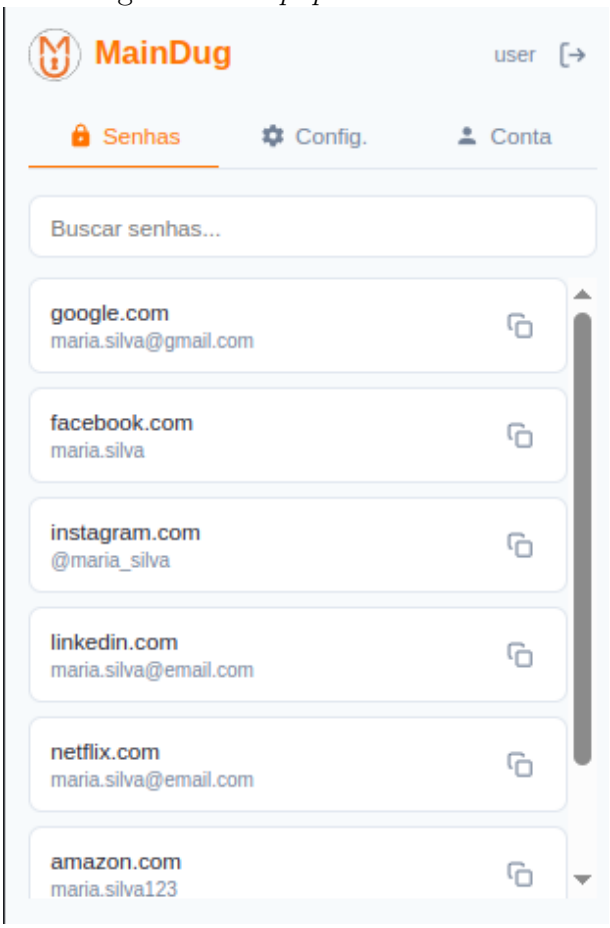
Figura 22: *Popup* de *login*.



A Tela da extensão *web* destinada ao *login* do usuário (Figura 22) é a primeira página que aparece quando o usuario instala a extensão, ela possui dois campos, um para *login* e outro para senha, nela também existem dois botões, um para enviar o formulário e outro para se registrar no serviço, que redireciona o usuário ao *site* oficial.

5.2.2 *Popup* da tela inicial

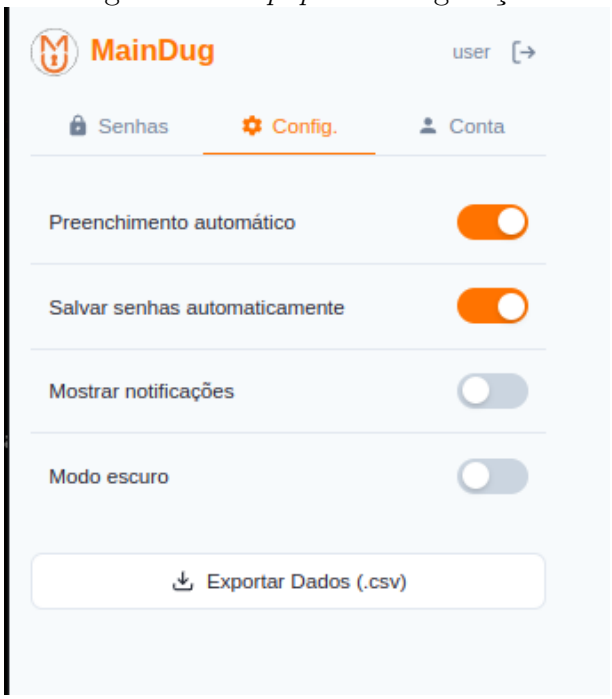
Figura 23: *Popup* da tela inicial.



Popup gerado pela extensão *web* que se torna a página inicial do aplicativo enquanto o usuário estiver logado. Dentro dele o usuário pode ver todas as credenciais salvas e copiar-las

5.2.3 *Popup* de configurações

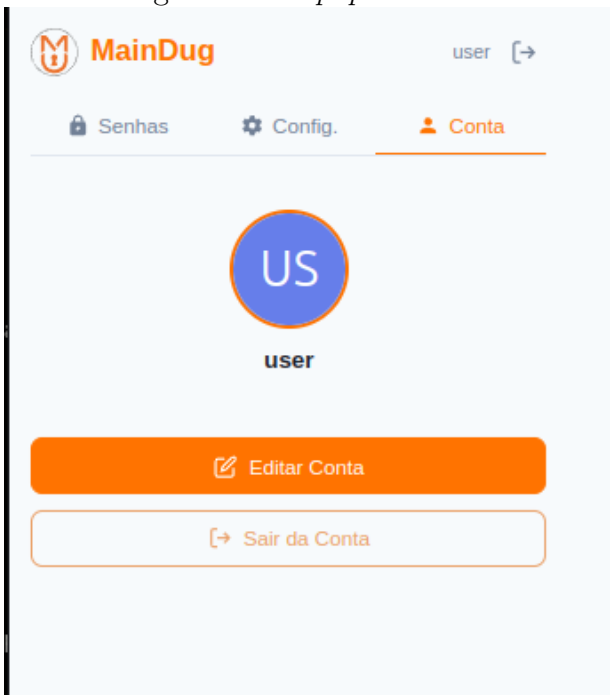
Figura 24: *Popup* da configuração.



Tela responsavel pela configuração do comportamento da extensão *web*. Dentro desta página existem quatro opções de personalização, sendo, respectivamente, o preenchimento automático de senhas (RF-004), a geração de senhas seguras (RF-003), o salvamento de credenciais automático, o recebimento de notificações e o modo escuro da extensão. Cada uma dessas opções pode ser ativada ou desativada pelo usuário. Além destas opções, aqui o usuário tambem pode exportar suas credenciais salvas em um arquivo '.csv'.

5.2.4 *Popup* de conta

Figura 25: *Popup* de conta.



Tela destinada ao gerenciamento da conta do usuário dentro da extensão *web*. Nela o usuario pode sair da conta atual ou ser redirecionado ao *webservice* para editar suas informações de perfil.

6 Conclusões

O desenvolvimento do *MainDug* demonstrou ser uma resposta técnica viável e necessária ao crescente problema da centralização de dados e da perda de autonomia do usuário em serviços de gerenciamento de credenciais. Este trabalho atingiu seu objetivo principal ao projetar e prototipar uma arquitetura de código aberto que, diferentemente das soluções proprietárias analisadas, coloca o controle da infraestrutura e dos dados criptografados diretamente nas mãos do usuário. Embora o protótipo atual sirva como uma fundação robusta, o caminho para uma aplicação pronta para produção exigirá auditorias de segurança independentes e a expansão para plataformas móveis, como sugerido para trabalhos futuros. Em suma, o MainDug cumpre sua proposta de valor, não apenas como uma aplicação funcional, mas como uma prova de conceito de que é possível desenhar sistemas que fortalecem ativamente o direito à privacidade e à soberania digital do indivíduo na era da informação.

Referências

1PASSWORD: A strong password is the first step to better security. 1Password. 2025. Disponível em: https://1password.com. Acesso em: 8 nov. 2025.

CHROME Extensions: Develop - Overview. URL original, não utilizar links do Google Translate. Google. 2025. Disponível em: https://developer.chrome.com/docs/extensions/develop. Acesso em: 8 nov. 2025.

GERENCIADOR de senhas do Google. Google. 2025. Disponível em: https://passwords.google.com. Acesso em: 8 nov. 2025.

LASTPASS: O gerenciador de senhas nº 1. LastPass. 2025. Disponível em: https://www.lastpass.com/pt. Acesso em: 8 nov. 2025.

CRYPTOGRAPHIC Storage Cheat Sheet. Open Web Application Security Project (OWASP). 2023. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html. Acesso em: 8 nov. 2025.

OWASP Top 10. Open Web Application Security Project (OWASP). 2021. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 8 nov. 2025.

PASSWORD Storage Cheat Sheet. Open Web Application Security Project (OWASP). 2024. Disponível em: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. Acesso em: 8 nov. 2025.

SQLALCHEMY - The Database Toolkit for Python. SQLAlchemy. 2025. Disponível em: https://www.sqlalchemy.org. Acesso em: 8 nov. 2025.

FLASK Documentation. The Pallets Projects. 2025. Disponível em: https://flask.palletsprojects.com/en/stable. Acesso em: 8 nov. 2025.

POSTGRESQL: The World's Most Advanced Open Source Relational Database. The PostgreSQL Global Development Group. 2025. Disponível em: https://www.postgresql.org. Acesso em: 8 nov. 2025.