



Desenvolvimento de Software 1 - Projeto Flash Pomo

Jala University

David Alex Souza Santos
Gustavo Henrique Jesus da Silva
Rhuan Miguel Esteves
Rinaldo Lira de Albuquerque Lima

Sumário

Introdução	2
BRD	3
Desenvolvimento	6
Processo de Desenvolvimento	15
Engenharia de Software	20
Manual do Usuário	96
Mapa do Site	109
API Reference	110
Schemas	111
Auth Endpoints	114
User Endpoints	118
Ensemble Endpoints	123
Flashcard Endpoints	128
PomodoroStory Endpoints	135
Favorite Endpoints	140
Box Endpoints	146
Purchase Endpoints	153
Tecnologias Utilizadas	160
Conclusão	162
Referencias Bibliográficas	163

Introdução

O aplicativo desenvolvido é voltado para o meio acadêmico, combinando as estratégias de estudo Pomodoro e Flashcards em um único hub. Ele oferece funcionalidades que permitem a criação e edição de flashcards, além de gerenciar o tempo de estudo de forma eficiente através da técnica Pomodoro.

Além disso, o aplicativo conta com funcionalidades de mercado, onde os usuários podem negociar seus flashcards usando uma moeda própria da aplicação. Essa integração de ferramentas proporciona um ambiente de aprendizado dinâmico e interativo, favorecendo o desenvolvimento acadêmico e a troca de conhecimento entre os usuários.

Objetivo do Relatório

O objetivo deste relatório é documentar a criação, o desenvolvimento e o funcionamento de um software voltado para o contexto acadêmico, abrangendo suas etapas de concepção, implementação, modelagem e sua aplicabilidade no ambiente de estudos.

BRD

A business requirements document (BRD) é um relatório detalhado que descreve tudo o que um novo projeto requer para obter sucesso. Este documento delineia os objetivos do projeto, o que é esperado ao longo do ciclo de vida do projeto e o que é necessário para alcançar o projeto.

Resumo Executivo

Este relatório descreve o desenvolvimento de um software acadêmico criado como parte do curso de Desenvolvimento de Software 1 da Jala University. O sistema desenvolvido combina a técnica Pomodoro de gerenciamento de tempo com o uso de flashcards, criando um ambiente de aprendizado dinâmico.

Objetivo do projeto

O aplicativo permite aos usuários gerenciar suas atividades de estudo com as seguintes funcionalidades:

- Gerenciamento de Usuários: Cadastro, edição e exclusão de usuários.
- Pomodoro: Ativação, pausa e desativação do timer de estudo.
- Relatórios do Pomodoro: Geração e consulta de relatórios baseados na técnica Pomodoro.
- Flashcards: Criação, edição, consulta e exclusão de flashcards.
- Histórico de Flashcards: Adição e remoção de flashcards do histórico.
- Favoritos: Gerenciamento de flashcards favoritos.
- Categorias de Flashcards: Criação e organização de flashcards por categorias.
- Marketplace de Flashcards: Compra e venda de flashcards utilizando uma moeda interna.

Escopo do projeto

A equipe será composta por um Product Owner, dois desenvolvedores e um Scrum Master, trabalhando sob a metodologia ágil Scrum.

Requerimento de negócios

Requisito de Negócio	Prioridade	Nível Crítico
Integração de Funcionalidades Pomodoro e Flashcards	1	Crítico
Facilidade de Uso	2	Médio
Escalabilidade	3	Médio
Acessibilidade e Segurança	4	Médio
Mercado de Flashcards	5	Baixo

Key Stakeholders

Nome	Ocupação
Rhuan Miguel Esteves	Product Owner
David Alex Souza Santos	Desenvolvedor Front
Gustavo Henrique de Jesus da Silva	Desenvolvedor Back
Rinaldo Gabriel Lira de Albuquerque Lima	Scrum Master

Restrições de Projeto

Restrição	Descrição
Prazo	Completo em 8 semanas, devido ao tempo de conclusão do curso
Recursos Humanos	A equipe tem apenas 4 membros para o desenvolvimento do projeto
Orçamento	O projeto deve ser desenvolvido com ferramentas gratuitas ou de baixo custo, sem despesas adicionais
Tecnologia	A aplicação deve ser desenvolvida com tecnologias já conhecidas pelos integrantes da equipe
Escopo	As funcionalidades, como sign up, log in, flashcards e pomodoro devem ser entregues como mínimo viável

Custo Benefício

Benefício	Descrição
Desenvolvimento	Experiência prática em desenvolvimento de software e metodologias ágeis, alinhando-se ao mercado de trabalho.
Treinamento e Aprendizado	Facilita a organização, produtividade e retenção de conhecimento, beneficiando usuários finais.

Desenvolvimento

Projeto

Este projeto visa desenvolver um aplicativo voltado para o meio acadêmico, combinando as estratégias de estudo Pomodoro e Flashcards em uma única plataforma. O objetivo é criar uma solução completa que permita aos usuários gerenciar seu tempo de estudo de forma eficiente e organizar seus materiais de aprendizado de maneira efetiva.

Uma das principais funcionalidades do aplicativo será a integração da técnica Pomodoro. Essa técnica de gerenciamento de tempo ajudará os usuários a manterem sua concentração e produtividade durante os momentos de estudo. Além disso, o aplicativo contará com ferramentas de criação e edição de flashcards, uma estratégia amplamente utilizada por estudantes para memorização e revisão de conteúdo.

Outra funcionalidade chave será o mercado interno, onde os usuários poderão negociar seus flashcards usando uma moeda própria da plataforma. Essa integração de um ambiente de compra e venda de material de estudo criará um ecossistema dinâmico de aprendizado, incentivando a troca de conhecimento e a colaboração entre os usuários.

Ao combinar estratégias de estudo comprovadas, gerenciamento de tempo e um mercado interno, o aplicativo se posicionará como uma solução inovadora e completa para as necessidades do meio acadêmico. Espera-se que essa integração de ferramentas melhore o desempenho e o engajamento dos estudantes, além de fomentar a colaboração e o compartilhamento de conhecimento entre os usuários.

O desenvolvimento deste aplicativo visa atender às demandas do público acadêmico, oferecendo uma plataforma que integre eficientemente as principais estratégias de estudo e aprendizado.

Objetivo Geral

O objetivo geral deste projeto é desenvolver um aplicativo que combine as estratégias de estudo Pomodoro e Flashcards, oferecendo aos usuários do meio acadêmico uma plataforma integrada e eficiente para gerenciar seu tempo de estudo e organizar seus materiais de aprendizado.

Objetivos Específicos

- 1. Implementar a funcionalidade de Pomodoro:** integrar a técnica Pomodoro no aplicativo, permitindo que os usuários gerenciem seu tempo de estudo de forma eficiente, alternando períodos de trabalho intenso e breves intervalos.
- 2. Desenvolver ferramentas de criação e edição de Flashcards:** criar uma interface intuitiva para que os usuários possam criar, armazenar, editar e organizar seus flashcards de forma eficaz.
- 3. Estabelecer um Mercado Interno de Flashcards:** implementar um sistema de compra e venda de flashcards entre os usuários, utilizando uma moeda própria da aplicação, a fim de fomentar a troca de conhecimento e a colaboração na comunidade acadêmica.
- 4. Integrar as funcionalidades de Pomodoro e Flashcards:** garantir uma experiência fluida e integrada entre as estratégias de estudo Pomodoro e o uso de Flashcards, otimizando o fluxo de aprendizado dos usuários.
- 5. Desenvolver uma Interface Intuitiva e Acessível:** criar uma interface de usuário intuitiva, responsiva e acessível, de modo a facilitar a adoção e a usabilidade do aplicativo por parte do público-alvo.
- 6. Promover a Colaboração e Compartilhamento de Conhecimento:** Através do Mercado Interno de Flashcards, incentivar a colaboração e o compartilhamento de conhecimento entre os usuários do aplicativo, fomentando o aprendizado mútuo.

Mercado

Análise de Mercado

Público Alvo

O aplicativo tem como público-alvo estudantes de ensino médio ao nível universitário, abrangendo:

Perfil	Descrição
Estudantes do Ensino Médio	Alunos que buscam ferramentas para melhorar sua organização e produtividade nos estudos, preparando-se para o ambiente universitário.
Estudantes Universitários	Alunos de graduação e pós-graduação que precisam gerenciar seu tempo de estudo e organizar seu material de aprendizado de forma eficiente.
Estudantes Autônomos	Indivíduos que estudam de forma independente, como preparatórios para concursos ou cursos livres, e necessitam de estratégias de estudo eficazes.

Esse público-alvo foi selecionado com base na necessidade de ferramentas que combinem eficientemente a gestão do tempo de estudo (técnica Pomodoro) e a organização de conteúdo (flashcards), além da oportunidade de compartilhar e trocar conhecimento por meio do mercado interno de flashcards, atendendo às demandas dos estudantes desde o ensino médio até o nível universitário.

Análise da Concorrência

Ao analisar o mercado, foram identificados alguns aplicativos e plataformas que oferecem funcionalidades semelhantes ao aplicativo proposto, embora de forma fragmentada. Alguns dos principais concorrentes são:

Concorrente	Funcionalidades	Pontos Fortes	Pontos Fracos
Anki	- Criação e revisão de flashcards - Gerenciamento de deck de flash cards	- Ampla base de usuários - Algoritmo de repetição espaçada eficiente	- Não possui integração com a técnica Pomodoro - Interface pouco intuitiva
Forest	- Técnica Pomodoro - Gerenciamento de tempo de estudo	- Interface amigável - Gamificação da técnica Pomodoro	- Não possui funcionalidades de flashcards - Apenas gerenciamento de tempo, sem organização de conteúdo
Quizlet	- Criação e revisão de flashcards - Compartilhamento de flashcards	- Ampla biblioteca de flashcards compartilhados - Ferramentas de aprendizado interativas	- Não possui integração com a técnica Pomodoro - Foco apenas na funcionalidade de flashcards, sem gerenciamento de tempo

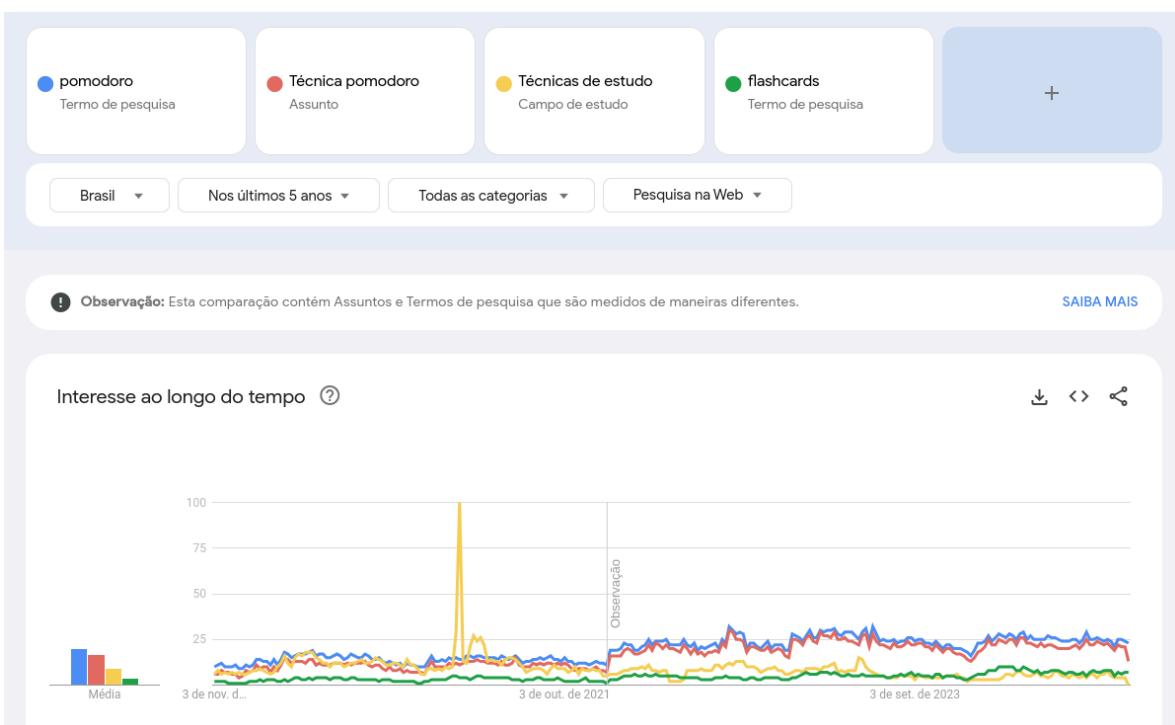
Essa análise da concorrência permite identificar as principais lacunas no mercado, as quais o aplicativo proposto pode preencher ao combinar as funcionalidades de gerenciamento de tempo (Pomodoro) e organização de conteúdo (flashcards), além de oferecer um mercado interno de flashcards para a troca de conhecimento entre os usuários.

Tendências:

- **Crescente demanda por ferramentas de gerenciamento de tempo e organização de estudo:** Estudantes e profissionais do meio acadêmico buscam cada vez mais soluções integradas que os ajudem a gerenciar seu tempo de estudo e aprendizado de forma eficiente.

- **Adoção da técnica Pomodoro no ambiente acadêmico:** A técnica Pomodoro vem ganhando popularidade entre estudantes e pesquisadores, que a utilizam para aumentar sua produtividade durante os momentos de estudo e trabalho.
- **Necessidade de compartilhamento e colaboração no aprendizado:** Existe uma demanda crescente por plataformas que permitam a troca de conhecimento e a colaboração entre estudantes e profissionais do meio acadêmico.

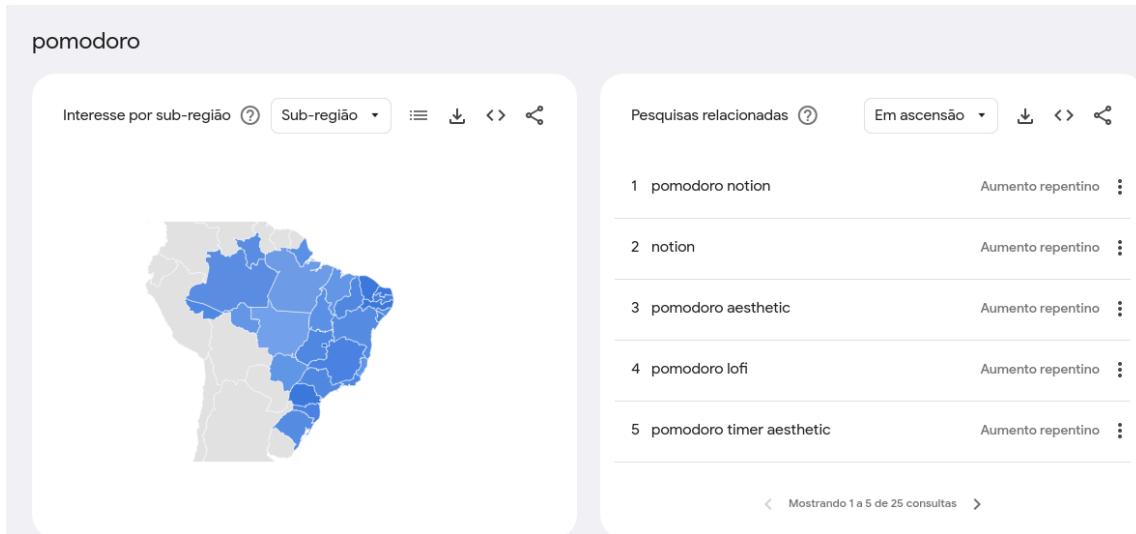
Usando alguns termos para pesquisa no Google Trends vemos que há uma busca ativa sobre os termos chaves como: pomodoro, flashcards, etc



grafico_trends.png

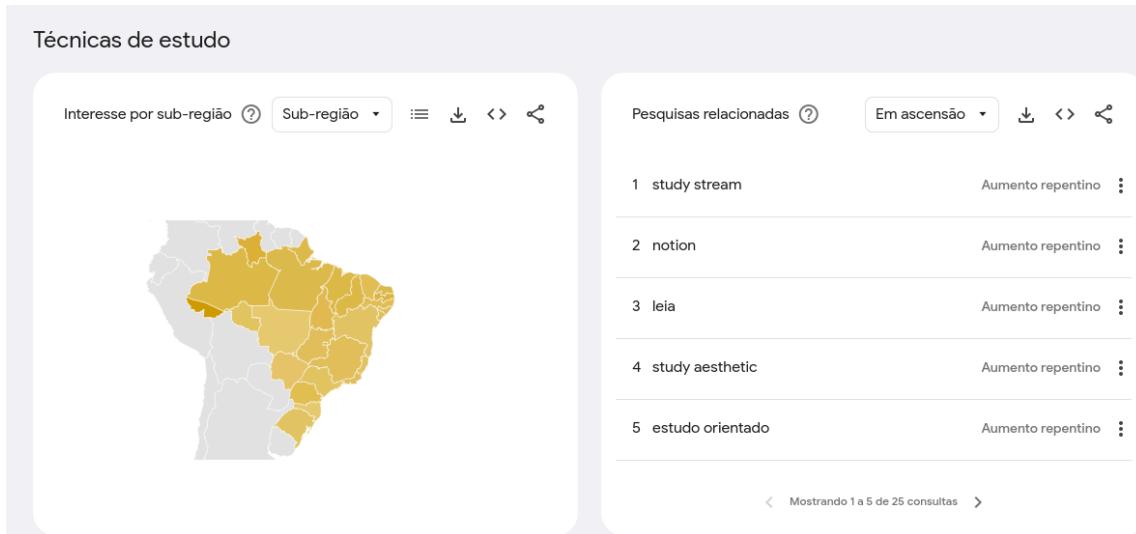
Gráficos por termos:

- Pomodoro



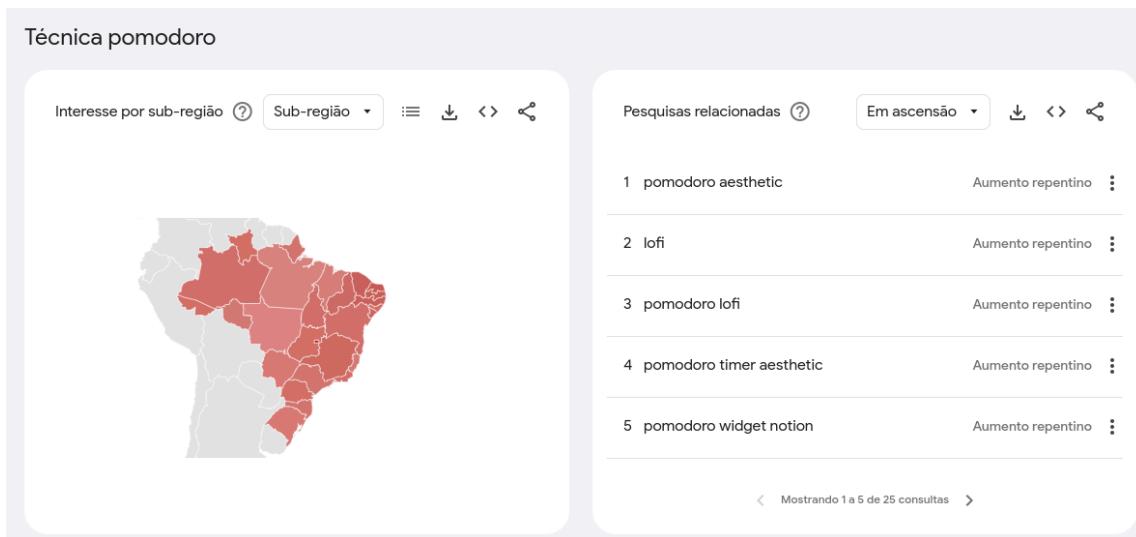
pomodoro.png

- Técnica Estudo



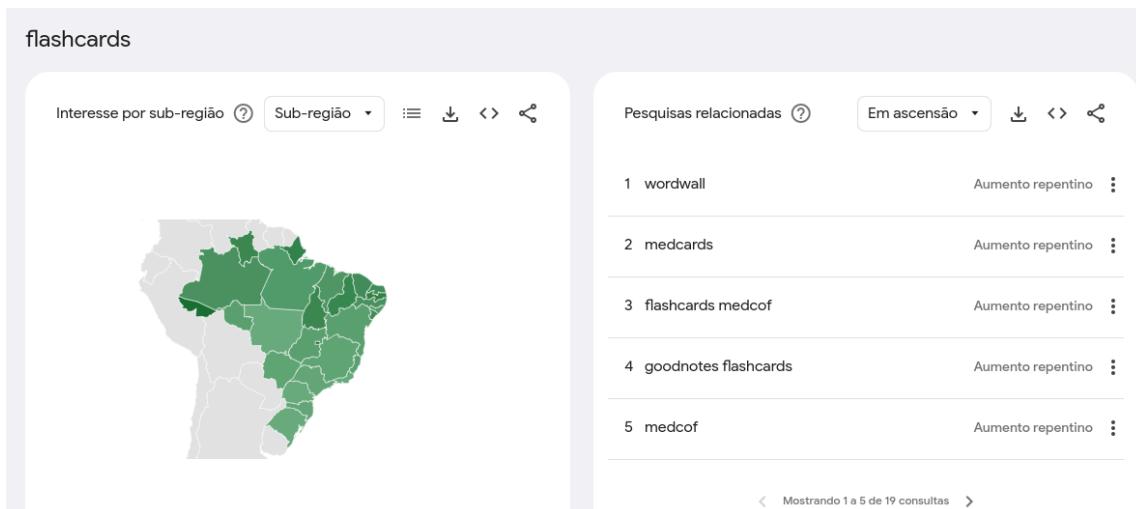
tecnica_estudo.png

- Técnica Pomodoro



tecnica_pomodoro.png

- Flashcards



flashcards.png

Oportunidades:

- **Integração de ferramentas de gerenciamento de tempo e organização de conteúdo:** O mercado carece de soluções que combinem de forma eficiente e fluida a técnica Pomodoro e a utilização de flashcards, atendendo às necessidades dos estudantes.
- **Criação de um ecossistema de compartilhamento de conhecimento:** O desenvolvimento de um mercado interno de flashcards, onde os usuários possam comprar, vender e trocar materiais de estudo, representa uma oportunidade

significativa para fomentar a colaboração e a troca de conhecimento no meio acadêmico.

- **Diferenciação por meio da integração de funcionalidades:** A maioria das soluções atuais se concentra em apenas uma das necessidades (gerenciamento de tempo ou organização de flashcards), deixando uma lacuna a ser preenchida por um aplicativo que integre essas funcionalidades de forma completa.

Essas tendências e oportunidades de mercado indicam que o aplicativo proposto pode preencher uma demanda relevante do público-alvo, oferecendo uma solução integrada e inovadora para o gerenciamento do tempo de estudo e a organização do conteúdo de aprendizado, além de promover a colaboração e a troca de conhecimento entre os usuários.

Posicionamento e Diferencial Competitivo

Com base na análise do mercado e das necessidades do público-alvo, o aplicativo proposto se posicionará como uma solução integrada e inovadora, com os seguintes diferenciais competitivos:

1. **Combinação de Pomodoro e Flashcards:** O aplicativo será a única solução do mercado que combina de forma fluida e eficiente a técnica Pomodoro para gerenciamento do tempo de estudo e as funcionalidades de criação, organização e revisão de flashcards. Essa integração permitirá que os usuários otimizem seu processo de aprendizado.
2. **Mercado Interno de Flashcards:** A implementação de um mercado interno, onde os usuários poderão comprar, vender e trocar flashcards usando uma moeda própria da plataforma, será um diferencial competitivo. Essa funcionalidade fomentará a colaboração e a troca de conhecimento entre os estudantes e profissionais do meio acadêmico.
3. **Foco no Público Acadêmico:** Ao concentrar-se especificamente nas necessidades dos estudantes do ensino médio, universitários e profissionais do meio acadêmico, o aplicativo se diferenciará de soluções genéricas de gerenciamento de tempo e flashcards, oferecendo uma experiência personalizada e adaptada às demandas desse público-alvo.
4. **Interface Intuitiva e Acessível:** O aplicativo priorizará o desenvolvimento de uma interface de usuário intuitiva, responsiva e acessível, facilitando a adoção e a

usabilidade por parte dos estudantes e profissionais do meio acadêmico, independentemente de seu nível de familiaridade com esse tipo de ferramenta.

Esse posicionamento estratégico, aliado aos diferenciais competitivos do aplicativo, permitirá que o produto se destaque no mercado e se torne uma solução de referência para o gerenciamento eficiente do tempo de estudo e a organização do conteúdo de aprendizado no meio acadêmico.

Processo de Desenvolvimento

O processo de desenvolvimento do software segue as práticas ágeis do Scrum, permitindo que a equipe trabalhe de maneira eficiente e iterativa. Abaixo estão os principais componentes que orientam a entrega das funcionalidades e a organização do time.

User Stories

As *User Stories* descrevem as funcionalidades e requisitos que os usuários esperam do sistema. Elas são priorizadas para garantir que as funcionalidades mais críticas sejam entregues primeiro. Cada *User Story* possui uma descrição clara e uma prioridade associada para guiar a implementação.

ID	Descrição	Prioridade
US 001	Como usuário, eu quero criar uma conta, para que eu possa fazer login e utilizar o software.	Alta
US 002	Como usuário, eu quero fazer login no sistema, para que eu possa acessar minhas informações e funcionalidades.	Alta
US 003	Como usuário, eu quero realizar o logout na minha conta, para que eu possa encerrar minha sessão de forma segura e proteger minhas informações pessoais.	Média
US 004	Como usuário eu quero editar meus dados na minha conta de usuário.	Média
US 005	Como usuário, eu quero poder excluir minha conta, para que eu possa remover meu perfil do sistema e minhas informações pessoais.	Baixa
US 006	Como usuário, eu quero poder ativar o método Pomodoro no sistema, para que eu possa gerenciar melhor meu tempo e produtividade durante minhas tarefas.	Alta
US 007	Como usuário eu quero poder pausar o método Pomodoro no sistema, caso eu não queira mais utilizá-lo.	Média
US 008	Como usuário eu quero criar meus flashcards.	Alta

US 00 9	Como usuário eu quero editar meus flashcards.	Média
US 010	Como usuário eu quero excluir meus flashcards.	Média
US 011	Como usuário eu quero criar ensembles.	Média
US 012	Como usuário eu quero editar minhas ensembles.	Média
US 013	Como usuário eu quero excluir minhas ensembles.	Baixa

Essas *User Stories* são o ponto de partida para o desenvolvimento do produto, com foco em atender as necessidades mais urgentes e importantes para os usuários.

Organização da Equipe

A equipe de desenvolvimento é composta por profissionais especializados que garantem o bom andamento do projeto, seguindo as metodologias ágeis para garantir a entrega contínua e a adaptação do produto às necessidades dos usuários. A estrutura da equipe é a seguinte:

Ocupação	Descrição
Product Owner	Responsável por gerenciar o produto e garantir que as funcionalidades atendam às necessidades dos usuários.
Desenvolvedor Front	Responsável pelo desenvolvimento da interface do usuário, garantindo uma experiência intuitiva e agradável.
Desenvolvedor Back	Responsável pelo desenvolvimento da lógica e funcionalidades do aplicativo, implementando as regras de negócio.
Scrum Master	Responsável por facilitar o processo de desenvolvimento, removendo impedimentos e aplicando as práticas ágeis da equipe.
Customer	Representa os usuários finais do aplicativo, cujas necessidades e feedback são fundamentais para o desenvolvimento do produto.

Organização Inicial

A equipe inicial é formada pelos seguintes membros:

Nome	Ocupação
David Alex Souza Santos	Desenvolvedor Front
Gustavo Henrique de Jesus da Silva	Desenvolvedor Back
Rhuan Miguel Esteves	Desenvolvedor Front
Rinaldo Gabriel Lira de Albuquerque Lima	Desenvolvedor Back

Cada membro da equipe tem um papel essencial para garantir que o desenvolvimento seja conduzido de forma ágil e eficiente, assegurando que as funcionalidades sejam entregues conforme o planejado.

Daily Reports

Para monitorar o progresso diário, são gerados *Daily Reports*, que são acessíveis através do link fornecido. Esses relatórios ajudam a equipe a se alinhar, identificar bloqueios e promover a colaboração contínua. O acompanhamento regular dessas reuniões e relatórios é fundamental para manter o ritmo de desenvolvimento e resolver qualquer obstáculo que possa surgir.



Para acessar os relatórios diários, clique aqui

(https://docs.google.com/spreadsheets/d/1o2UgdqqENKe9Lt3L4_an7J_vgRb6nPahw7DFNoWM57E/edit?usp=sharing).

Engenharia de Software

A engenharia de software é a abordagem sistemática para o desenvolvimento, operação e manutenção de sistemas de software. Nesse contexto, a organização da documentação técnica do nosso projeto é essencial para garantir a qualidade, a manutenibilidade e a evolução do sistema.

Como afirma Pressman (2010), "a documentação técnica é a espinha dorsal da engenharia de software, pois captura os requisitos, a arquitetura, a implementação e os procedimentos operacionais de um sistema".

Requisitos

Requisitos Funcionais

Requisito	Descrição	Prioridade	Caso de Uso
RF01	O sistema deverá permitir o cadastro, edição e exclusão de usuários.	1	UC01
RF02	O sistema deverá permitir ao usuário logar e deslogar sua conta.	2	UC02, UC03
RF03	O sistema deverá permitir ao usuário iniciar, desativar e pausar o temporizador.	1	UC04
RF04	O sistema deverá permitir realizar ao usuário as funções de criar e consultar o relatório dos pomodoros.	3	UC05
RF05	O sistema deverá permitir ao usuário realizar as funções de criar, editar, consultar e excluir flashcards.	1	UC06
RF06	O sistema deverá permitir ao usuário realizar as funções de adicionar e remover flashcards do histórico.	2	UC07
RF07	O sistema deverá permitir ao usuário realizar as funções de adicionar e remover flashcards dos favoritos.	2	UC08
RF08	O sistema deverá permitir ao usuário realizar as funções de criar, consultar, editar e excluir conjuntos dos flashcards.	1	UC09
RF09	O sistema deverá permitir ao usuário a compra de flashcards usando a moeda do sistema.	3	UC10
RF10	O sistema deverá permitir a criação de box sells para venda dos flashcards.	1	UC11

RF11	O sistema deverá permitir ao usuário a venda de flashcard s.	1	UC12
------	--	---	------

Requisitos Não Funcionais

Requisito	Descrição	Prioridade
RNF01	O sistema deve ter uma interface responsiva e adaptável a diferentes tamanhos de tela.	1
RNF02	O sistema deve ter um design intuitivo e de fácil usabilidade.	1
RNF03	O sistema deve ter um tempo de resposta rápido, mesmo com um grande número de usuários.	2
RNF04	O sistema deve garantir a segurança e privacidade dos dados dos usuários.	1
RNF05	O sistema deve ser acessível, seguindo as diretrizes de acessibilidade web.	2
RNF06	O sistema deve ser multiplataforma, funcionando em diferentes sistemas operacionais e dispositivos.	2
RNF07	O sistema deve fornecer documentação clara e detalhada para os desenvolvedores.	3
RNF08	O sistema deve realizar backups automáticos diários para prevenir perda de dados.	3
RNF09	O sistema deve permitir integrações com APIs externas para funcionalidades adicionais.	3

Arquitetura de Informação

A arquitetura de informação (AI) para a documentação de software do nosso projeto deve ser orientada a fornecer uma estrutura clara e intuitiva para organizar e disseminar o conhecimento técnico. Segundo Rosenfeld e Morville (2015), a AI "envolve a concepção da estrutura e da navegação de um espaço de informação para facilitar a execução de tarefas e o acesso ao conteúdo".

Essa estrutura de arquitetura de informação ajudará a garantir que a documentação do nosso projeto de software seja compreensível, acessível e eficaz para os desenvolvedores e outras partes interessadas.

Wireframe

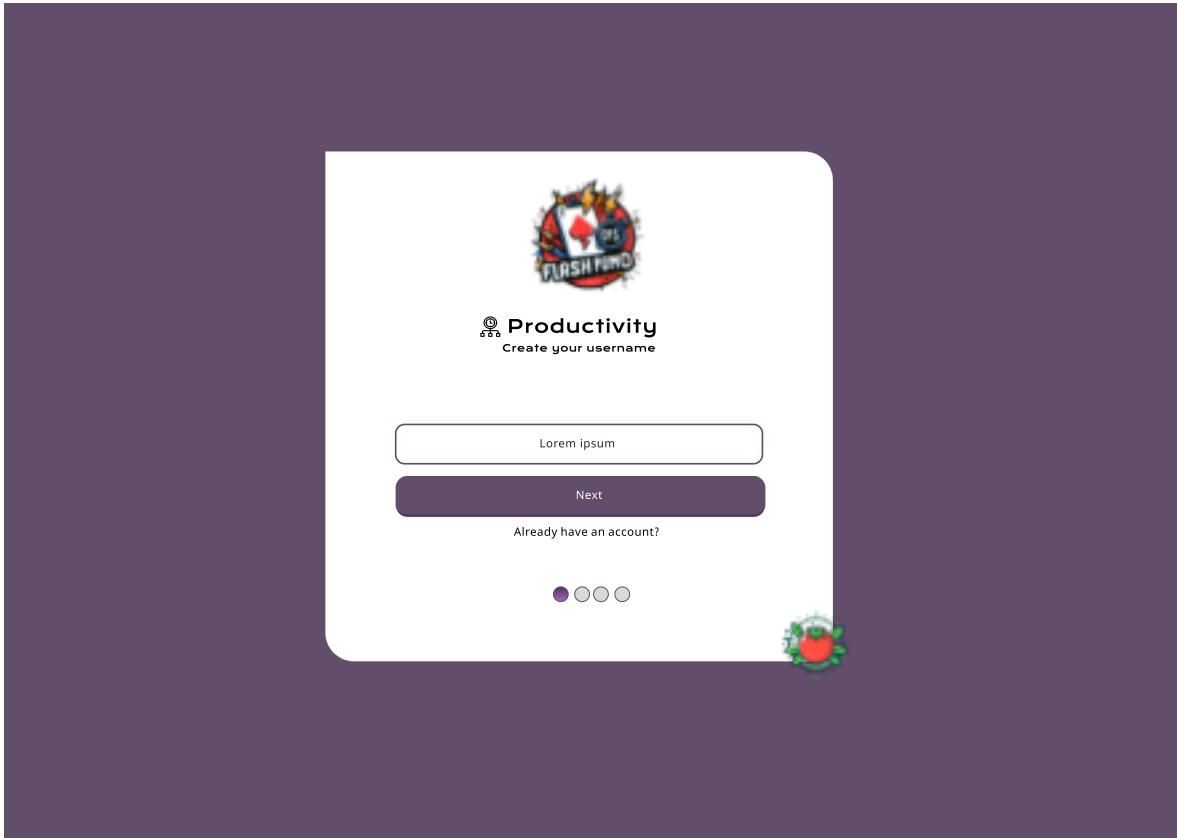


Link para o Figma do projeto:

<https://www.figma.com/design/keAudUZHHInEkyf2bfI5ef/Flash-Pomo?node-id=0-1&t=EVf3oq7Mg9h7qvvg-1>

Wireframe de cadastro - Campo Username

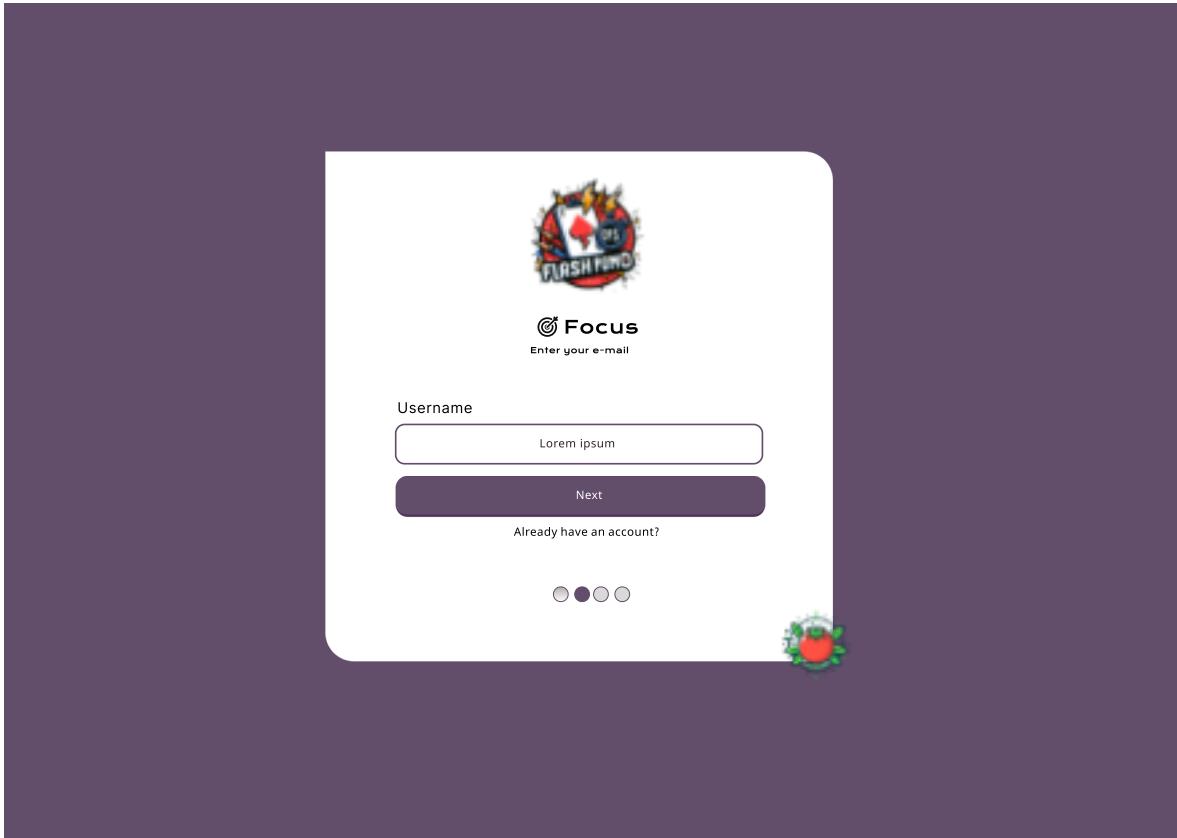
Este wireframe apresenta o campo para o usuário inserir seu nome de usuário no formulário de cadastro.



CADASTRO - 1 - USERNAME

Wireframe de cadastro - Campo Email

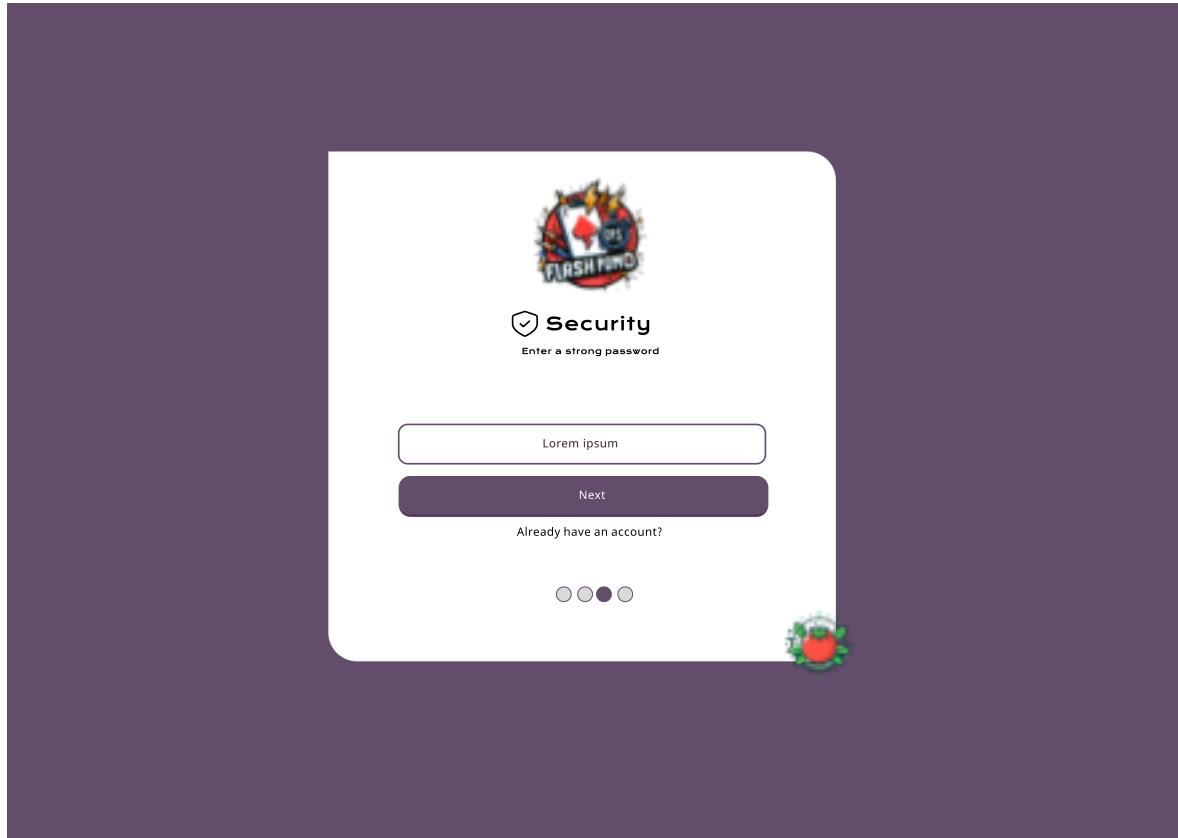
Este wireframe mostra o campo onde o usuário insere seu email no formulário de cadastro.



CADASTRO - 1 - EMAIL

Wireframe de cadastro - Campo Senha

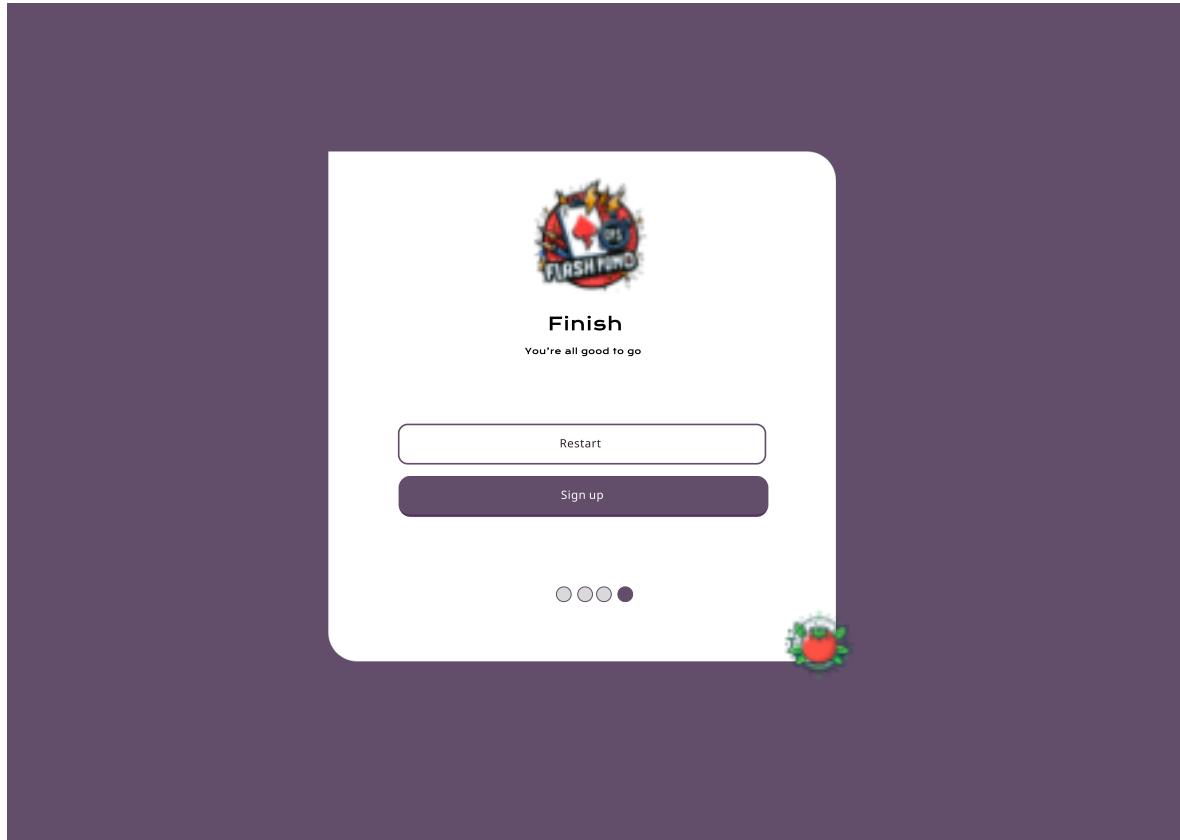
Este wireframe detalha o campo para o usuário criar sua senha no formulário de cadastro.



CADASTRO - 1 - PASSWORD

Wireframe de cadastro - Botão "Sign Up"

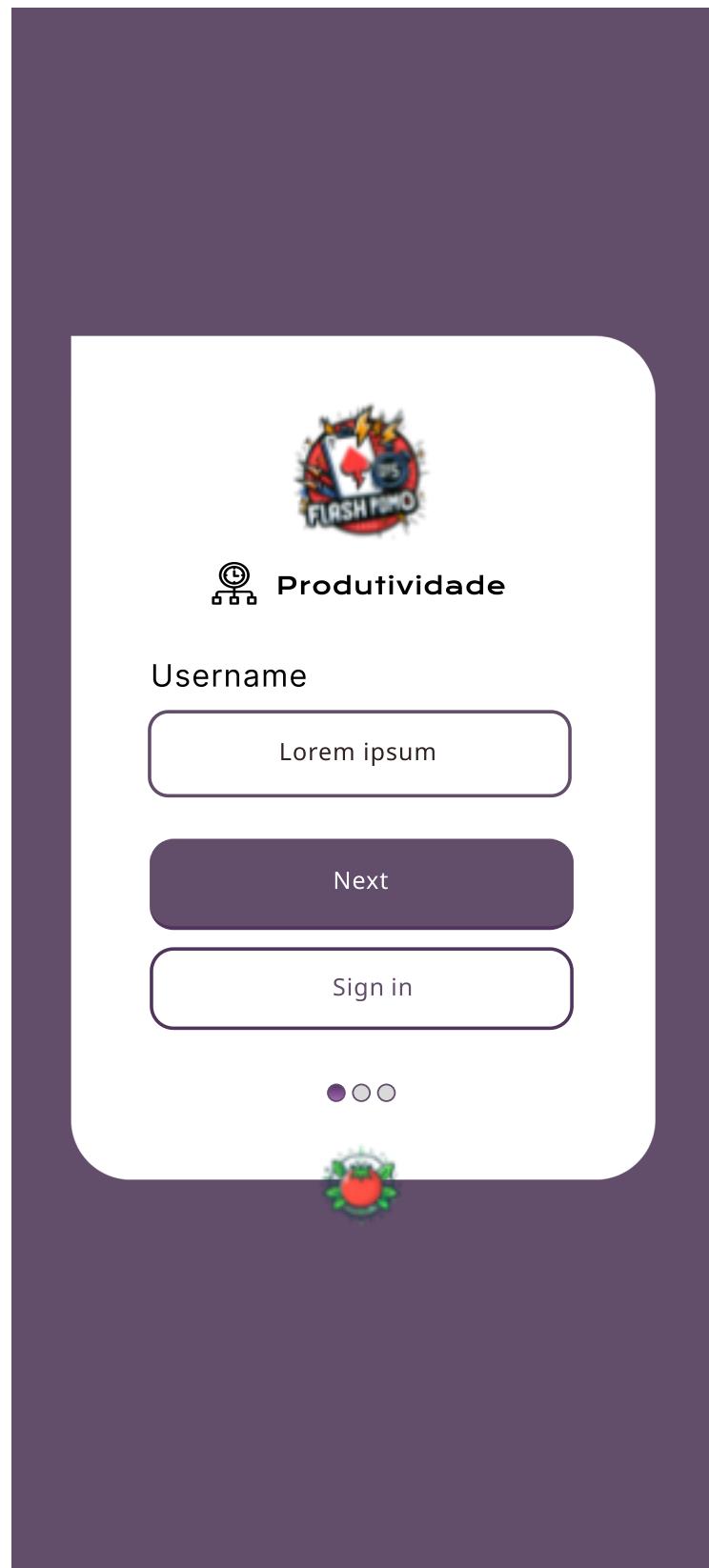
Este wireframe apresenta o botão de confirmação para finalizar o cadastro do usuário.



CADASTRO - 1 - SIGN UP

Wireframe de cadastro - Versão Mobile (Etapa 1)

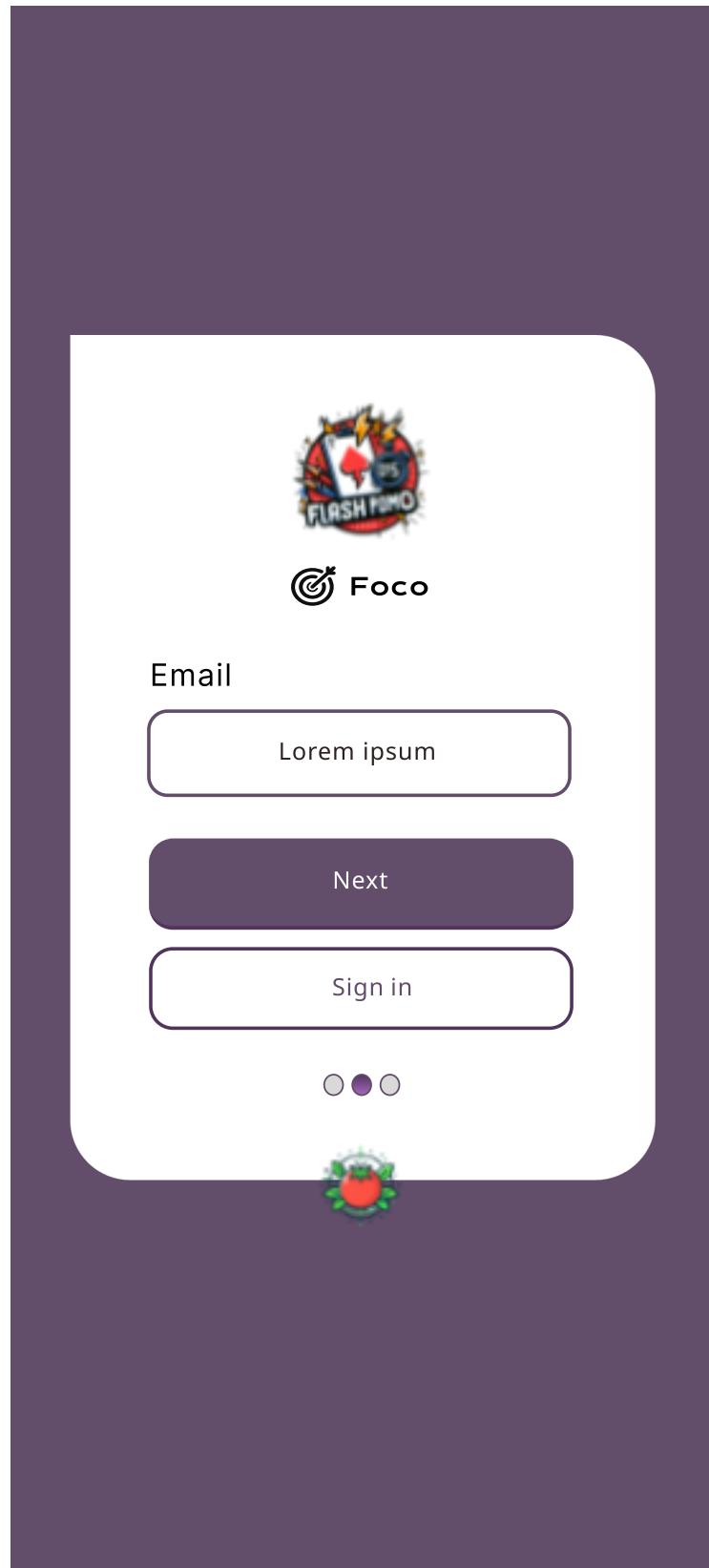
Este wireframe exibe o design para a primeira etapa do cadastro em dispositivos móveis.



CADASTRO - 1 - MOBILE

Wireframe de cadastro - Versão Mobile (Etapa 2)

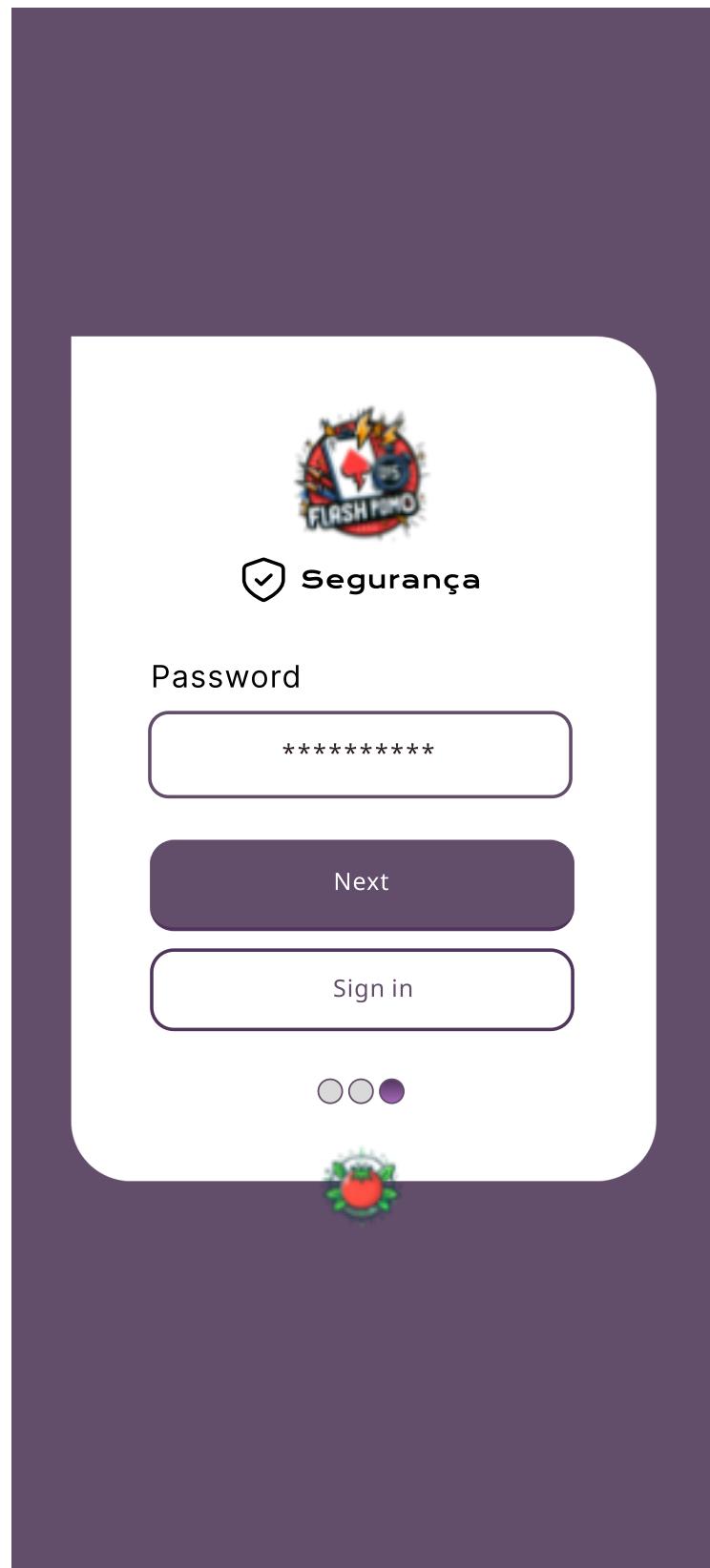
Este wireframe exibe a segunda etapa do cadastro em dispositivos móveis.



CADASTRO - 2 - MOBILE

Wireframe de cadastro - Versão Mobile (Etapa 3)

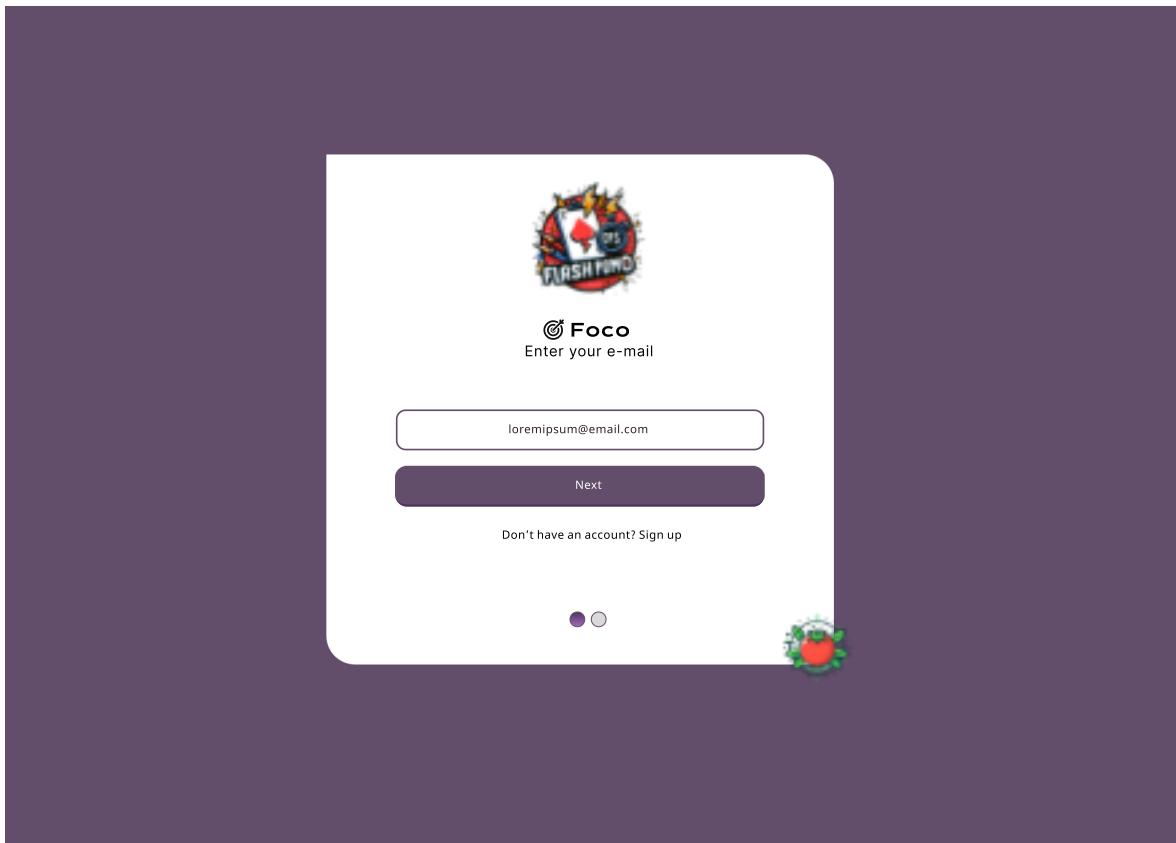
Este wireframe mostra a terceira etapa do cadastro em dispositivos móveis.



CADASTRO - 3 - MOBILE

Wireframe de login – Versão Desktop (Página Inicial)

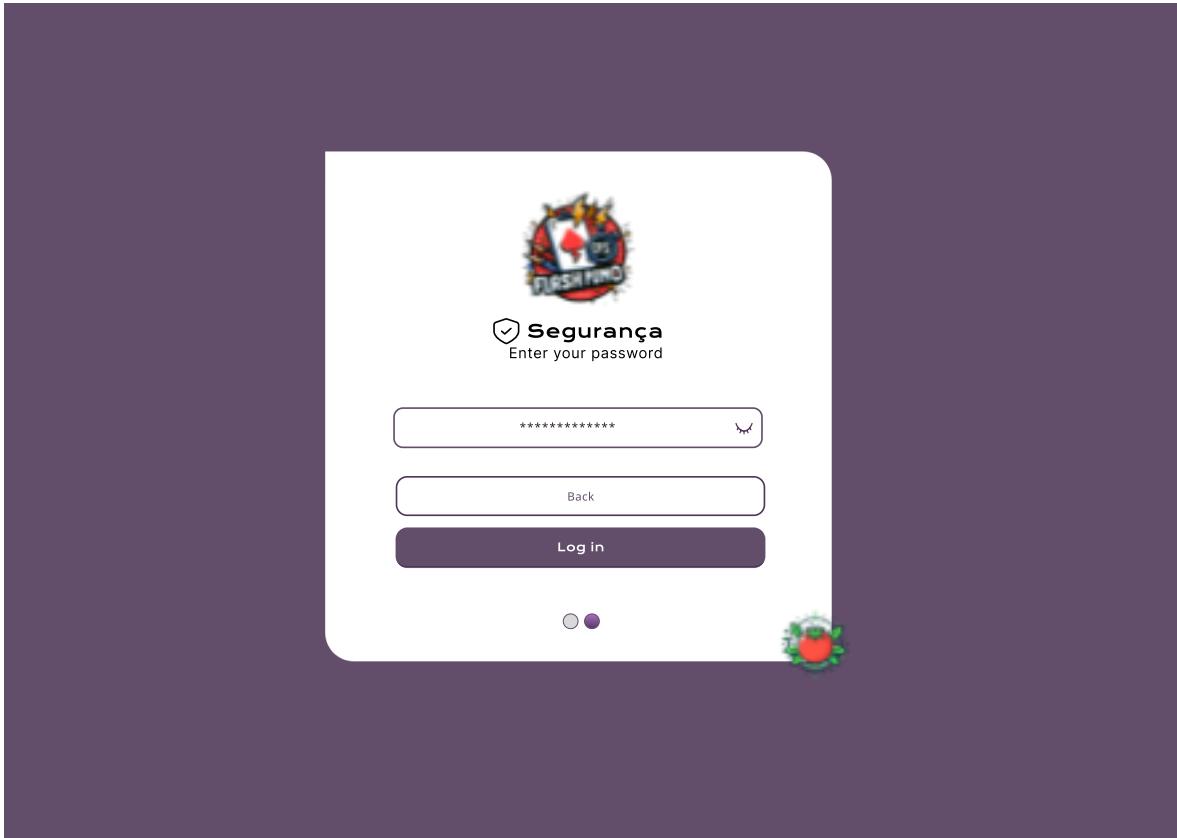
Este wireframe apresenta a interface de login para usuários em dispositivos desktop.



LOGIN - 1 - DESKTOP

Wireframe de login - Segurança

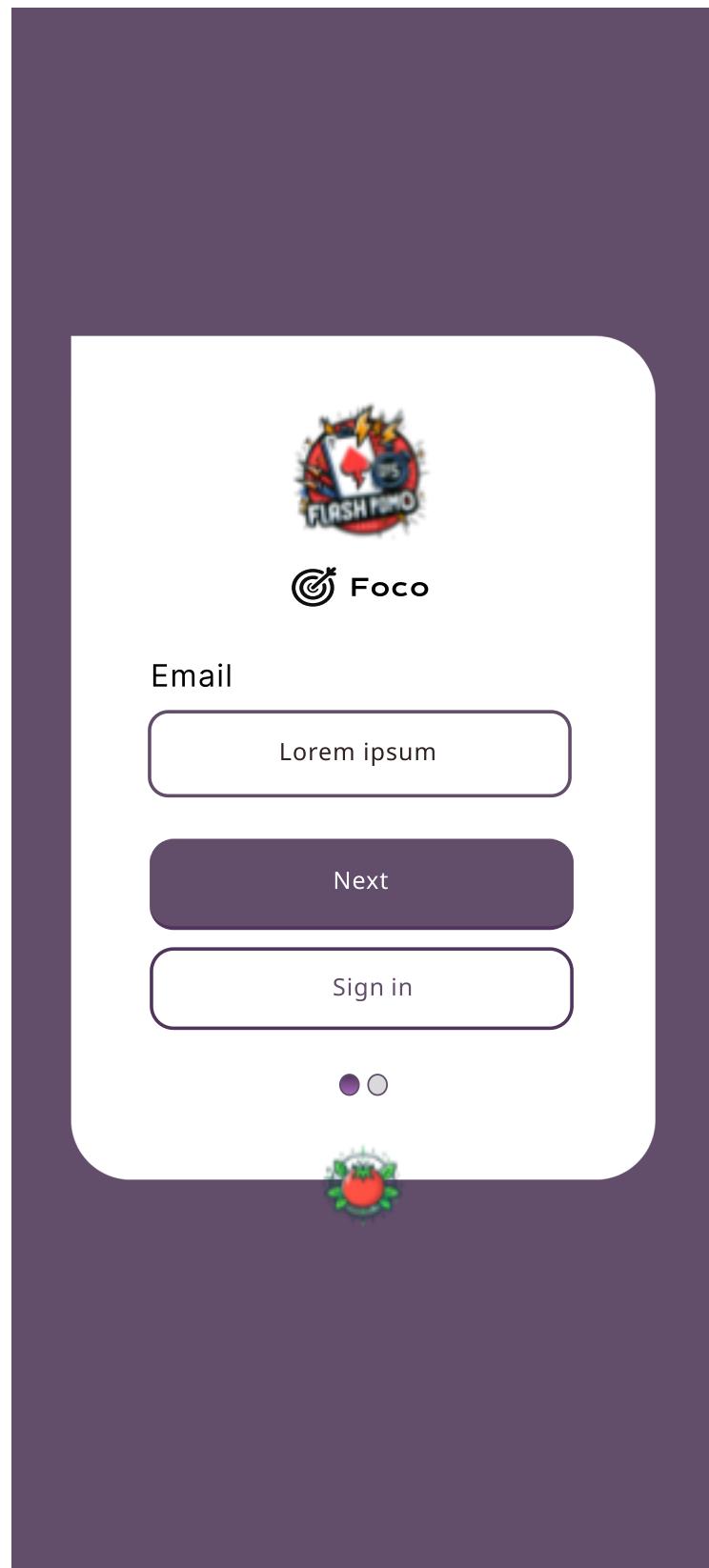
Este wireframe destaca a funcionalidade de segurança na interface de login.



LOGIN - 1 - SECURITY

Wireframe de login - Versão Mobile (Etapa 1)

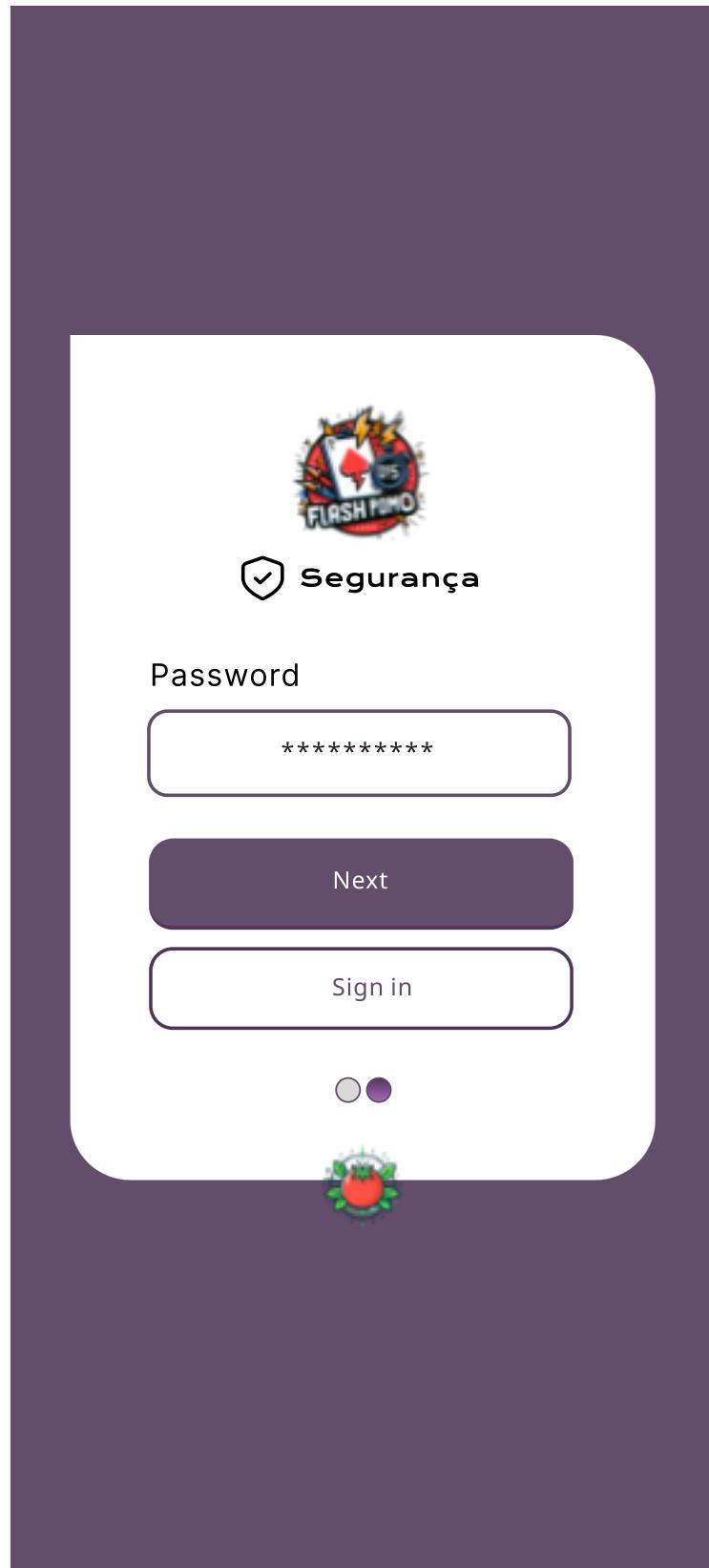
Este wireframe ilustra a primeira etapa do login em dispositivos móveis.



LOGIN - 2 - MOBILE

Wireframe de login – Versão Mobile (Etapa 2)

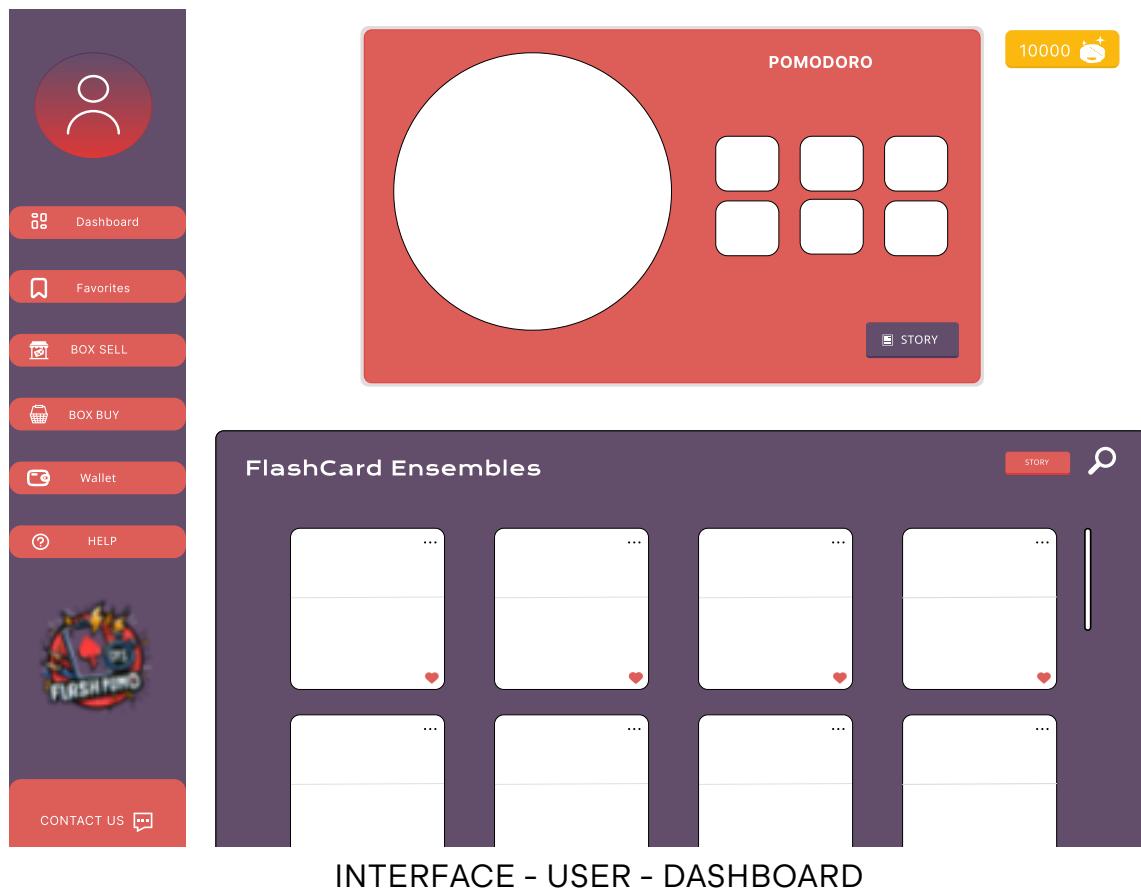
Este wireframe apresenta a segunda etapa do login em dispositivos móveis.



LOGIN - 3 - MOBILE

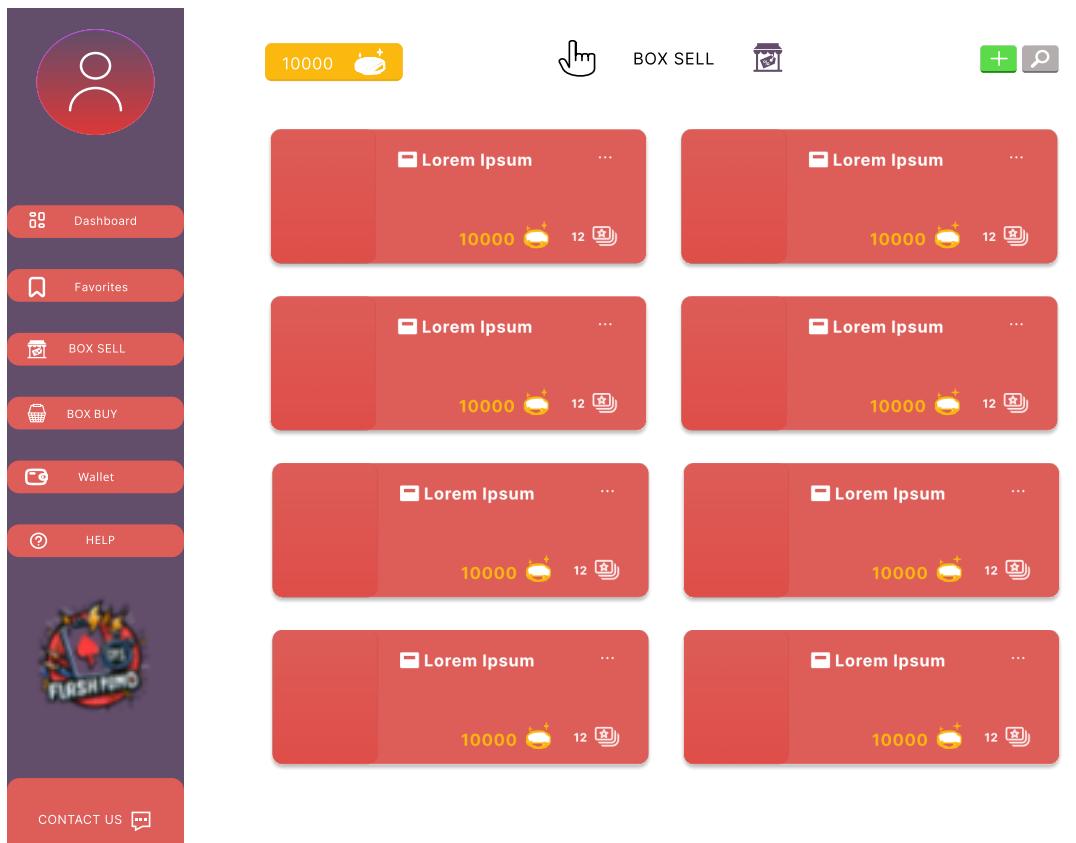
Wireframe da interface - Painel do Usuário

Este wireframe apresenta o painel principal da interface do usuário.



Wireframe da interface - Caixa principal no desktop

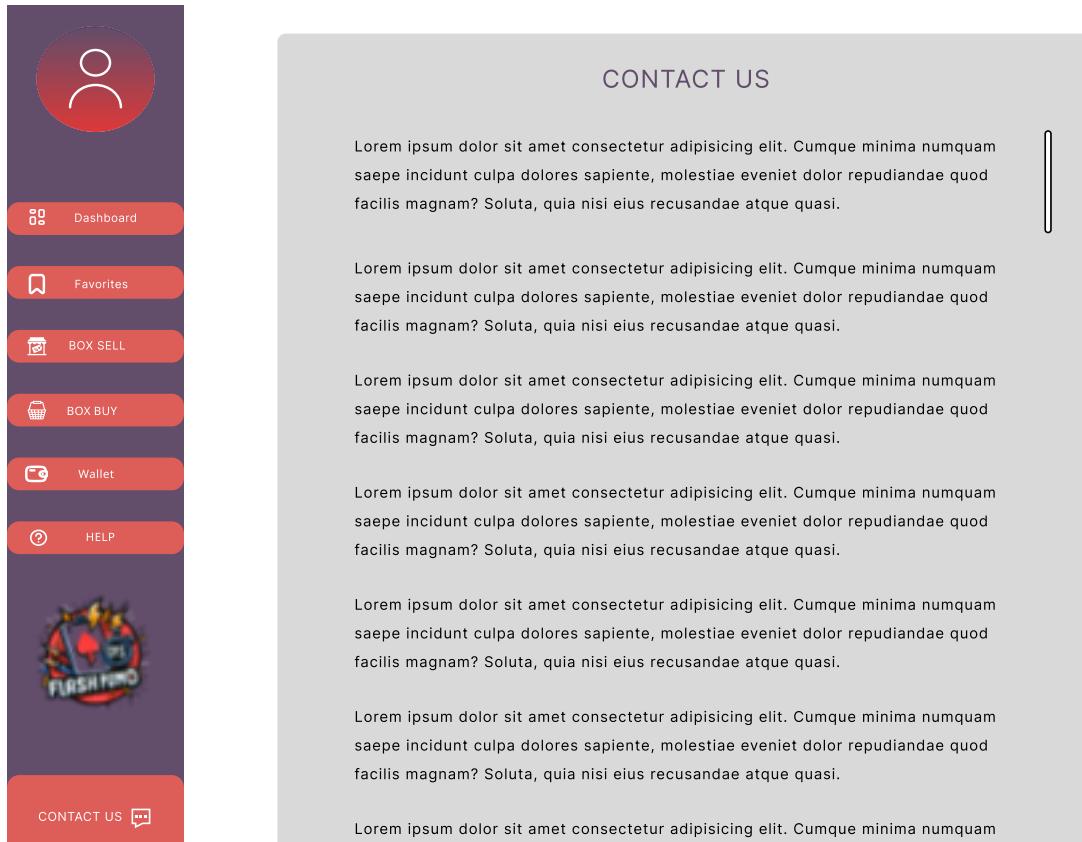
Este wireframe exibe o layout da caixa principal da interface para desktop.



DESKTOP - USER - BOX SHELL

Wireframe da interface - Página "Contato"

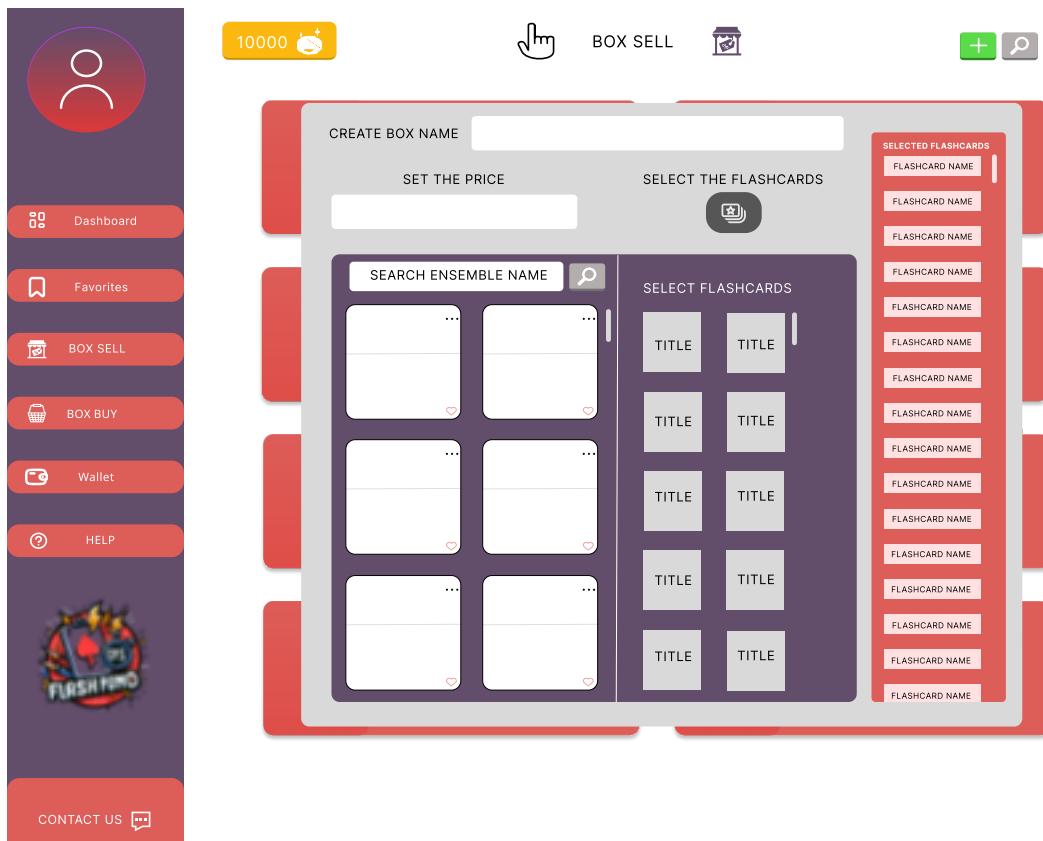
Este wireframe ilustra a página de contato para suporte ao usuário.



DESKTOP - USER - CONTACT US

Wireframe da interface - Criando nova caixa

Este wireframe apresenta a interface para criar uma nova caixa na plataforma.



DESKTOP - USER - CREATING BOX SHELL

Wireframe da interface - Página de Ajuda

Este wireframe mostra a interface da página de ajuda ao usuário.

The wireframe shows a user profile page with a sidebar on the left and a main content area on the right.

Left Sidebar:

- User icon (red gradient circle with a white person icon)
- Dashboard** (orange button with a bar chart icon)
- Favorites** (orange button with a bookmark icon)
- BOX SELL** (orange button with a box icon)
- BOX BUY** (orange button with a shopping cart icon)
- Wallet** (orange button with a wallet icon)
- HELP** (orange button with a question mark icon)

Main Content Area:

HELP

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

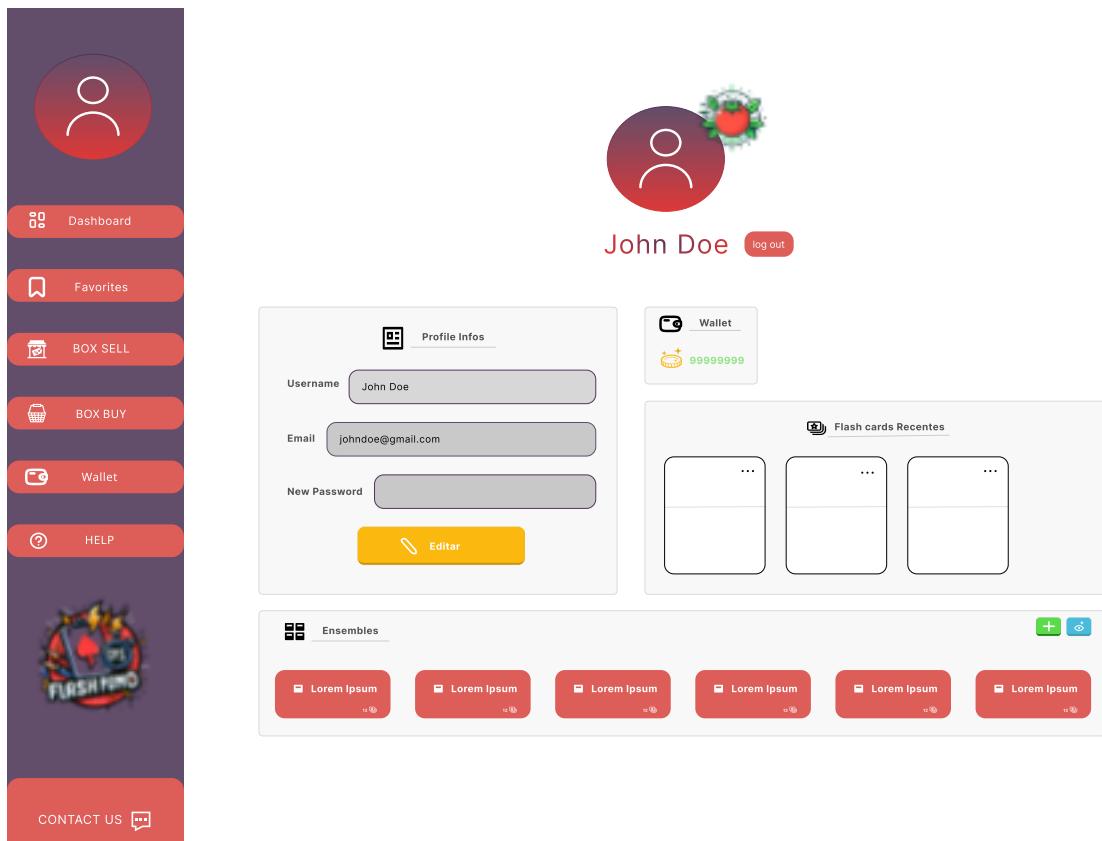
Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam saepe incident culpa dolores sapiente, molestiae eveniet dolor repudiandae quod facilis magnam? Soluta, quia nisi eius recusandae atque quasi.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Cumque minima numquam

DESKTOP - USER - HELP

Wireframe da interface - Perfil do Usuário

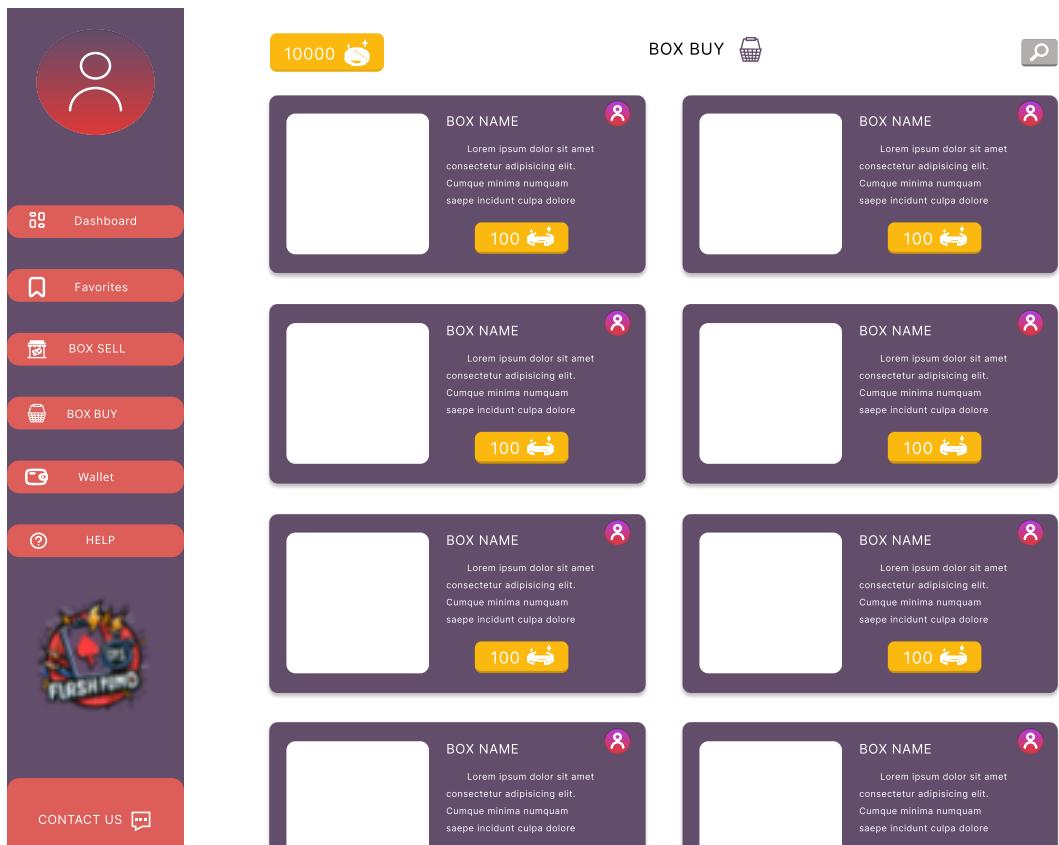
Este wireframe apresenta o layout da página de perfil do usuário.



DESKTOP - USER - USER PROFILE

Wireframe da interface - Compra de Caixa

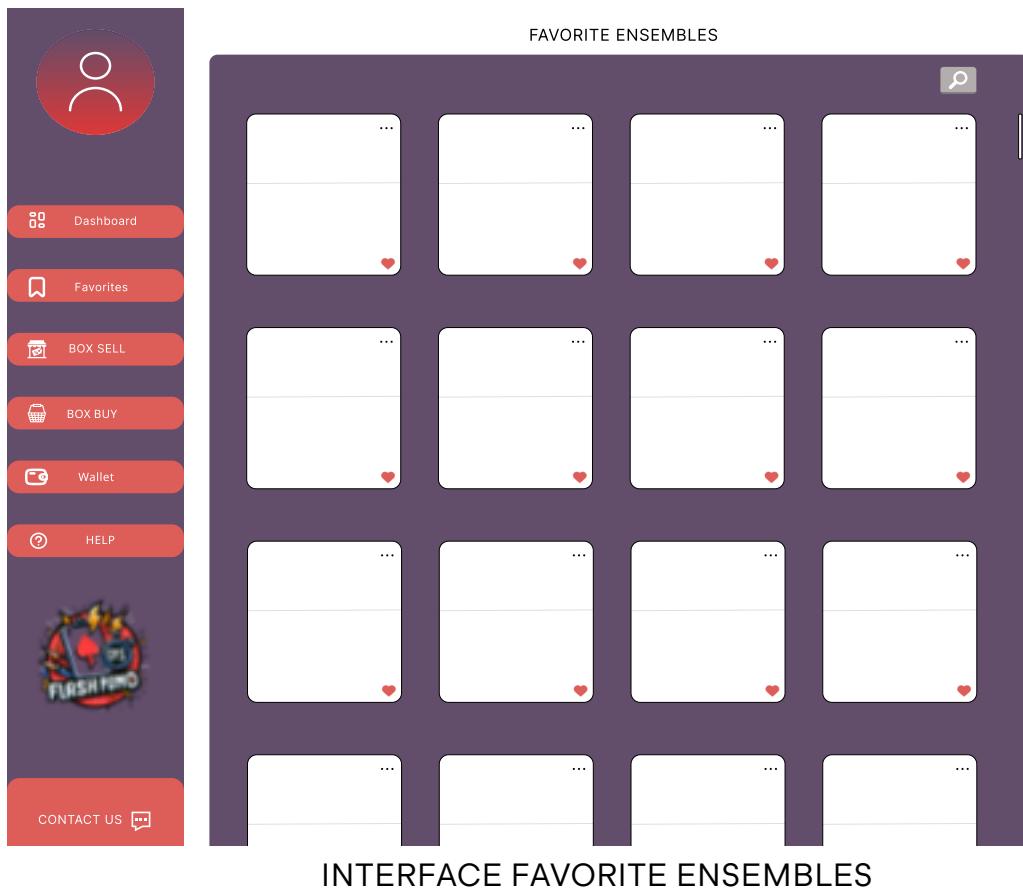
Este wireframe ilustra a interface de compra de caixas.



INTERFACE BOX BUY

Wireframe da interface - Favoritos

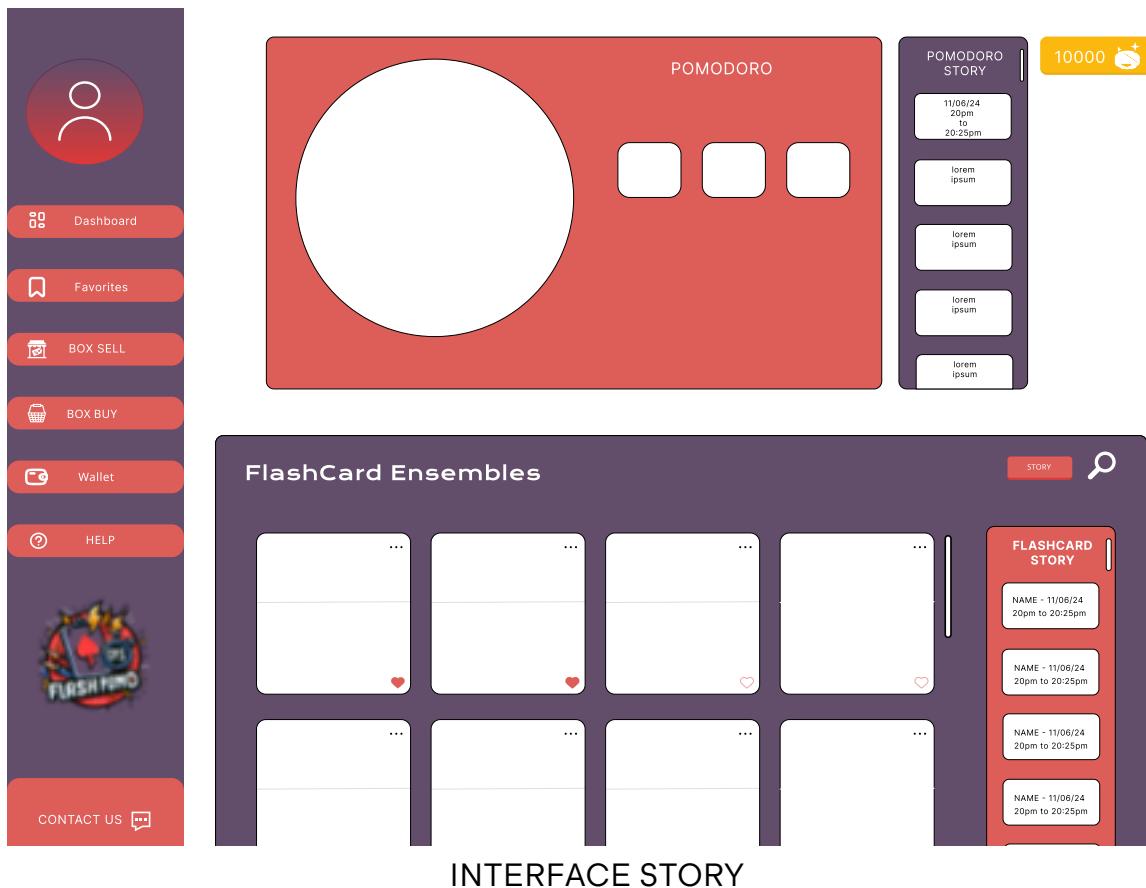
Este wireframe apresenta a seção de conjuntos favoritos do usuário.



INTERFACE FAVORITE ENSEMBLES

Wireframe da interface - Histórias

Este wireframe detalha a seção de histórias na interface do usuário.



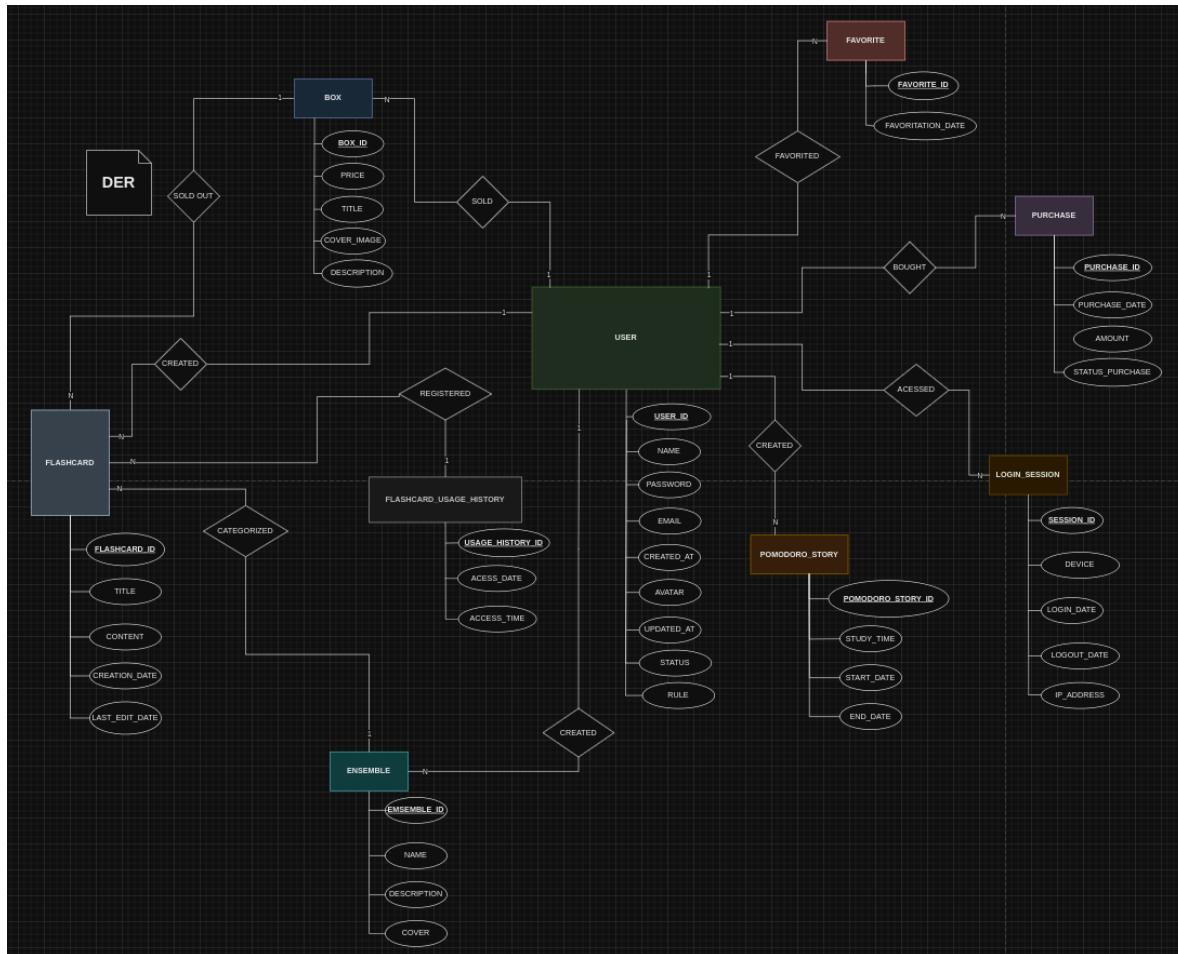
INTERFACE STORY

Banco de Dados

⚠ Resource:

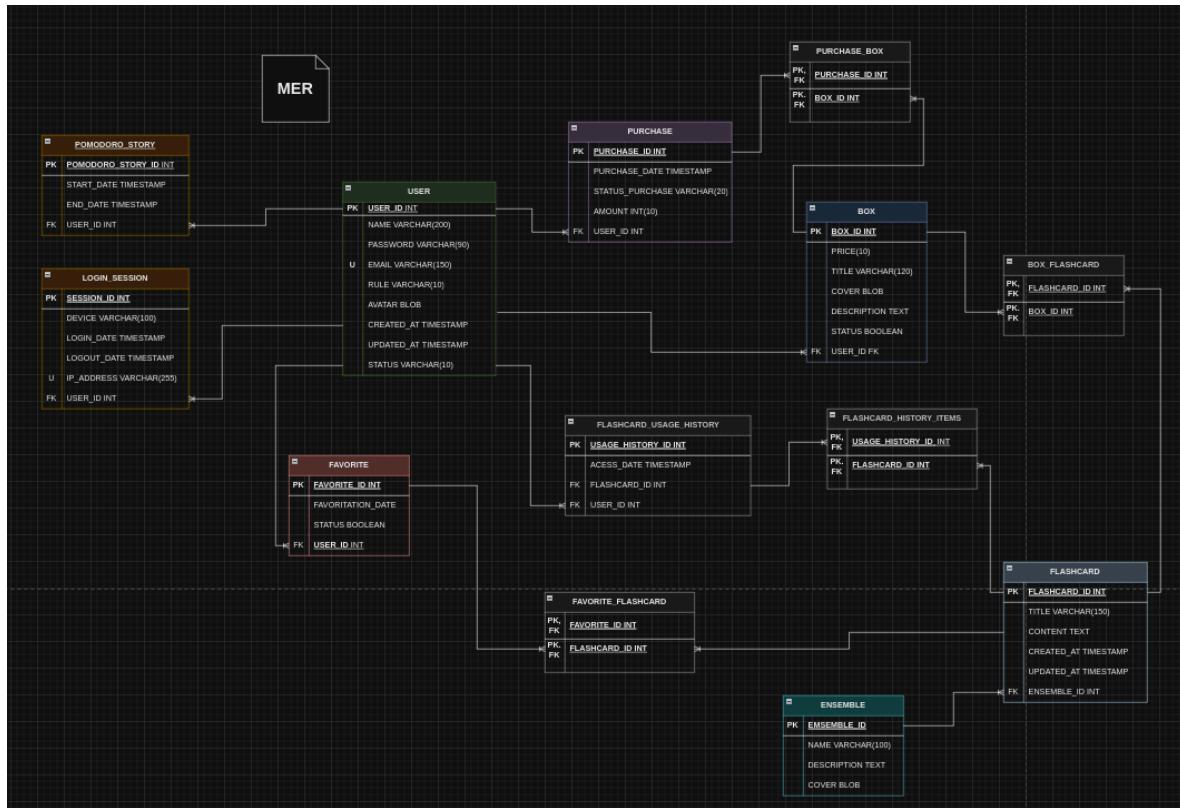
<https://drive.google.com/file/d/1IfJADPBk8vBuoPQ7H7mxsLTCevSHhe98/view?usp=sharing>
`(https://drive.google.com/file/d/1IfJADPBk8vBuoPQ7H7mxsLTCevSHhe98/vie`
`w?usp=sharing)`

Diagrama Entidade Relacionamento



DER.png

Modelo Entidade Relacionamento



MER.png

Modelo Físico

Tabela USER

-- Tabela para armazenar informações dos usuários

```
CREATE TABLE USER (
    USER_ID INT PRIMARY KEY,
    NAME VARCHAR(200),
    PASSWORD VARCHAR(90),
    EMAIL VARCHAR(150) UNIQUE,
    RULE VARCHAR(10),
    AVATAR BLOB,
    CREATED_AT TIMESTAMP,
    UPDATED_AT TIMESTAMP,
    CONSTRAINT UQ_EMAIL UNIQUE(EMAIL)
) COMMENT -- Identificador único do usuário
-- Nome do usuário
-- Senha do usuário para login
-- Email único do usuário
-- Função ou tipo de usuário
-- Imagem ou avatar do usuário
-- Data e hora em que o usuário foi criado
-- Data e hora da última
```

```

atualização
    STATUS VARCHAR(10)          -- Status do usuário
);

```

Tabela ENSEMBLE

- Tabela para armazenar informações sobre os conjuntos de flashcards

```

CREATE TABLE ENSEMBLE (
    ENSEMBLE_ID INT PRIMARY KEY,           -- Identificador único do
    conjunto
    NAME VARCHAR(100),                   -- Nome do conjunto
    DESCRIPTION TEXT,                    -- Descrição do conjunto
    COVER BLOB                          -- Capa ou imagem
representando o conjunto
);

```

Tabela FLASHCARD

- Tabela para armazenar informações dos flashcards

```

CREATE TABLE FLASHCARD (
    FLASHCARD_ID INT PRIMARY KEY,        -- Identificador único do
    flashcard
    TITLE VARCHAR(150),                 -- Título do flashcard
    CONTENT TEXT,                      -- Conteúdo ou descrição do
    flashcard
    CREATED_AT TIMESTAMP,              -- Data de criação do
    flashcard
    UPDATED_AT TIMESTAMP,              -- Data da última
    atualização do flashcard
    ENSEMBLE_ID INT,                  -- Identificador do conjunto
de flashcards a que este flashcard pertence
    FOREIGN KEY (ENSEMBLE_ID) REFERENCES ENSEMBLE(ENSEMBLE_ID)  --
);

```

```
Relacionamento com o conjunto de flashcards
);
```

Tabela FAVORITE

- Tabela para armazenar informações sobre os favoritos dos usuários

```
CREATE TABLE FAVORITE (
    FAVORITE_ID INT PRIMARY KEY,           -- Identificador único do
    favorito
    FAVORITATION_DATE_TIMESTAMP TIMESTAMP, -- Data e hora em que o
    flashcard foi marcado como favorito
    STATUS BOOLEAN,                      -- Status do favorito (se
    está ativo ou não)
    USER_ID INT,                         -- Identificador do usuário
    que marcou o favorito
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) --
    Relacionamento com o usuário
);
```

Tabela Pivo FAVORITE_FLASHCARD

- Tabela para associar flashcards a favoritos

```
CREATE TABLE FAVORITE_FLASHCARD (
    FAVORITE_ID INT,                     -- Identificador do favorito
    FLASHCARD_ID INT,                   -- Identificador do
    flashcard
    PRIMARY KEY (FAVORITE_ID, FLASHCARD_ID), -- Chave composta para
    garantir que um flashcard não seja duplicado nos favoritos
    FOREIGN KEY (FLASHCARD_ID) REFERENCES FLASHCARD(FLASHCARD_ID),
    -- Relacionamento com o flashcard
    FOREIGN KEY (FAVORITE_ID) REFERENCES FAVORITE(FAVORITE_ID) --
```

```
Relacionamento com o favorito
);
```

Tabela PivoFLASHCARD_USAGE_HISTORY

- Tabela para armazenar o histórico de uso dos flashcards pelos usuários

```
CREATE TABLE FLASHCARD_USAGE_HISTORY (
    USAGE_HISTORY_ID INT PRIMARY KEY,      -- Identificador único do
                                            -- histórico de uso
    ACCESS_DATE TIMESTAMP,                  -- Data e hora de acesso ao
                                            -- flashcard
    FLASHCARD_ID INT,                      -- Identificador do
                                            -- flashcard
    USER_ID INT,                          -- Identificador do usuário
                                            -- que acessou o flashcard
    FOREIGN KEY (FLASHCARD_ID) REFERENCES FLASHCARD(FLASHCARD_ID), -- Relacionamento com o flashcard
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) -- Relacionamento com o usuário
);
```

Tabela PivoFLASHCARD_HISTORY_ITEMS

- Tabela para armazenar os itens do histórico de uso dos flashcards

```
CREATE TABLE FLASHCARD_HISTORY_ITEMS (
    USAGE_HISTORY_ID INT,                  -- Identificador do
                                            -- histórico de uso
    FLASHCARD_ID INT,                      -- Identificador do
                                            -- flashcard
    PRIMARY KEY (USAGE_HISTORY_ID, FLASHCARD_ID), -- Chave composta
                                            -- para garantir a associação
    FOREIGN KEY (USAGE_HISTORY_ID) REFERENCES
    FLASHCARD_USAGE_HISTORY(USAGE_HISTORY_ID), -- Relacionamento com o
```

```

histórico de uso
  FOREIGN KEY (FLASHCARD_ID) REFERENCES FLASHCARD(FLASHCARD_ID) -
- Relacionamento com o flashcard
);

```

Tabela TABLE_BOX

- Tabela para armazenar as caixas de flashcards compradas pelos usuários

```

CREATE TABLE BOX (
  BOX_ID INT PRIMARY KEY,                               -- Identificador único da
  caixa
  PRICE INT,                                         -- Preço da caixa
  TITLE VARCHAR(120),                                -- Título da caixa
  COVER BLOB,                                         -- Capa da caixa
  DESCRIPTION TEXT,                                   -- Descrição da caixa
  STATUS BOOLEAN,                                     -- Status da caixa (ativa,
  inativa, etc.)
  USER_ID INT,                                         -- Identificador do usuário
  que comprou ou criou a caixa
  FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) -- Relacionamento com o usuário
);

```

Tabela BOX_FLASHCARD

- Tabela para associar flashcards a caixas

```

CREATE TABLE BOX_FLASHCARD (
  BOX_ID INT,                                         -- Identificador da caixa
  FLASHCARD_ID INT,                                    -- Identificador do
  flashcard
  PRIMARY KEY (BOX_ID, FLASHCARD_ID), -- Chave composta para
  garantir que um flashcard só apareça uma vez na caixa
  FOREIGN KEY (BOX_ID) REFERENCES BOX(BOX_ID), -- Relacionamento

```

```

com a caixa
    FOREIGN KEY (FLASHCARD_ID) REFERENCES FLASHCARD(FLASHCARD_ID) -
- Relacionamento com o flashcard
);

```

Tabela PURCHASE

- Tabela para registrar as compras realizadas pelos usuários

```

CREATE TABLE PURCHASE (
    EMAIL VARCHAR(150) UNIQUE,           -- Email único do usuário
    que fez a compra
    RULE VARCHAR(10),                   -- Função do usuário que
    fez a compra
    AVATAR BLOB,                      -- Avatar do usuário que
    fez a compra
    PURCHASE_ID INT PRIMARY KEY,       -- Identificador único da
    compra
    PURCHASE_DATE TIMESTAMP,           -- Data e hora da compra
    STATUS VARCHAR(20),                -- Status da compra
    (pendente, concluída, etc.)
    AMOUNT INT,                       -- Quantidade de itens na
    compra
    USER_ID INT,                      -- Identificador do usuário
    que fez a compra
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) --
    Relacionamento com o usuário
);

```

Tabela PivoPURCHASE_BOX

- Tabela para associar caixas a compras

```

CREATE TABLE PURCHASE_BOX (
    PURCHASE_ID INT,                  -- Identificador da compra

```

```

    BOX_ID INT,                      -- Identificador da caixa
comprada
    PRIMARY KEY (PURCHASE_ID, BOX_ID), -- Chave composta para
garantir que a mesma caixa não apareça mais de uma vez na mesma
compra
    FOREIGN KEY (PURCHASE_ID) REFERENCES PURCHASE(PURCHASE_ID), -- Relacionamento com a compra
    FOREIGN KEY (BOX_ID) REFERENCES BOX(BOX_ID) -- Relacionamento
com a caixa
);

```

Tabela POMODORO_STORY

- Tabela para registrar as sessões de pomodoro dos usuários:

```

CREATE TABLE POMODORO_STORY (
    POMODORO_STORY_ID INT PRIMARY KEY,      -- Identificador único da
sessão de pomodoro
    START_DATE_TIMESTAMP TIMESTAMP,          -- Data e hora de início
da sessão
    END_DATE_TIMESTAMP TIMESTAMP,            -- Data e hora de término
da sessão
    USER_ID INT,                          -- Identificador do
usuário que realizou a sessão
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) -- Relacionamento com o usuário
);

```

Tabela LOGIN_SESSION

- Tabela para registrar as sessões de login dos usuários:

```

CREATE TABLE LOGIN_SESSION (
    SESSION_ID INT PRIMARY KEY,           -- Identificador único da
sessão de login

```

```

    DEVICE VARCHAR(100),           -- Dispositivo usado para
login
    LOGIN_DATE_TIMESTAMP TIMESTAMP, -- Data e hora do login
    LOGOUT_DATE_TIMESTAMP TIMESTAMP, -- Data e hora do logout
    IP_ADDRESS VARCHAR(255),        -- Endereço IP de onde o
usuário fez login
    USER_ID INT,                  -- Identificador do
usuário que fez login
    FOREIGN KEY (USER_ID) REFERENCES USER(USER_ID) --
Relacionamento com o usuário
);

```

UML

A Unified Modeling Language (UML) é uma linguagem de modelagem visual padronizada amplamente utilizada na indústria de desenvolvimento de software. Conforme definido pela Object Management Group (OMG), "a UML fornece uma maneira padronizada de visualizar a estrutura e o comportamento de um sistema de software" (OMG, 2017).

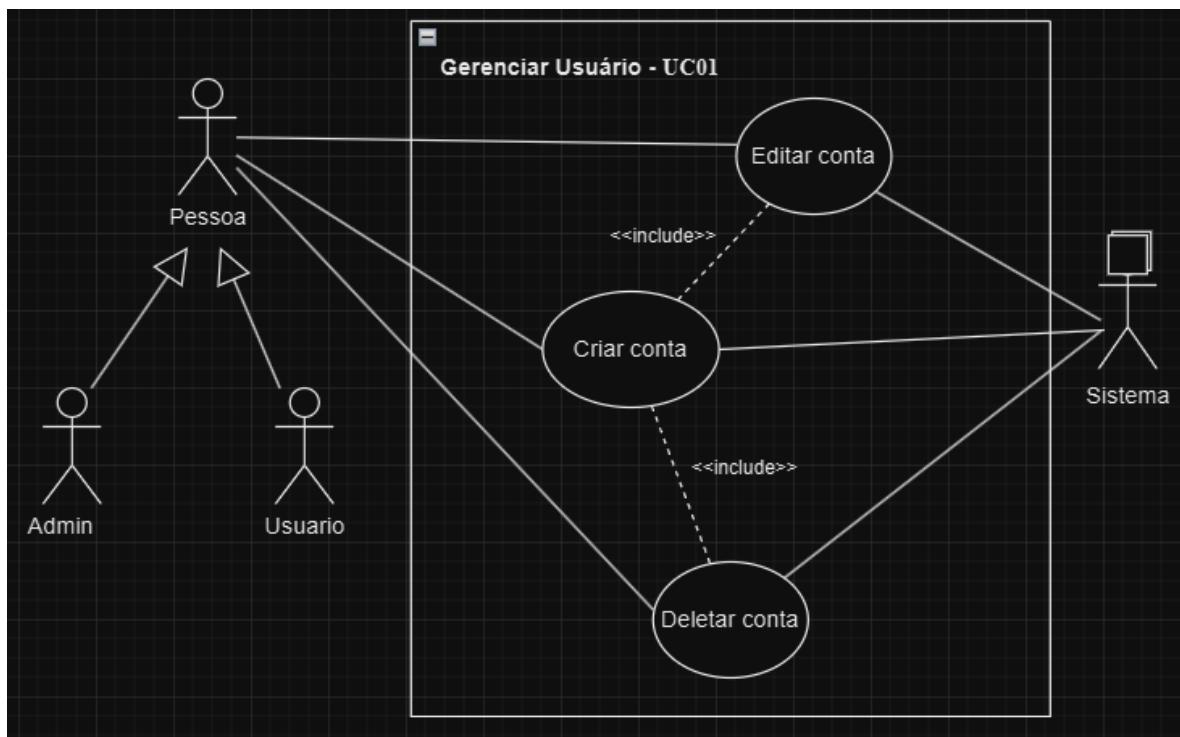
De acordo com a especificação da OMG, "a UML é uma linguagem de modelagem visual unificada que permite aos desenvolvedores especificar, visualizar, construir e documentar os artefatos de um sistema de software" (OMG, 2017).

Diagrama de Casos de Uso

UC01 - Gerenciar Usuário

Nome	Gerenciar usuário.
Descrição	Este caso de uso é responsável pela criação e edição de contas de usuários.
Fluxo de criação	<ol style="list-style-type: none"> 1. O usuário seleciona, na área de login, a opção "criar conta". 2. O usuário preenche as informações requisitadas pelo sistema, como nome da conta, e-mail e senha. 3. O usuário confirma a senha. 4. O usuário aceita os termos de uso. 5. O sistema valida os dados inseridos. 6. O sistema cria a conta do usuário, exibindo uma mensagem de sucesso.
Fluxo de edição	<ol style="list-style-type: none"> 1. O usuário seleciona o menu. 2. O sistema exibe as opções de menu. 3. O usuário seleciona "conta". 4. O sistema mostra as informações da conta. 5. O usuário seleciona a opção "editar conta". 6. O usuário altera as informações permitidas, como nome ou imagem de perfil. 7. O sistema valida as alterações. 8. O sistema atualiza as informações da conta e exibe uma mensagem de sucesso.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema para editar a conta.

Pós-condições	Dados do usuário atualizados ou nova conta criada com sucesso.
Fluxo de Exceção	<ol style="list-style-type: none"> 1. Se o e-mail já estiver em uso, o sistema exibe uma mensagem indicando erro e solicita a correção. 2. Se informações obrigatórias não forem preenchidas, o sistema exibe uma mensagem indicando os campos pendentes. 3. O sistema não executa a ação até que todos os erros sejam corrigidos.

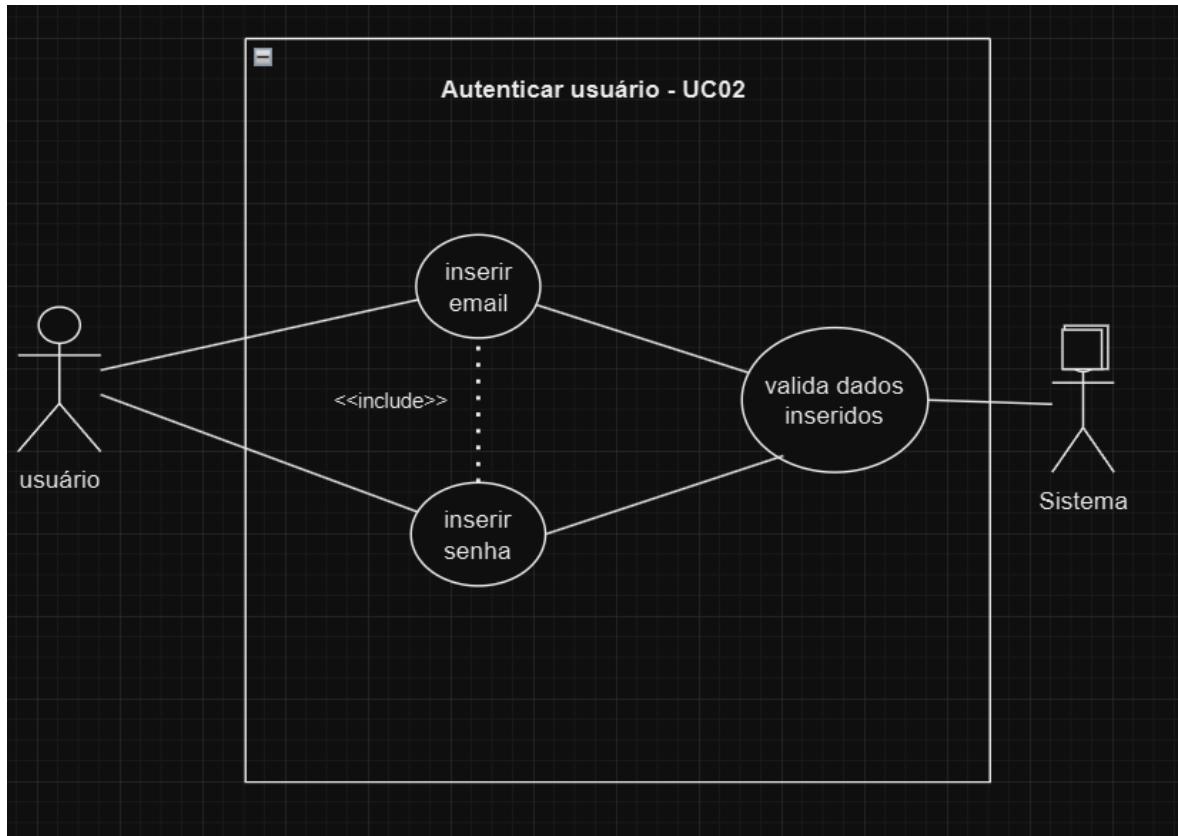


Gerenciar_Usuário

Imagen Caso de Uso UC01

UC02 - Autenticar Usuário

Nome	Autenticar usuário.
Descrição	Este caso de uso é responsável pelo log in da conta de usuário.
Fluxo Entrar	<ol style="list-style-type: none"> 1. O usuário insere seu e-mail na página de login. 2. O usuário insere sua senha na página de login. 3. O sistema valida os dados inseridos. 4. Caso os dados sejam válidos, o usuário é autenticado e redirecionado à página inicial.
Autor	Usuário
Pré-condições	O usuário deve estar registrado no sistema.
Pós-condições	Entrada no sistema.
Fluxo de Exceção	<ol style="list-style-type: none"> 1. Caso o e-mail não esteja registrado, o sistema exibe uma mensagem de erro. 2. Caso a senha esteja incorreta, o sistema exibe uma mensagem indicando erro e solicita nova tentativa.

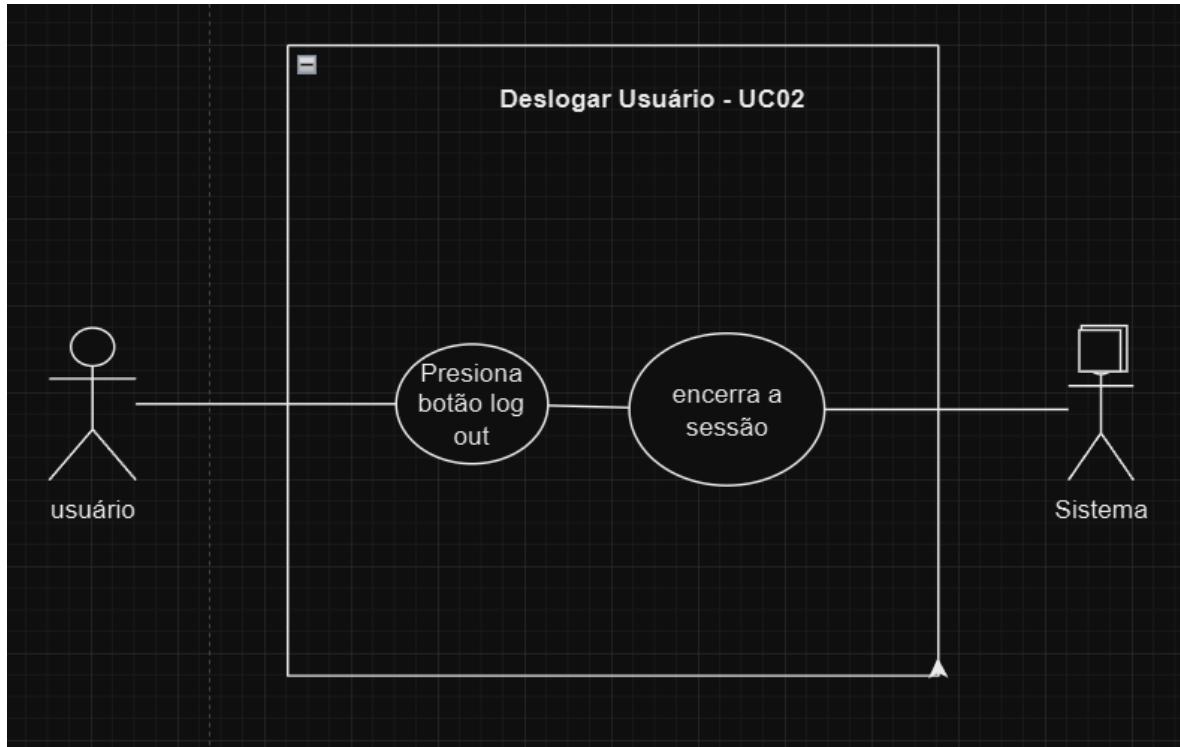


Uc02

Imagen Caso de Uso UC02

UC03 - Deslogar Usuário

Nome	Deslogar usuário.
Descrição	Este caso de uso é responsável pelo log out da conta de usuário.
Fluxo Sair	<ol style="list-style-type: none"> 1. O usuário pressiona no perfil do menu. 2. O usuário pressiona o botão "log out". 3. O sistema exibe um pop-up para confirmação. 4. O usuário confirma o log out. 5. O sistema encerra a sessão e redireciona o usuário para a página de login.
Autor	Usuário
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Saída do sistema.

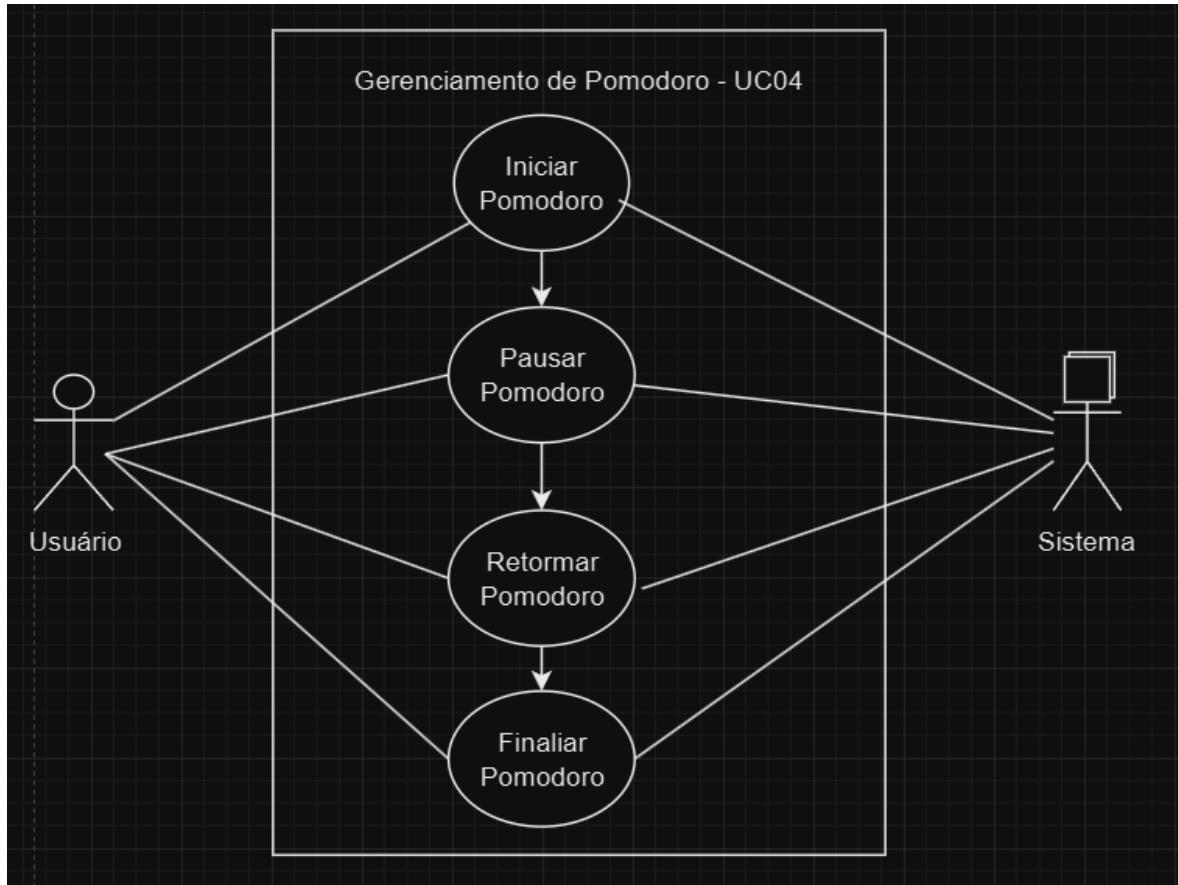


Uc03

Imagen Caso de Uso UC03

UC04 - Gerenciamento de Pomodoro

Nome	Gerenciamento de Pomodoro
Descrição	Este caso de uso descreve as funcionalidades de ativação, desativação, pausa e retomada do temporizador de Pomodoro no sistema.
Fluxo de ativação	<ol style="list-style-type: none"> 1. O usuário acessa a funcionalidade de Pomodoro no sistema. 2. O sistema verifica se o usuário está autenticado e tem acesso à funcionalidade. 3. O sistema exibe a interface do temporizador de Pomodoro. 4. O usuário clica no botão "Iniciar Pomodoro". 5. O sistema ativa o temporizador, definindo a duração padrão (ex.: 25 minutos). 6. O sistema inicia a contagem regressiva e exibe o tempo restante. 7. Ao atingir o final do ciclo, o sistema emite um aviso sonoro. 8. O sistema exibe uma mensagem informando a conclusão.
Fluxo de pausa	<ol style="list-style-type: none"> 1. Durante um ciclo ativo, o usuário clica no botão "Pausar Pomodoro". 2. O sistema interrompe a contagem regressiva. 3. O sistema altera o estado para "Pausado". 4. O usuário pode retomar ou parar o ciclo.
Fluxo de retomada	<ol style="list-style-type: none"> 1. Durante um ciclo pausado, o usuário clica no botão "Retomar Pomodoro". 2. O sistema verifica se há um ciclo em estado "Pausado". 3. O sistema retoma a contagem regressiva. 4. O sistema atualiza a interface. 5. Ao atingir o final do ciclo, o sistema emite um aviso sonoro e exibe uma mensagem informando a conclusão.

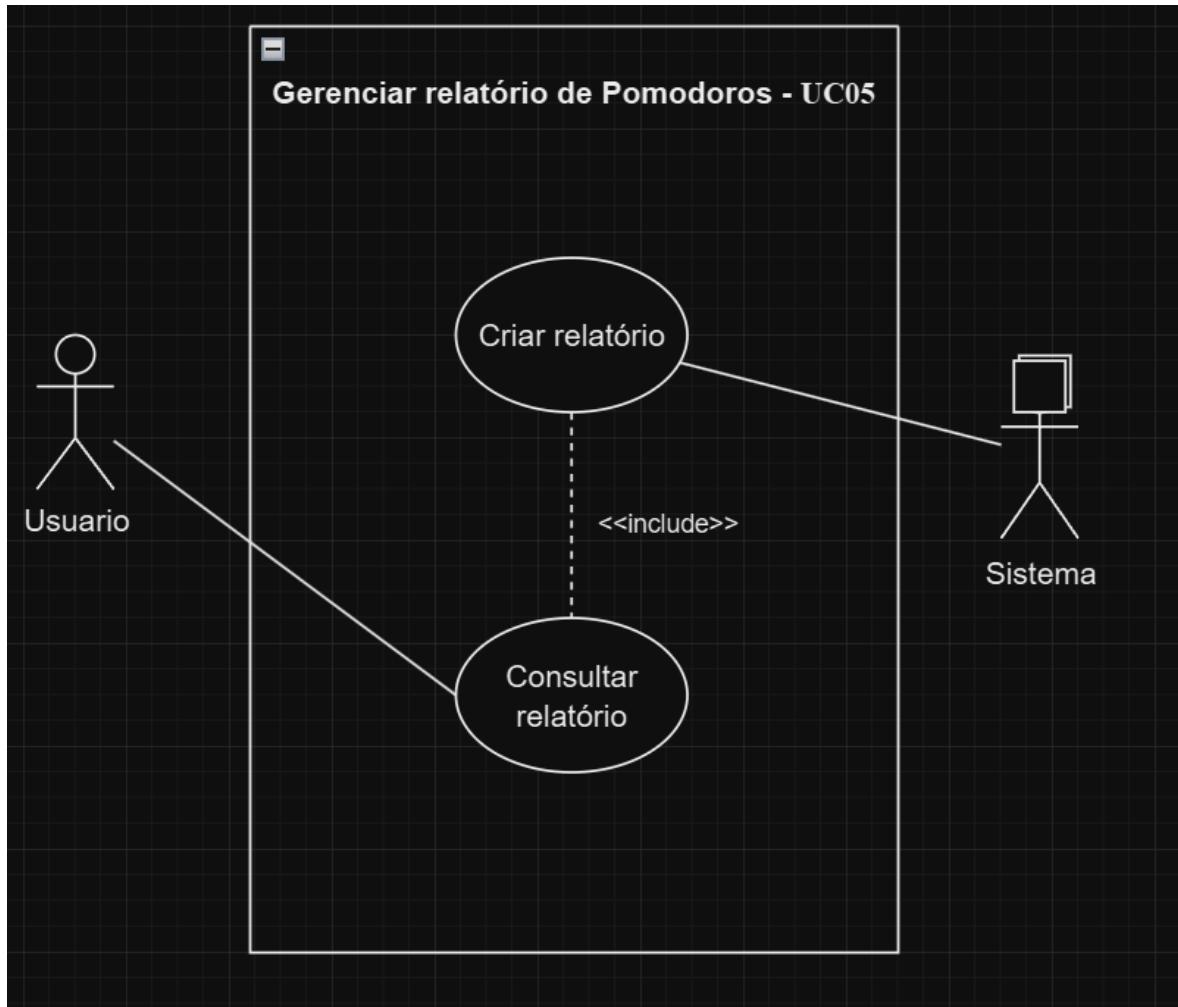


Uc04

Imagen Caso de Uso UC04

UC05 - Gerenciar Relatório de Pomodoros

Nome	Gerenciar Relatório de Pomodoros
Descrição	Este caso de uso é responsável pela criação e consulta de relatórios dos Pomodoros.
Fluxo de criação	<ol style="list-style-type: none"> 1. O usuário seleciona a opção “criar relatório de Pomodoro”. 2. O usuário informa o período desejado (datas de início e fim). 3. O sistema gera o relatório com base nos dados. 4. O sistema valida os dados e exibe o relatório gerado. 5. O usuário confirma o relatório.
Fluxo de consulta	<ol style="list-style-type: none"> 1. O usuário seleciona a opção “consultar relatório de Pomodoro”. 2. O usuário informa o período desejado ou escolhe relatórios existentes. 3. O sistema exibe o relatório solicitado. 4. O usuário visualiza o relatório.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Relatório de Pomodoros criado ou consultado.



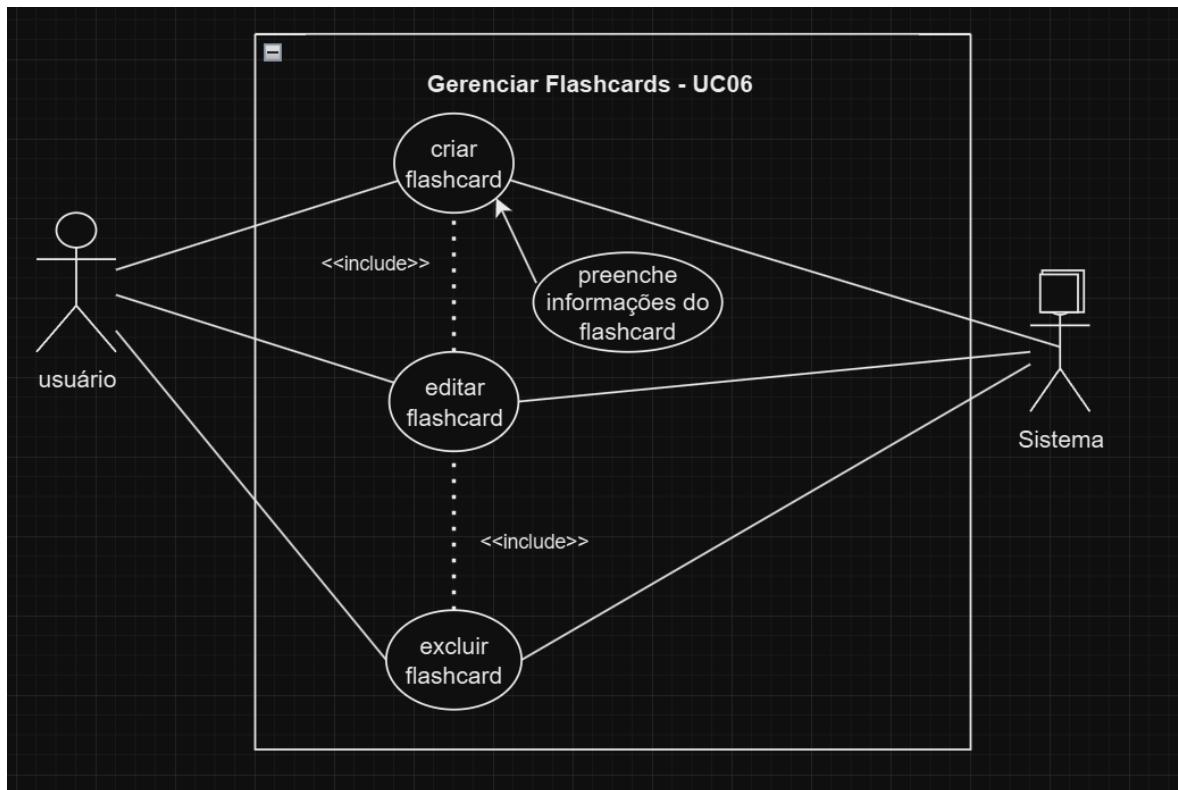
Uc05

Imagen Caso de Uso UC05

UC06 - Gerenciar Flashcards

Nome	Gerenciar Flashcards
Descrição	Este caso de uso é responsável pela criação, edição e consulta de flash cards do usuário.
Fluxo de criação	<ol style="list-style-type: none"> 1. O usuário seleciona a opção de adicionar flashcards “+” logo abaixo do Pomodoro. 2. O sistema exibe um formulário para que o usuário preencha as informações do novo flashcard. 3. O usuário preenche as informações solicitadas. 4. O usuário clica no botão "Salvar". 5. O sistema valida as informações. 6. Caso válidas, o sistema salva o novo flashcard. 7. O sistema exibe uma mensagem de sucesso. 8. O sistema atualiza a lista de flashcards do usuário.
Fluxo de edição	<ol style="list-style-type: none"> 1. O usuário seleciona os três pontos do flashcard. 2. O sistema exibe as opções de favoritar, editar ou excluir. 3. O usuário seleciona a opção “editar”. 4. O usuário preenche os novos dados do flashcard. 5. O sistema valida as informações. 6. Caso válidas, o flashcard é atualizado no banco de dados. 7. O sistema exibe uma mensagem de sucesso. 8. O sistema atualiza a lista de flashcards do usuário.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.

Pós-condições	Alteração da lista de flashcards do usuário.
---------------	--

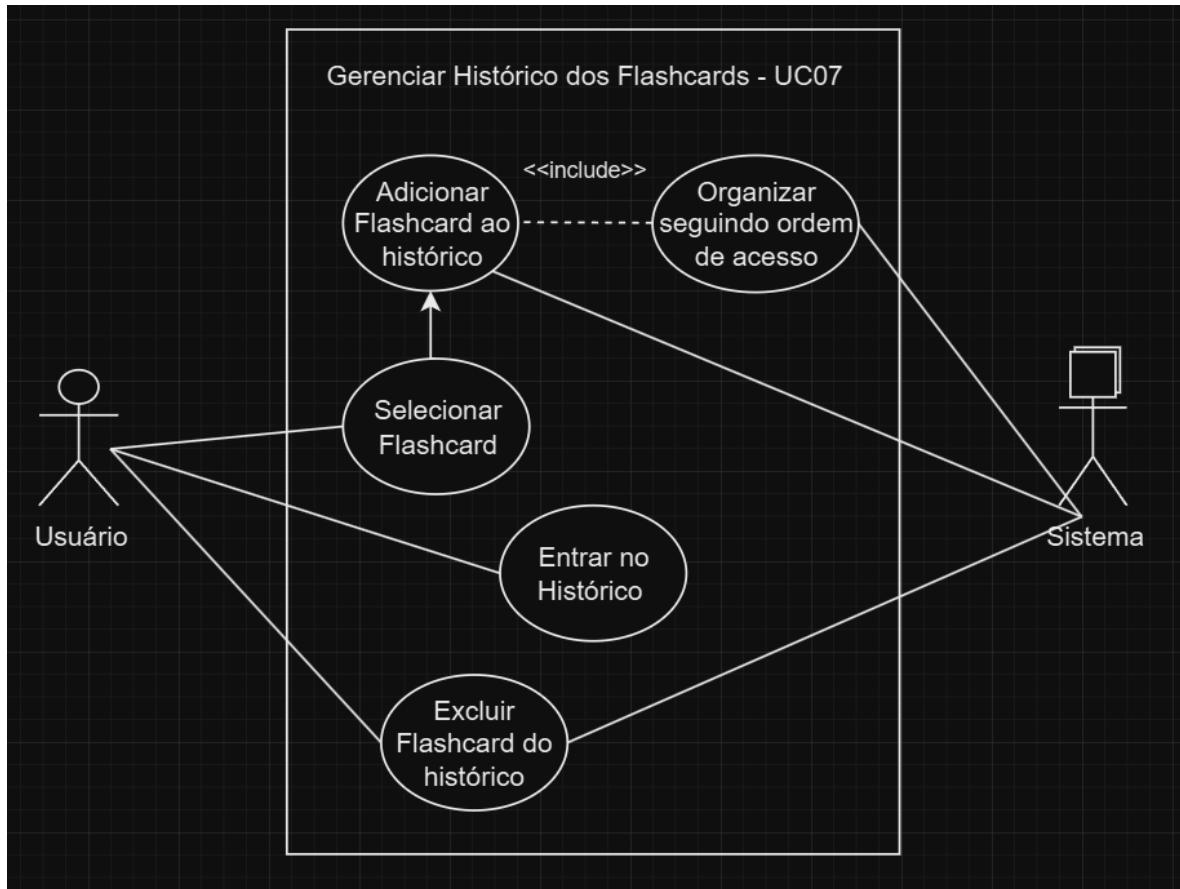


Uc06

Imagen Caso de Uso UC06

UC07 - Gerenciar Histórico dos Flashcards

Nome	Gerenciar Histórico dos Flashcards
Descrição	Este caso de uso é responsável pela adição e remoção de flashcards do histórico.
Fluxo principal	<ol style="list-style-type: none"> 1. O usuário seleciona um flashcard. 2. O sistema adiciona o flashcard no histórico. 3. O sistema organiza os flashcards conforme a ordem de acesso.
Fluxo de reação	<ol style="list-style-type: none"> 1. O usuário seleciona a opção “menu”. 2. O sistema exibe as opções de menu. 3. O usuário seleciona a opção “flashcards”. 4. O usuário acessa a interface de gerenciamento de flashcards. 5. O usuário seleciona a opção “Histórico”. 6. O usuário seleciona o flashcard correspondente. 7. O usuário seleciona a opção “Remover do Histórico”. 8. O sistema exibe uma mensagem de confirmação. 9. Caso o usuário confirme, o sistema remove o flashcard do histórico. Caso contrário, volta à tela anterior.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Alteração no histórico de flashcards do usuário.



Uc07

Imagen Caso de Uso UC07

UC08 - Gerenciar Favoritos

Nome	Gerenciar Favoritos
Descrição	Este caso de uso é responsável pela adição e remoção de flashcards dos favoritos.
Fluxo de favoritar	<ol style="list-style-type: none"> 1. O usuário seleciona os três pontos no quadrado do flashcard específico. 2. O sistema exibe as opções de favoritar, editar ou excluir. 3. O usuário seleciona a opção “favoritar”.
Fluxo de desfavaritar	<ol style="list-style-type: none"> 1. O usuário seleciona os três pontos no quadrado do flashcard específico. 2. O sistema exibe as opções de favoritar, editar ou excluir. 3. O usuário seleciona a opção “desfavoritar”.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Alteração na lista de favoritos do usuário.

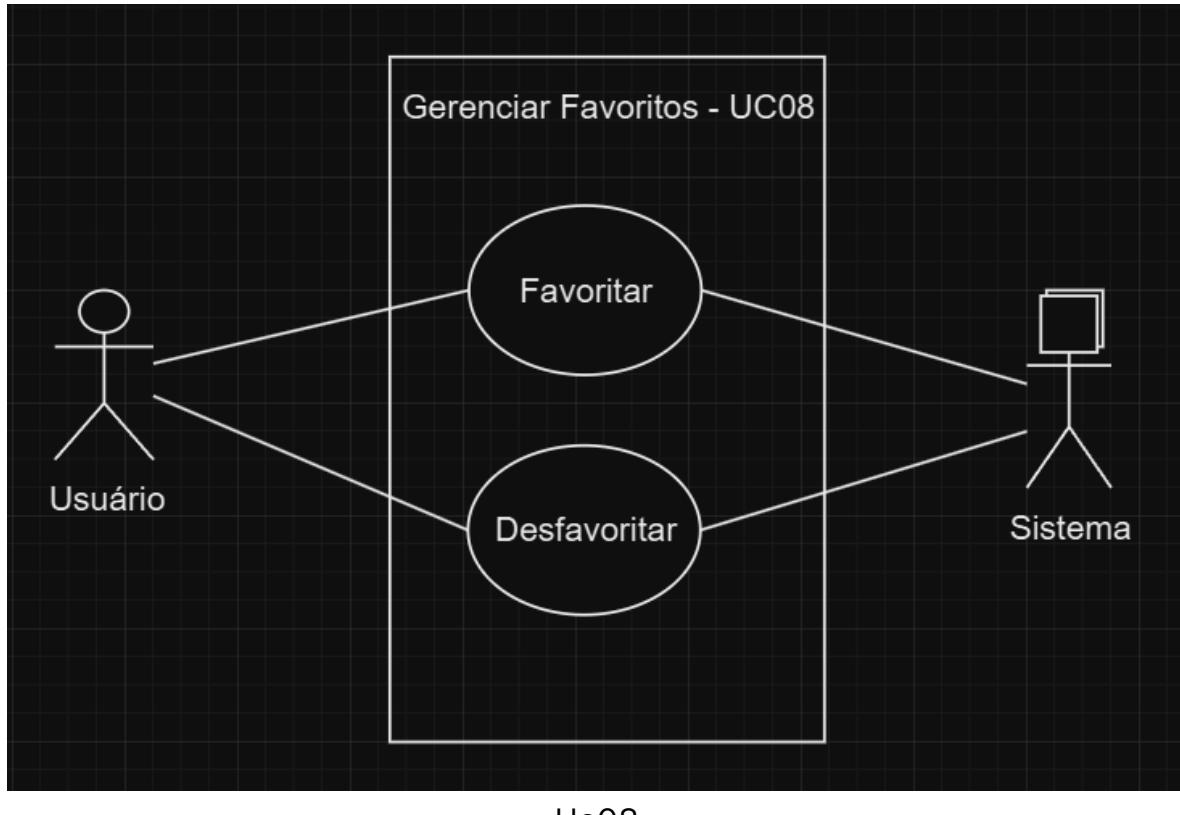


Imagen Caso de Uso UC08

UC09 - Gerenciar Conjuntos de Flashcards

Nome	Gerenciar Conjuntos
Descrição	Este caso de uso é responsável pela criação, consulta, edição e exclusão das categorias de flashcards.
Fluxo de criação	<ol style="list-style-type: none"> 1. O usuário seleciona a opção de criar conjunto “+ com uma pasta” logo abaixo do Pomodoro. 2. O sistema exibe um formulário para que o usuário preencha o nome, descrição e cor do novo conjunto. 3. O usuário preenche os requisitos. 4. O usuário confirma a criação do conjunto. 5. O sistema valida as informações. 6. O sistema atualiza as informações e adiciona o novo conjunto.
Fluxo de consulta	<ol style="list-style-type: none"> 1. O usuário seleciona a lupa na parte dos conjuntos. 2. O sistema abre um campo de texto para escrita. 3. O usuário digita o nome do conjunto desejado. 4. O usuário confirma a pesquisa. 5. O sistema mostra os conjuntos que correspondem à pesquisa.
Fluxo de edição	<ol style="list-style-type: none"> 1. O usuário seleciona a opção “ver mais” logo abaixo dos conjuntos. 2. O sistema exibe um modal com todos os conjuntos. 3. O usuário seleciona os três pontos de um conjunto. 4. O sistema exibe as opções de editar ou excluir. 5. O usuário escolhe “editar”. 6. O sistema exibe o conjunto com opções para alterar flashcards, nome, descrição ou cor. 7. O usuário confirma as alterações. 8. O sistema valida e atualiza o conjunto no banco de dados.

Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Criar, consultar, editar ou excluir as categorias dos flashcards.

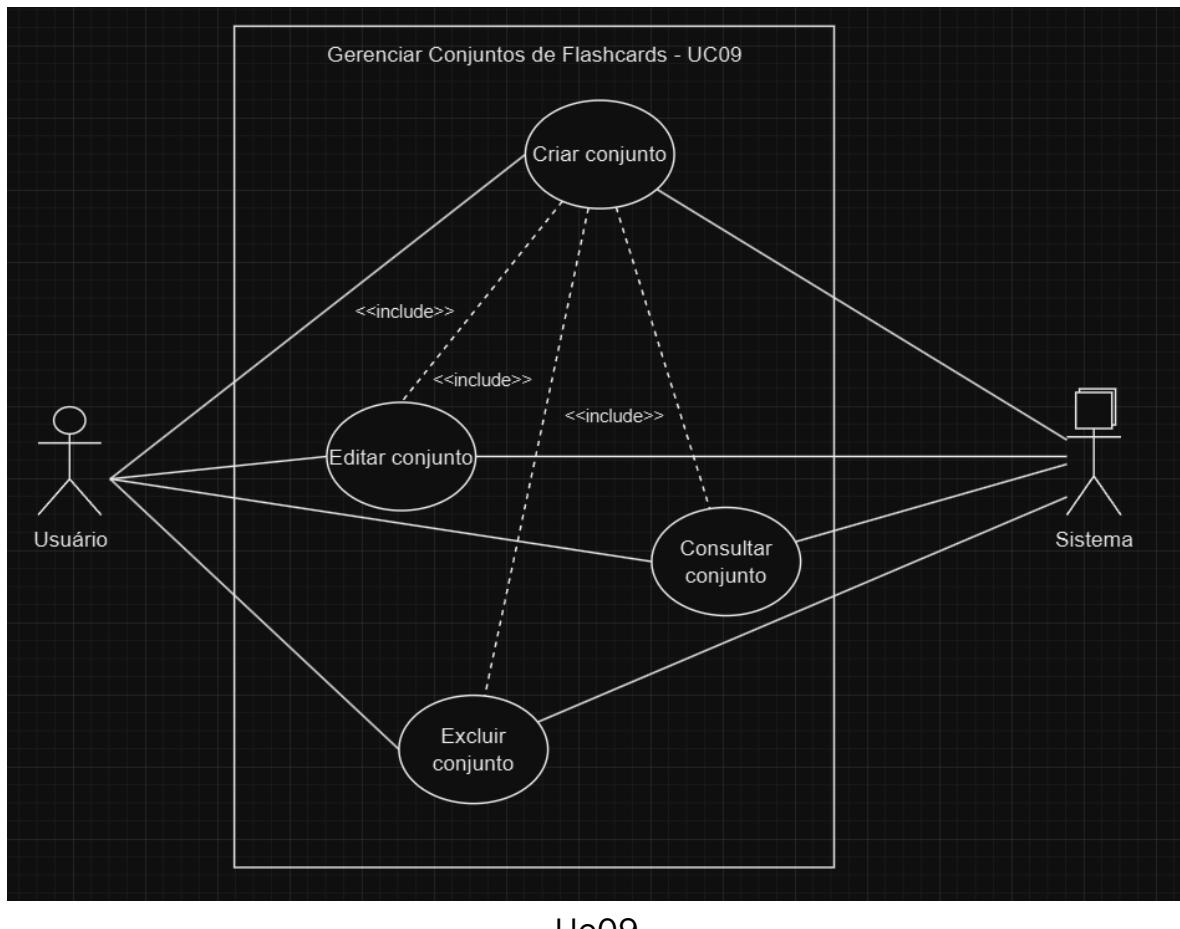
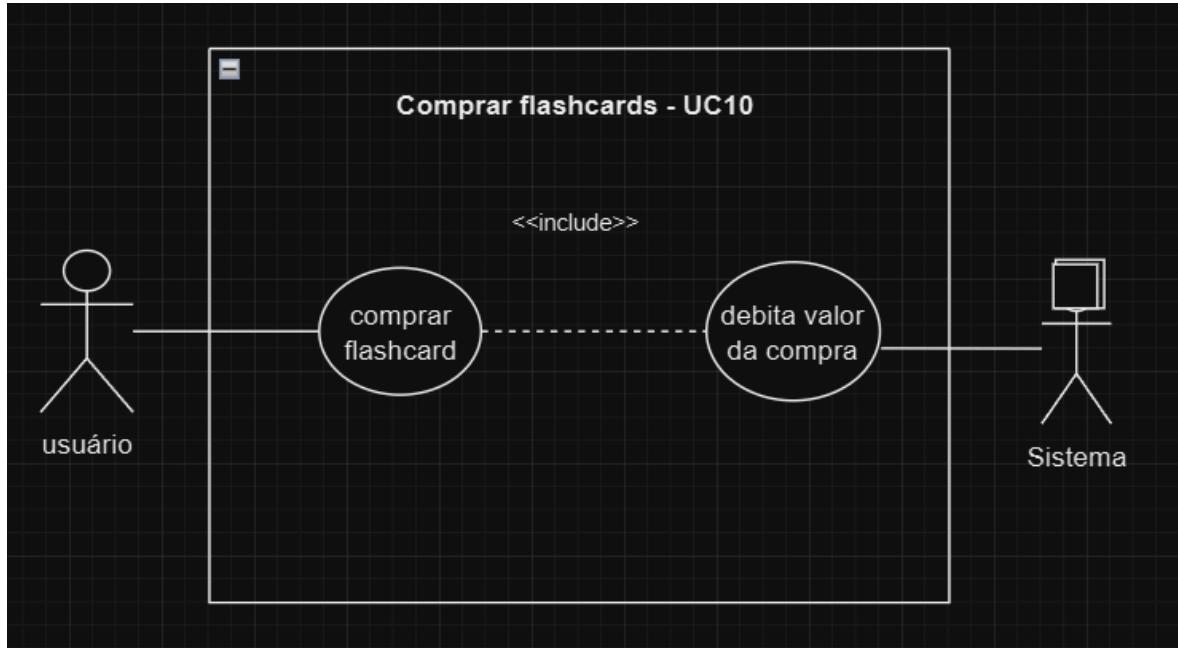


Imagen Caso de Uso UC09

UC10 - Comprar Flashcards

Nome	Comprar Flashcards
Descrição	Este caso de uso é responsável pela compra de flashcards usando a moeda do sistema.
Fluxo de Compra	<ol style="list-style-type: none"> 1. O usuário acessa a interface da loja. 2. O sistema exibe uma lista de flashcards disponíveis para compra, com filtros por categoria e preço. 3. O usuário seleciona o flashcard desejado. 4. O sistema verifica se o usuário possui saldo suficiente. 5. Caso tenha saldo, o sistema exibe uma mensagem de confirmação. 6. O usuário confirma a compra. 7. O sistema atualiza a lista de flashcards do usuário e débita o saldo da conta. 8. O sistema exibe o flashcard comprado ao usuário.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema e possuir saldo suficiente.
Pós-condições	Alteração no saldo e atualização da lista de flashcards do usuário.

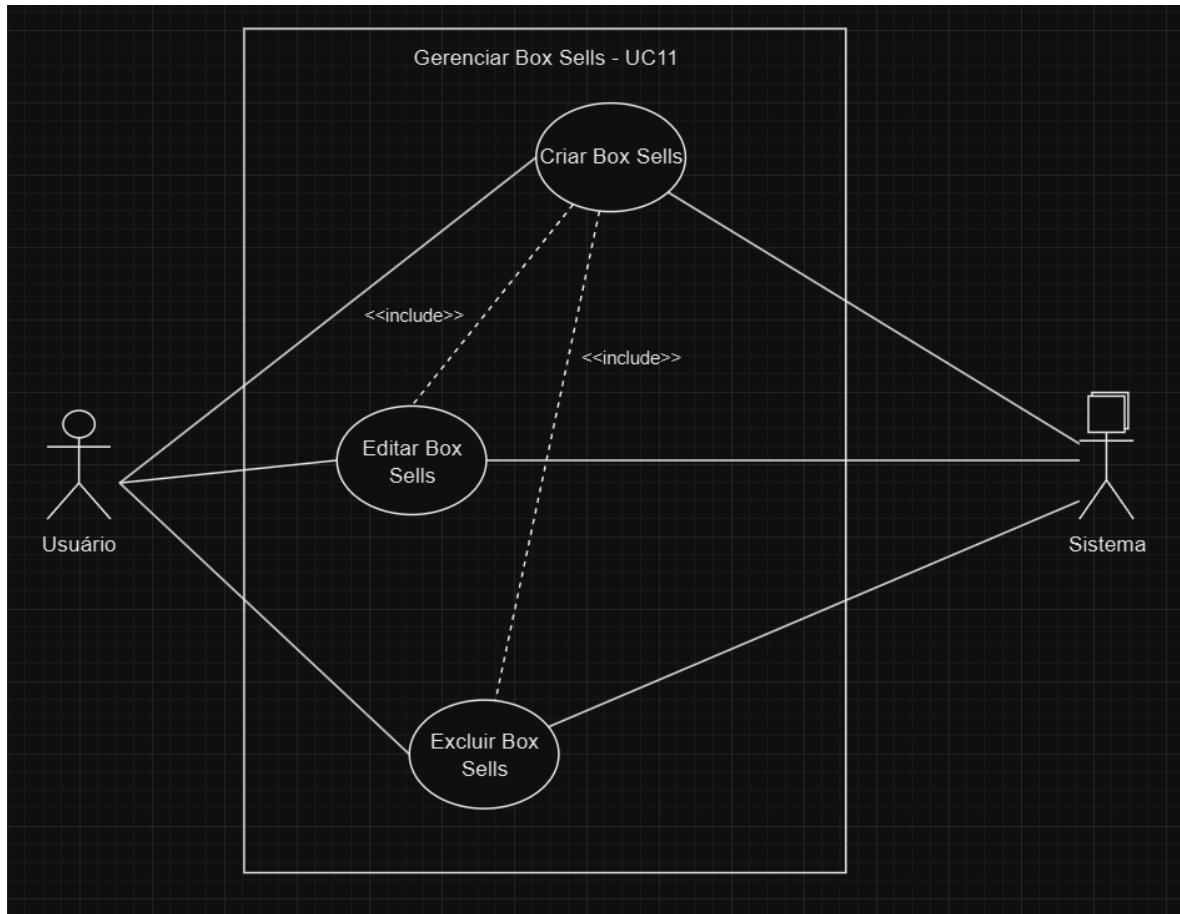


Uc10

Imagen Caso de Uso UC10

UC11 - Gerenciar Box Sells

Nome	Gerenciamento de Box Sells
Descrição	Este caso de uso é responsável pela criação, edição e exclusão de Box Sells.
Fluxo de criação	<ol style="list-style-type: none"> 1. O usuário acessa a interface de gerenciamento de Box Sells. 2. O usuário seleciona a opção “Criar Box Sell”. 3. O sistema solicita ao usuário que defina um nome e uma descrição. 4. O usuário escolhe os flashcards a serem incluídos. 5. O sistema exibe um resumo da Box Sell com os flashcards selecionados. 6. O usuário define o preço de venda. 7. O usuário confirma a criação. 8. O sistema adiciona a Box Sell à loja e à lista de vendas disponíveis.
Fluxo de edição	<ol style="list-style-type: none"> 1. O usuário acessa a interface de gerenciamento de Box Sells. 2. O usuário seleciona a Box Sell que deseja editar. 3. O sistema exibe os detalhes da Box Sell. 4. O usuário altera nome, descrição ou flashcards incluídos. 5. O usuário confirma as alterações. 6. O sistema valida e atualiza a Box Sell no banco de dados.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema e ter permissões para gerenciar Box Sells.
Pós-condições	Alteração na lista de Box Sells disponíveis e no banco de dados.

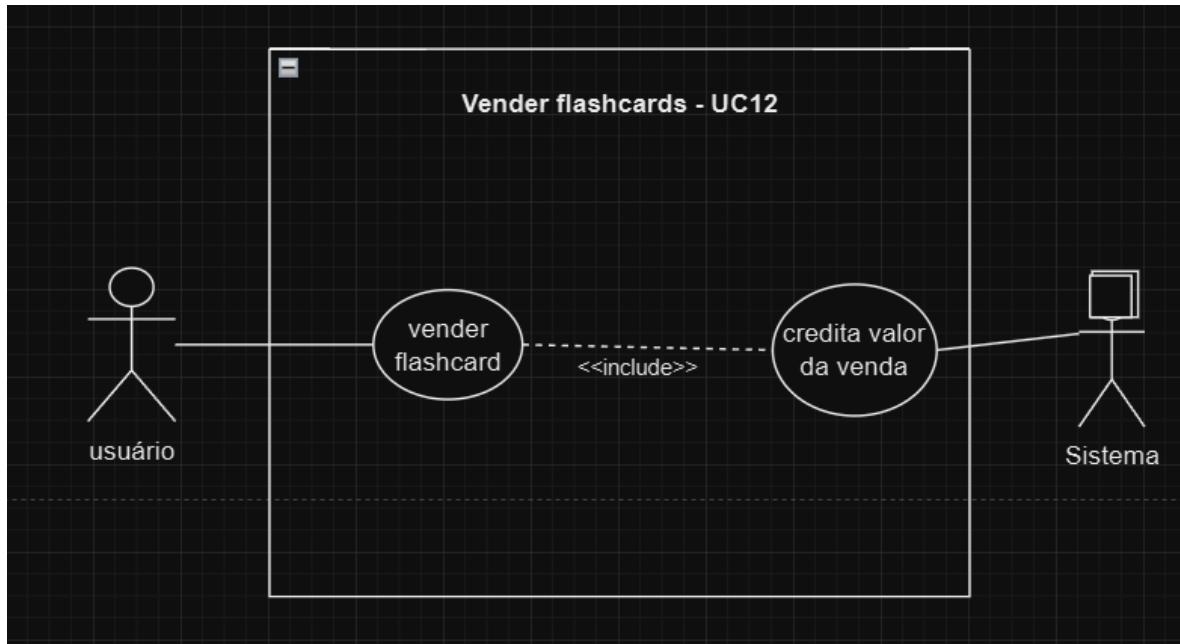


Uc11

Imagen Caso de Uso UC11

UC12 - Vender Flashcards

Nome	Vender Flashcards
Descrição	Este caso de uso é responsável pela venda de flashcards usando a moeda do sistema.
Fluxo de Venda	<ol style="list-style-type: none"> 1. O usuário acessa a interface da loja. 2. O usuário seleciona a opção “Venda de Flashcards”. 3. O usuário escolhe o flashcard a ser vendido. 4. O usuário define o valor da venda em moedas do sistema. 5. O usuário confirma a venda. 6. O sistema lista o flashcard para outros usuários. 7. Quando vendido, o sistema credita o valor na conta do usuário e remove o flashcard.
Autor	Usuário e Sistema
Pré-condições	O usuário deve estar logado no sistema.
Pós-condições	Alteração no saldo e na lista de flashcards do usuário.



Uc12

Imagen Caso de Uso UC12

Diagrama de Classes

O diagrama de classes representa a estrutura do sistema, detalhando as principais entidades e seus relacionamentos. Cada classe é projetada para capturar um conjunto de informações sobre um conceito específico, e os métodos listados permitem a manipulação e o acesso a esses dados.

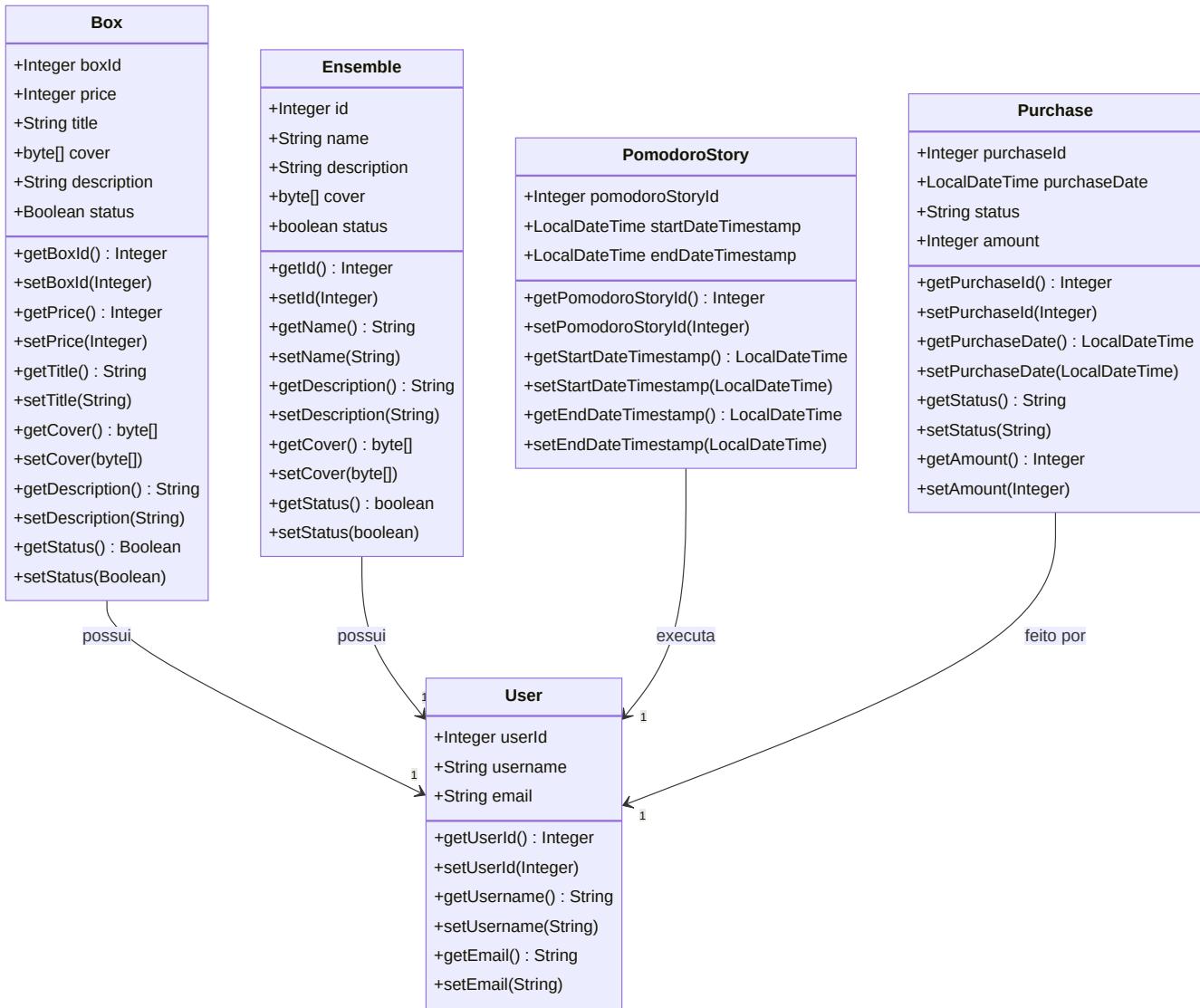
O diagrama de classes representa a estrutura do sistema, detalhando as principais entidades e seus relacionamentos. Cada classe é projetada para capturar um conjunto de informações sobre um conceito específico, e os métodos listados permitem a manipulação e o acesso a esses dados.

- **Box:** Representa uma "caixa" no sistema, com atributos como preço, título, descrição e imagem de capa. Um usuário pode possuir várias caixas, e os métodos fornecem acesso e modificação dos atributos da classe, como `getPrice()`, `setPrice()`, `getTitle()`, entre outros.
- **Ensemble:** Representa um "conjunto" com atributos como nome, descrição e imagem de capa. A classe tem métodos para acessar e modificar esses atributos e está associada a um único usuário, que pode possuir vários ensembles.

- **PomodoroStory:** Reflete uma história de sessão Pomodoro, contendo os timestamps de início e fim. Um usuário pode realizar várias sessões Pomodoro, e os métodos `getStartDateTimestamp()` e `setStartDateTimestamp()` permitem acessar e definir esses valores.
- **Purchase:** Relacionada às compras feitas pelos usuários, com informações sobre o valor, data e status da compra. A classe também permite acessar e modificar esses atributos, como `getAmount()`, `setAmount()`, `getStatus()`, e está associada a um único usuário.
- **User:** Representa o usuário do sistema, com atributos como `userId`, `username` e `email`. Ele pode estar relacionado a várias instâncias das outras classes (Box, Ensemble, PomodoroStory e Purchase).

Relacionamentos:

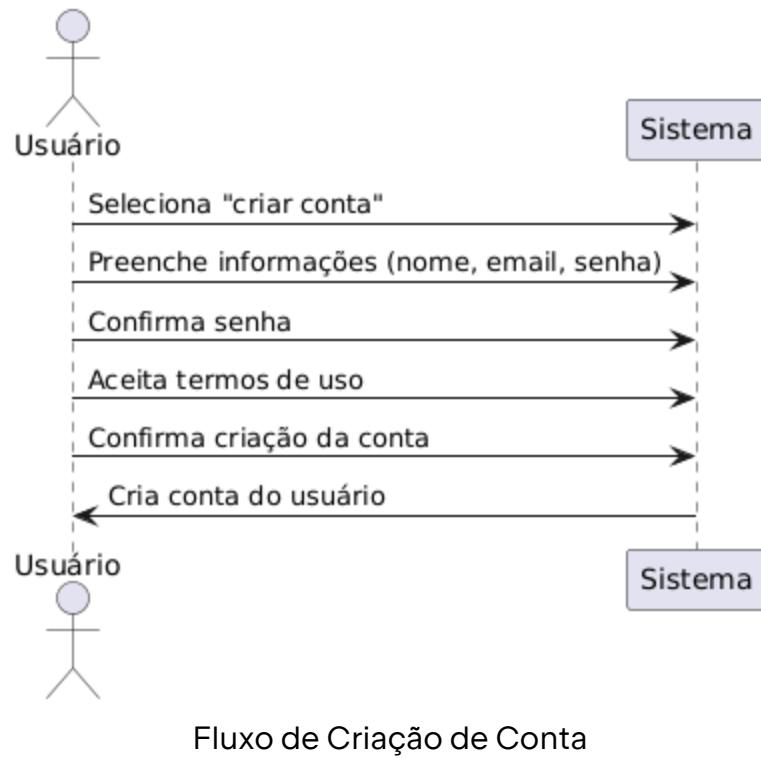
- **Box** e **Ensemble** possuem um relacionamento de **muitos para um** com **User**, ou seja, um usuário pode ter várias caixas e ensembles.
- **PomodoroStory** e **Purchase** também possuem um relacionamento de **muitos para um** com **User**, representando as sessões Pomodoro e as compras feitas por um usuário.



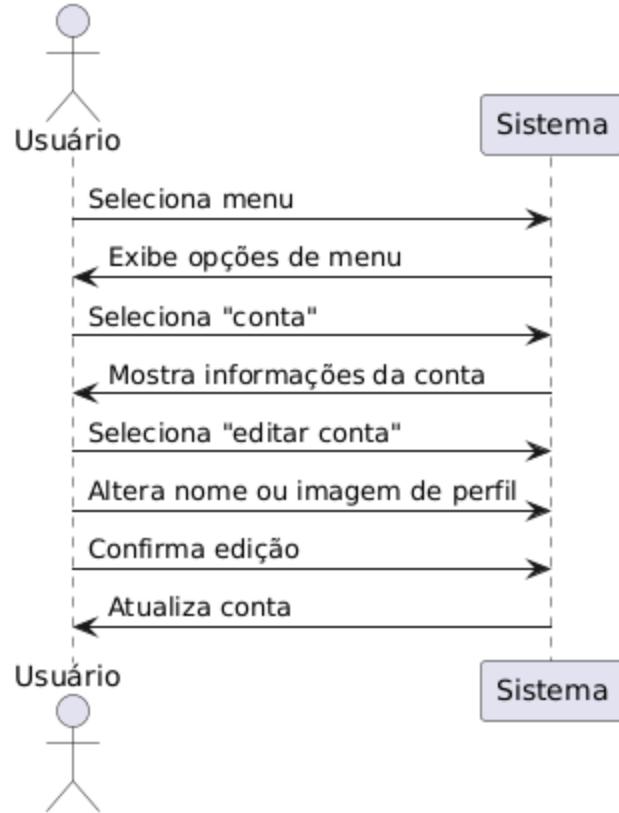
Diagramas de Sequência

Diagrama de Sequencia UC-01

Criação da conta



Edição da conta



Fluxo de Edição de Conta

Exclusão da conta

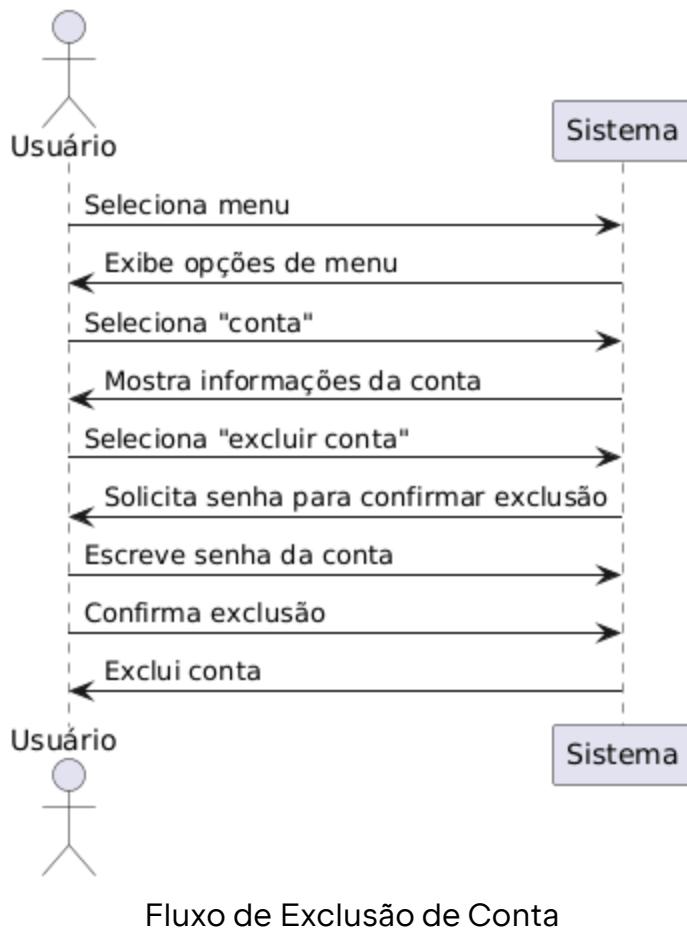
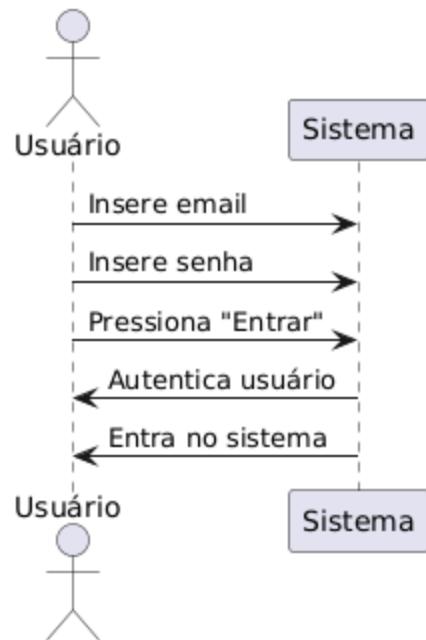


Diagrama de Sequencia : UC-02

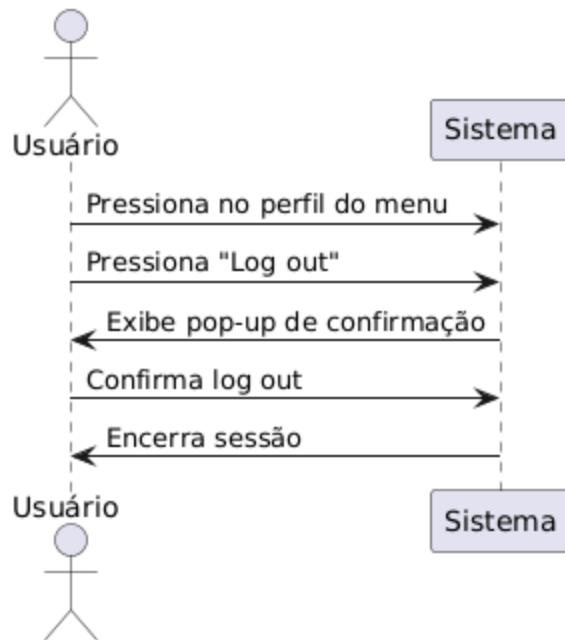
Entrada no Sistema



Fluxo de Entrar do Sistema

Diagrama de Sequencia : UC-03

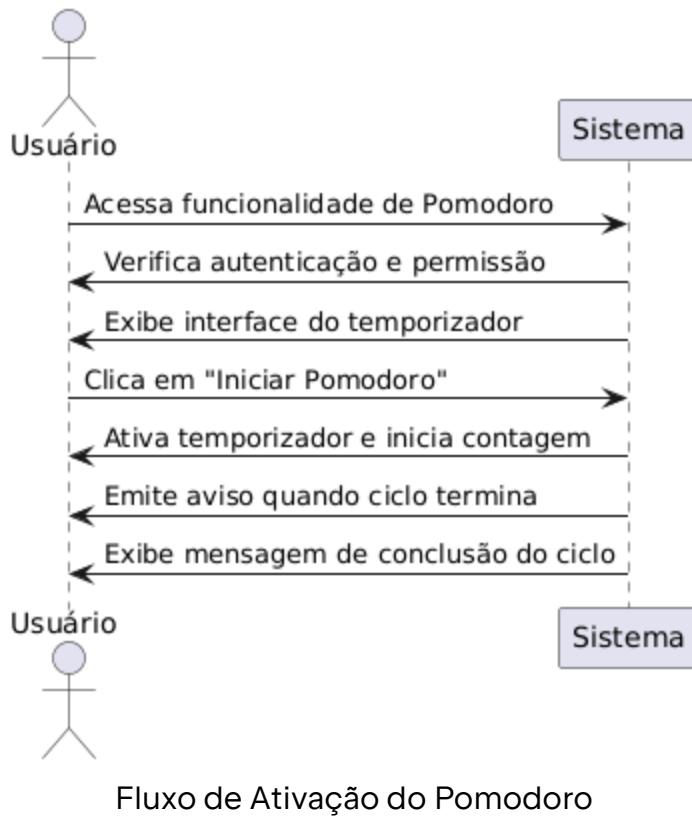
Saida do Sistema



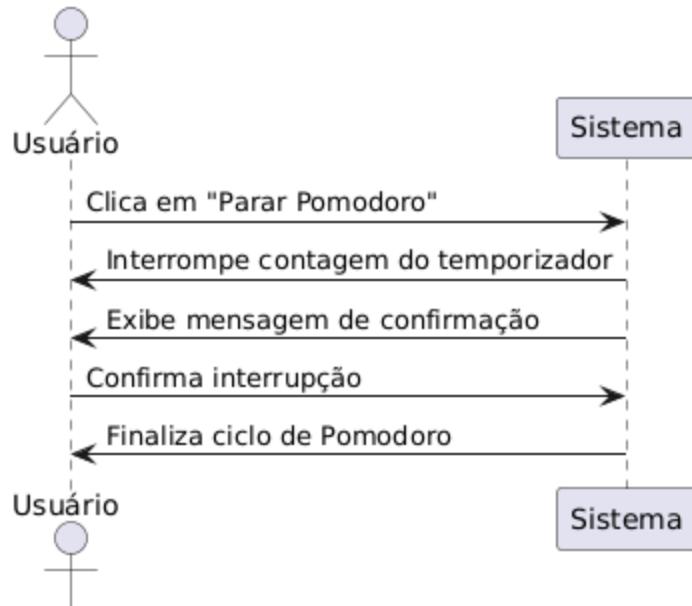
Fluxo de Saida do Sistema

Diagrama de Sequencia : UC-04

Ativação do Pomodoro

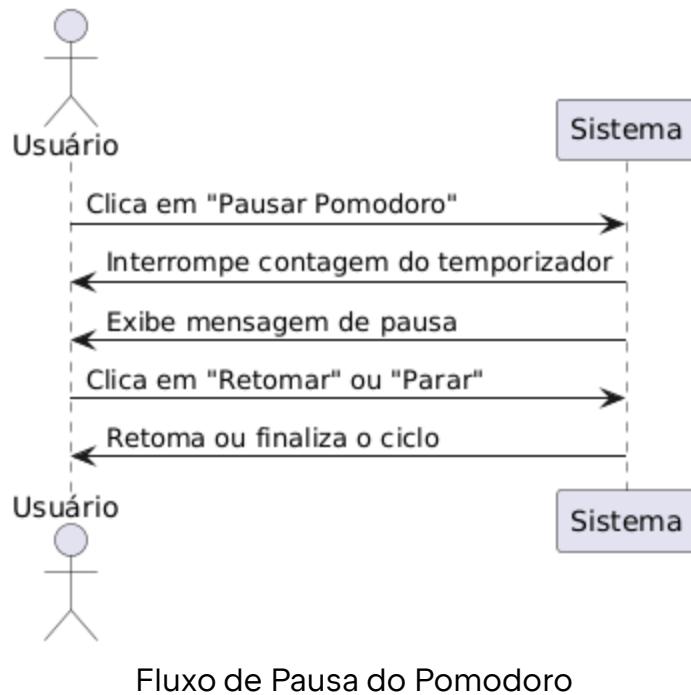


Desativação do Pomodoro



Fluxo de Desativação do Pomodoro

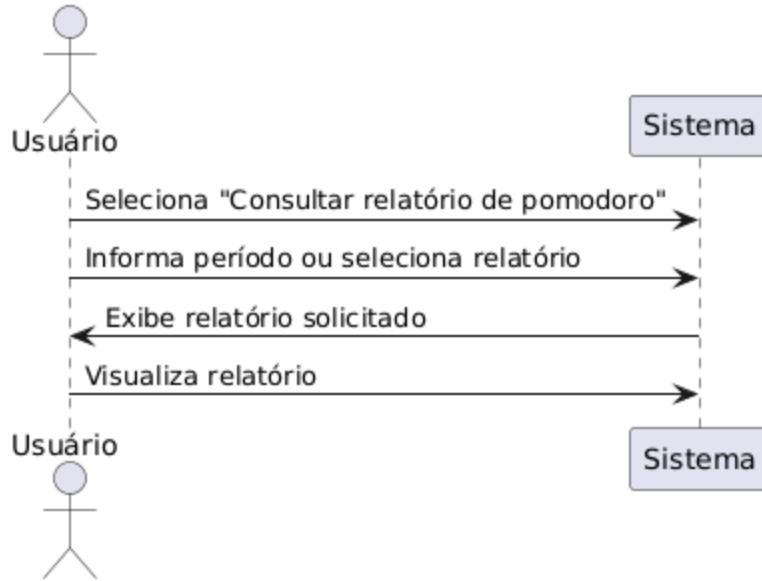
Pausa do Pomodoro



Fluxo de Pausa do Pomodoro

Diagrama de Sequencia : UC-05

Consulta do Relatórios



Fluxo de Consulta dos Relatório dos Pomodoros

Criação dos Relatórios

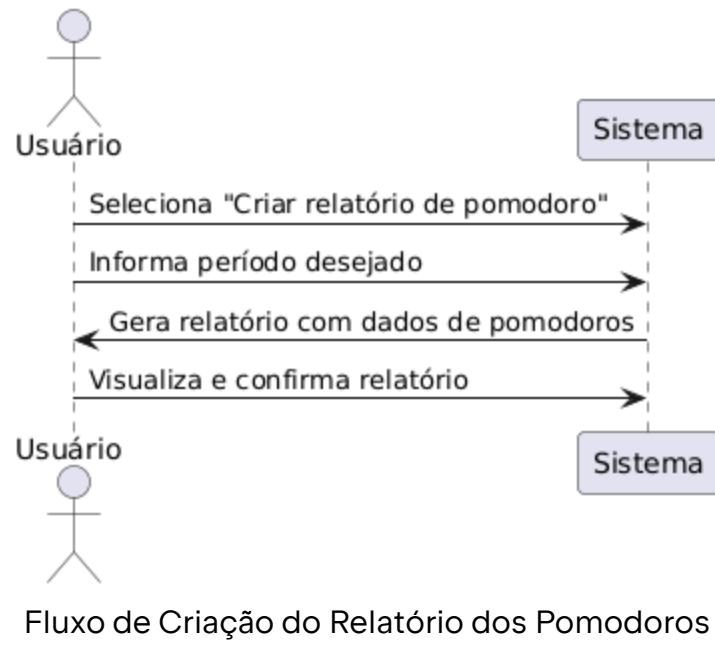
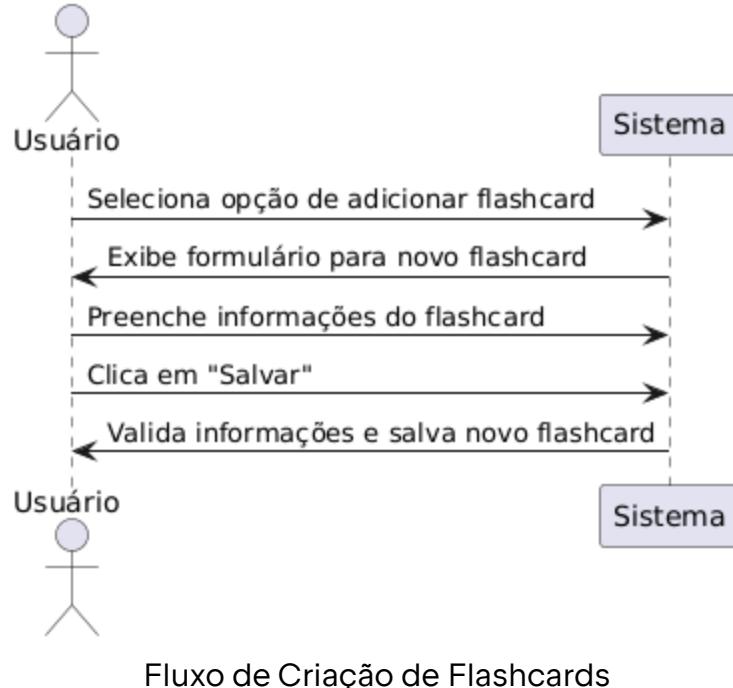
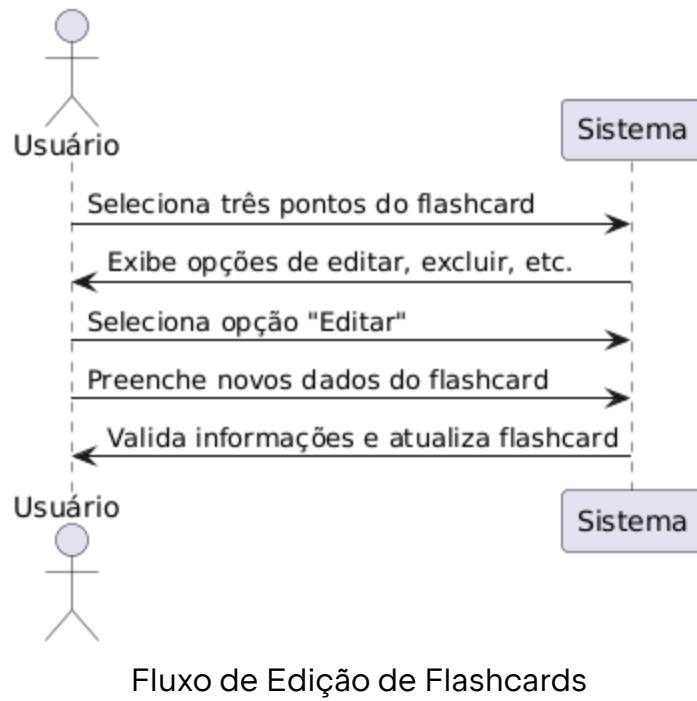


Diagrama de Sequencia : UC-06

Criação dos Flashcards



Edição dos Flashcards



Exclusão dos Flashcards

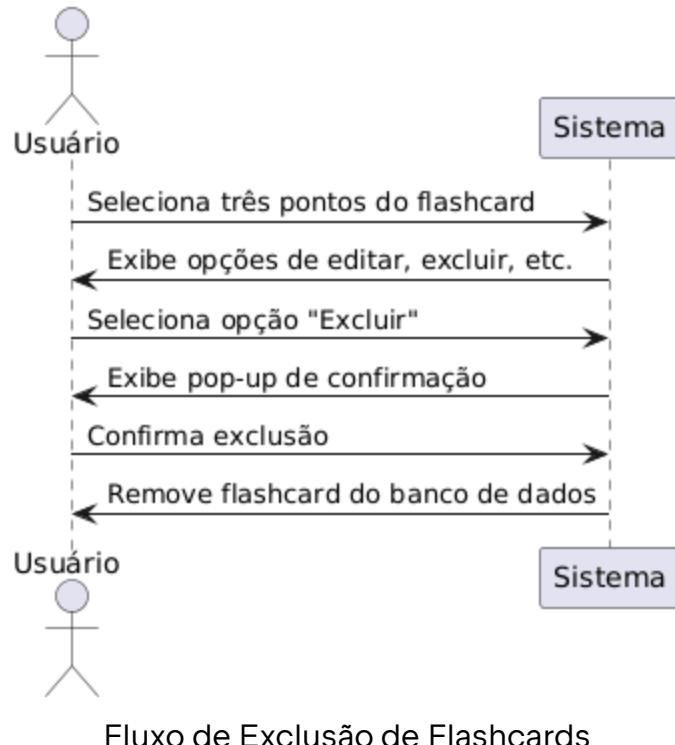
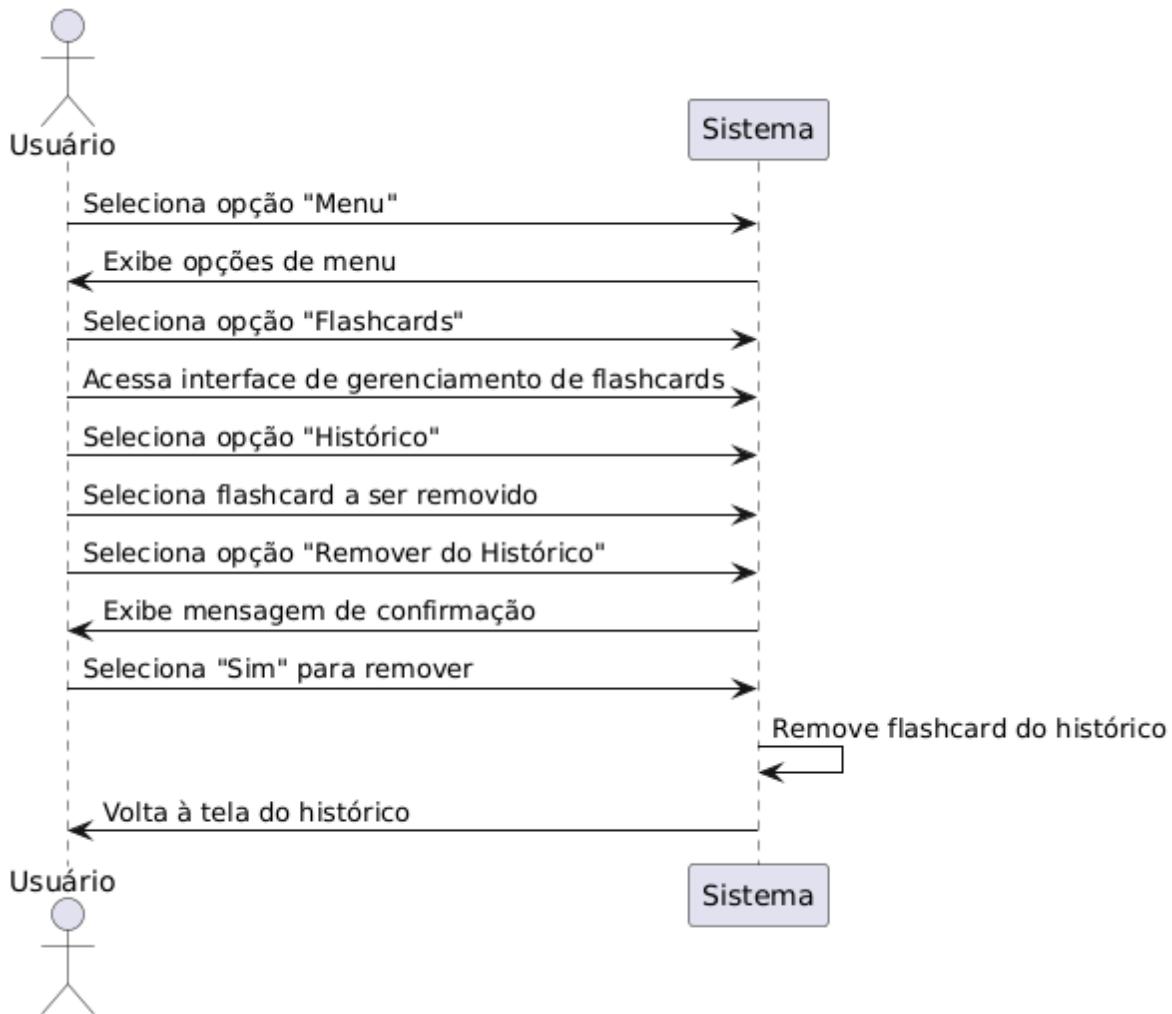


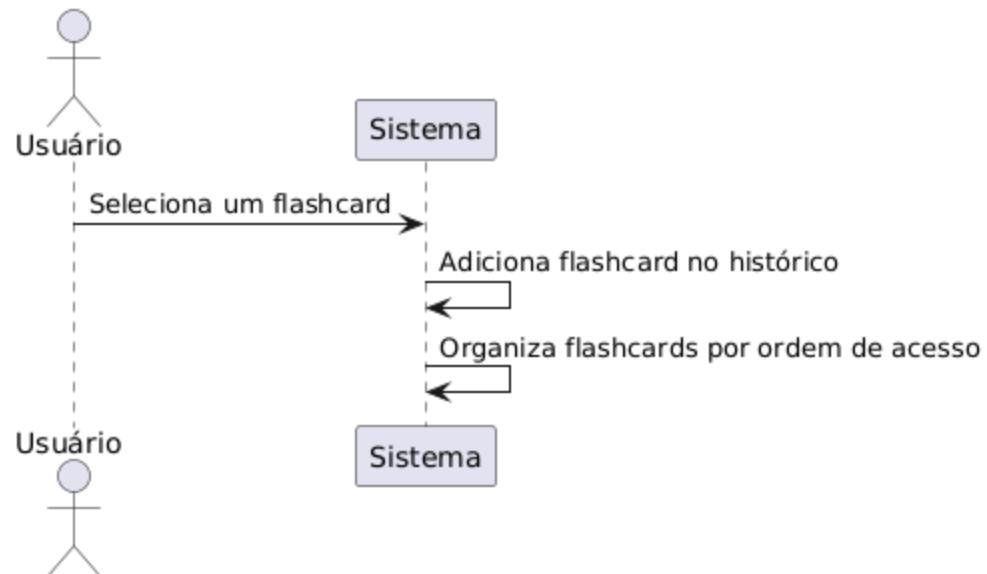
Diagrama de Sequencia : UC-07

Remoção do Histórico dos Flashcards



Fluxo de Remoção de um Log do Histórico

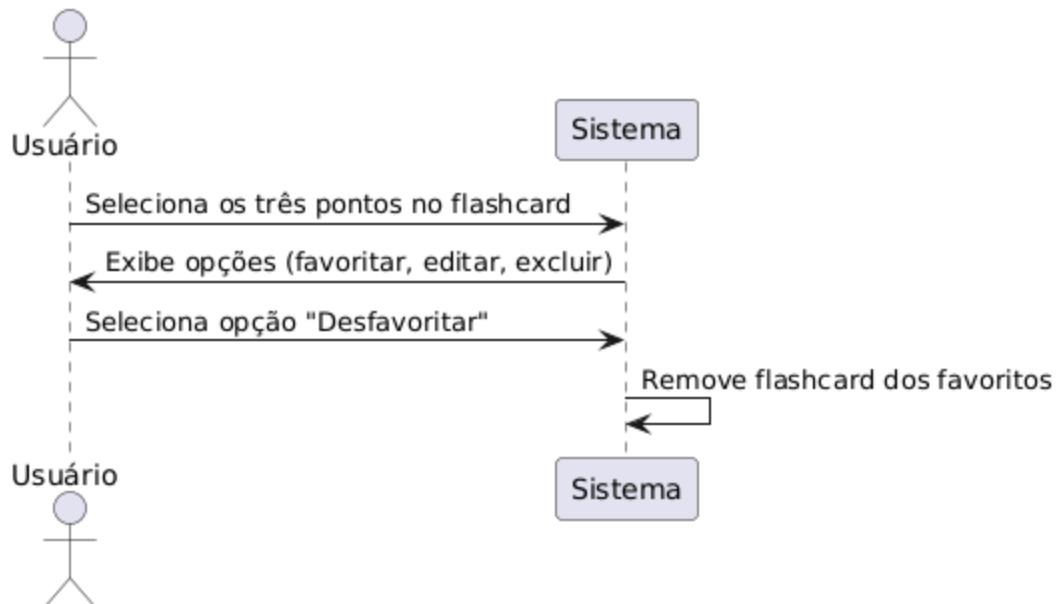
Seleção de um Flashcard do Histórico



Fluxo de Seleção de um Log do Histórico

Diagrama de Sequencia : UC-08

Desfavoritar Flashcard dos Favoritos



Fluxo de Desfavoritar Flashcards

Favoritar Flashcard dos Favoritos

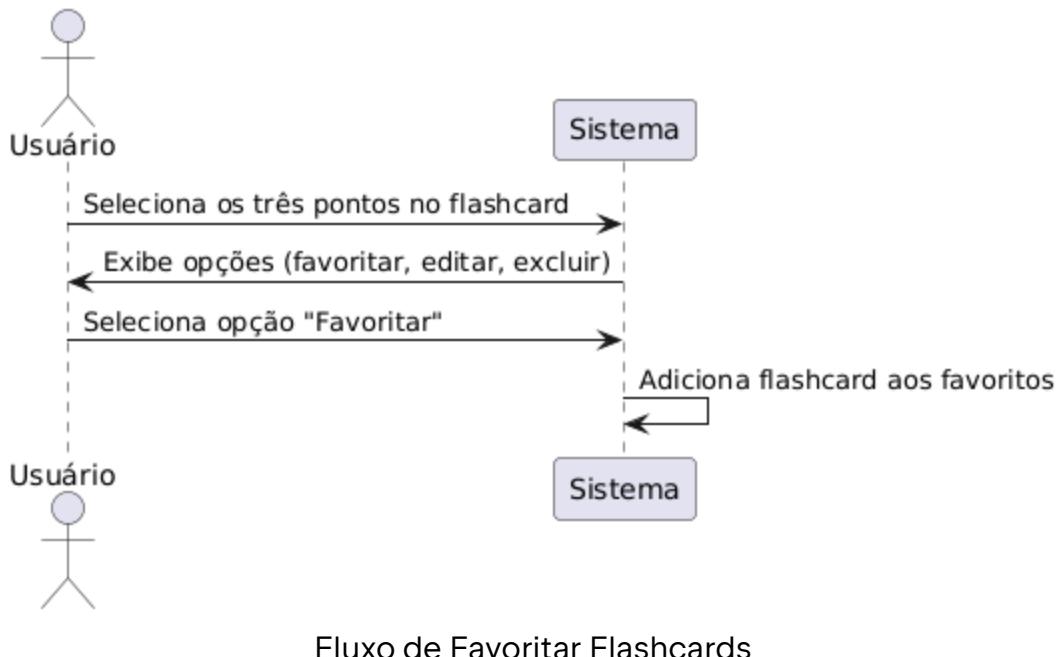
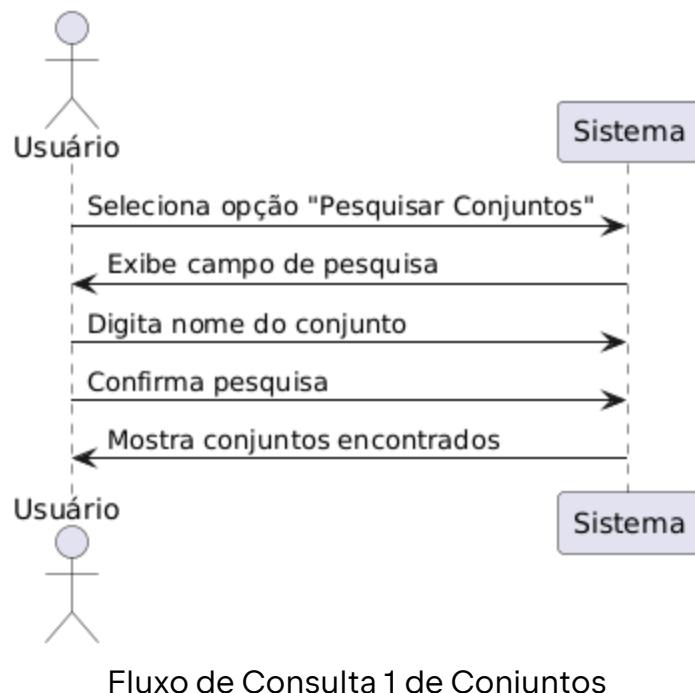
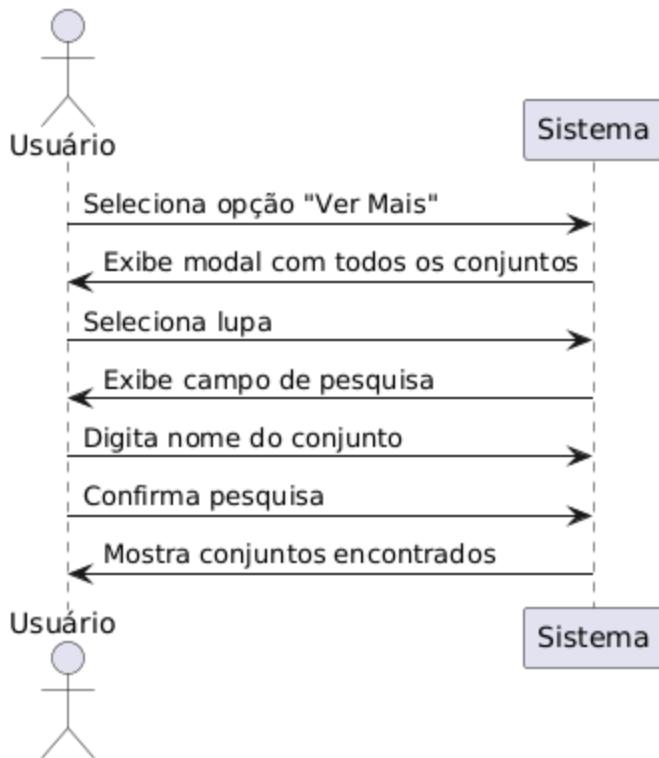


Diagrama de Sequencia : UC-09

1 Tipo de Consulta dos Conjuntos

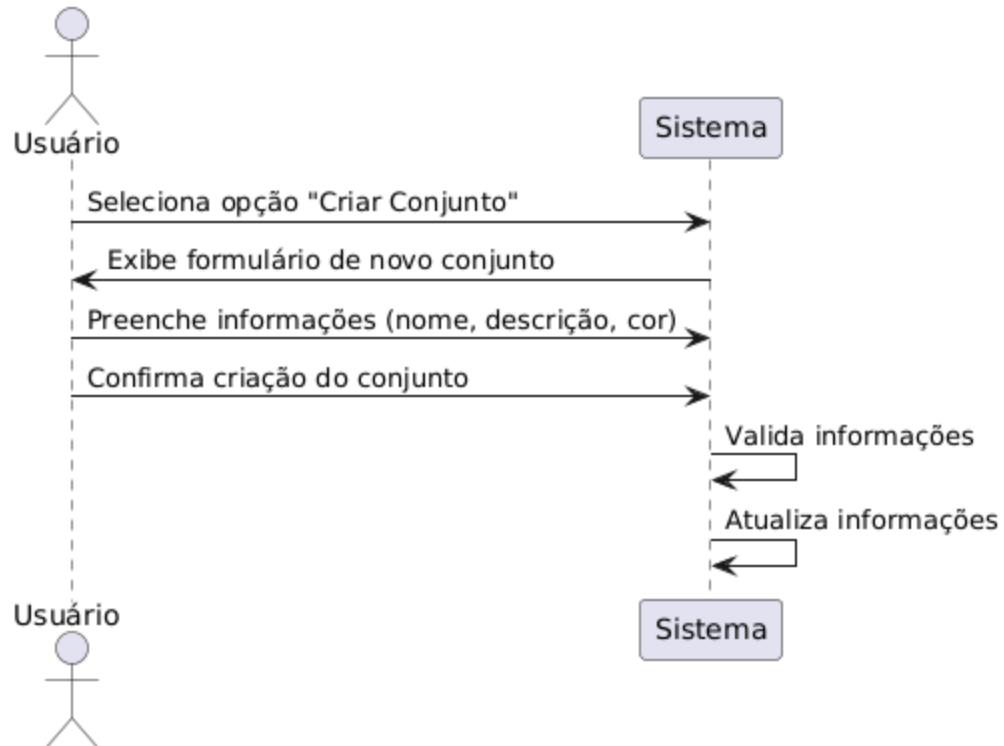


2 Tipo de Consulta dos Conjuntos



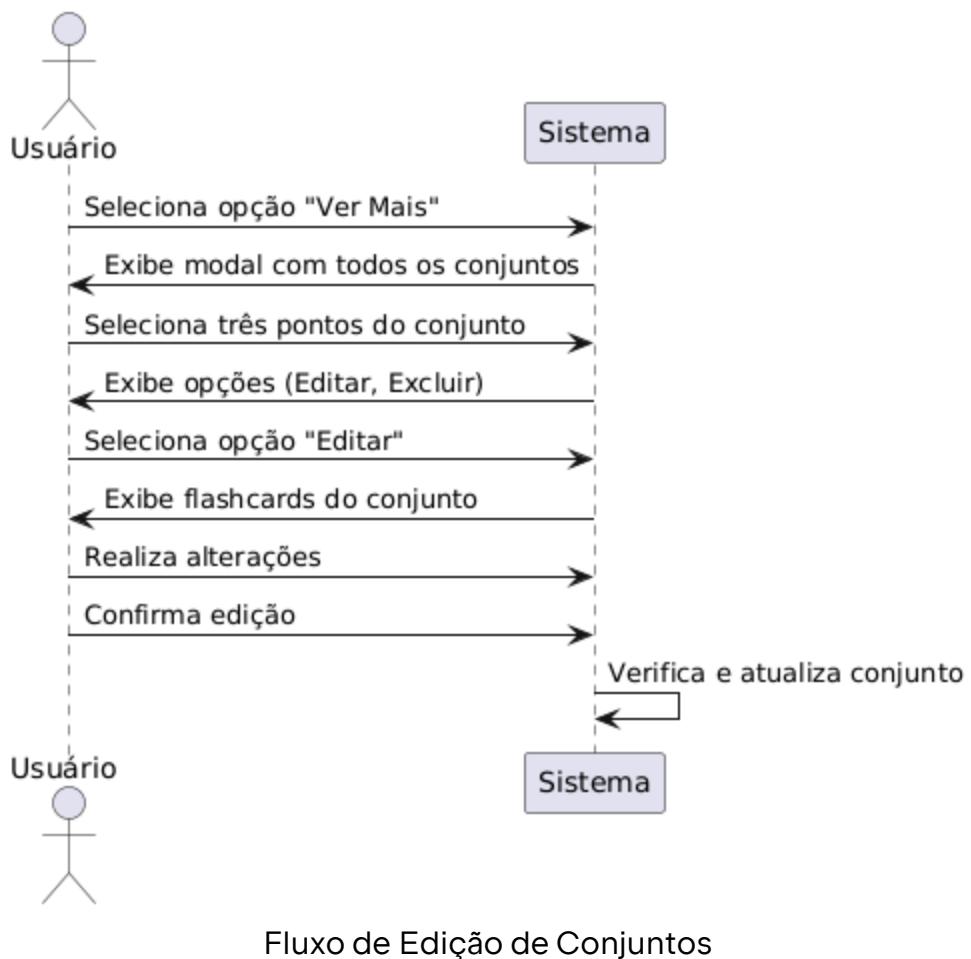
Fluxo de Consulta 2 de Conjuntos

Criação dos Conjuntos



Fluxo de Criação de Conjuntos

Edição dos Conjuntos



Exclusão dos Conjuntos

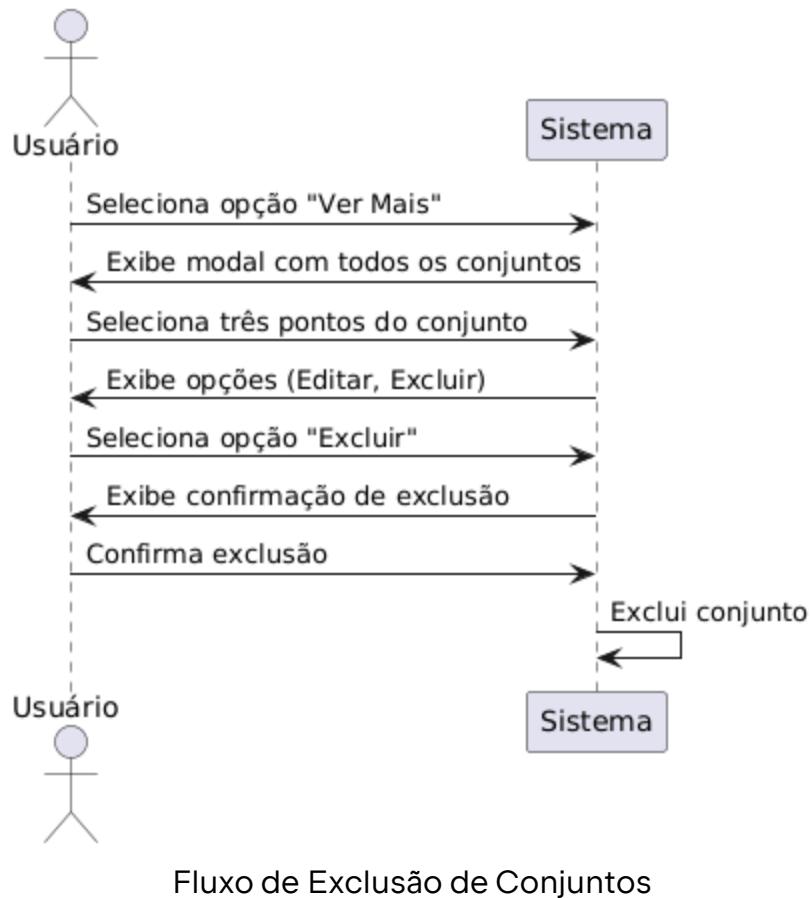
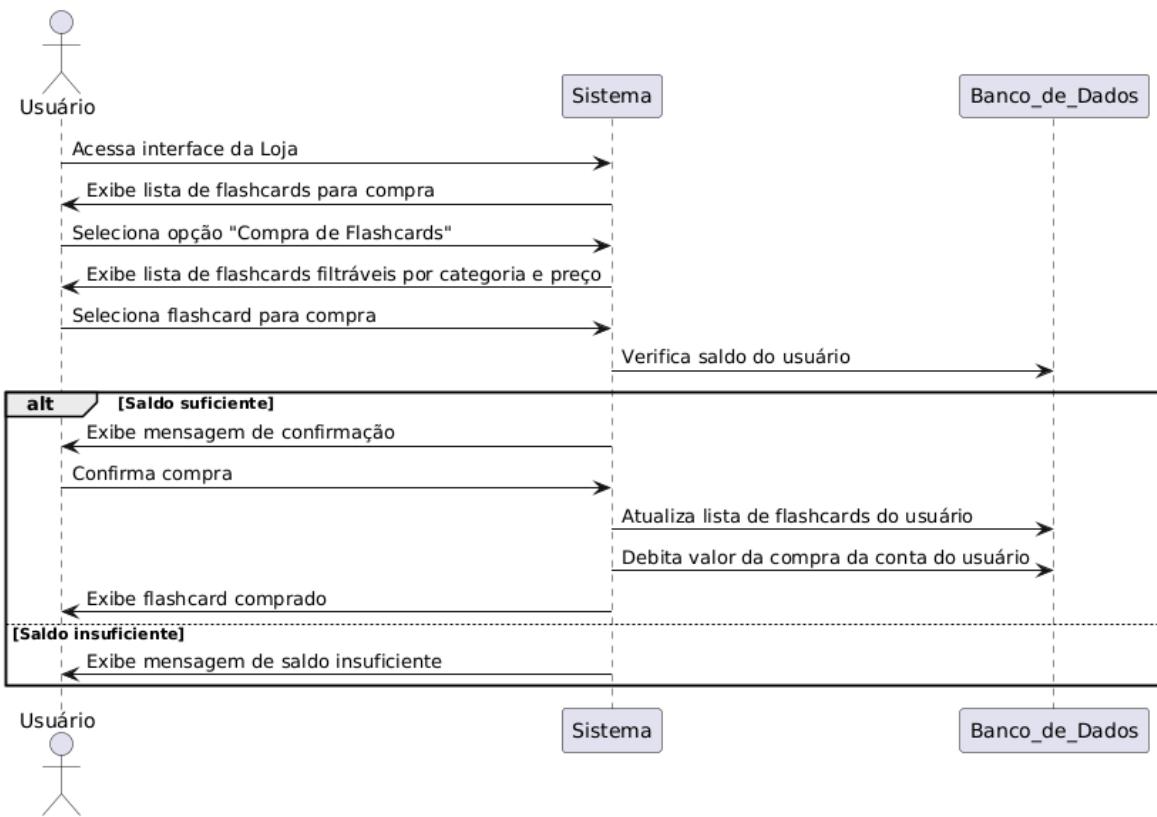


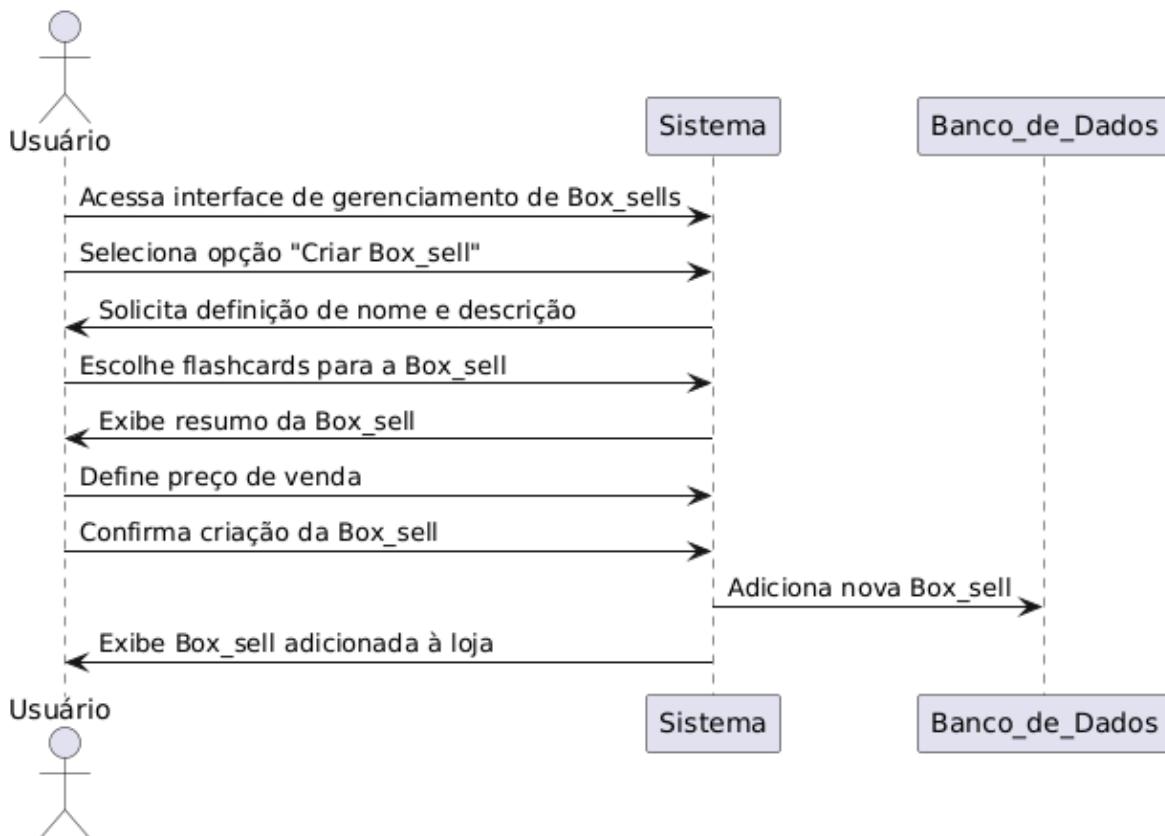
Diagrama de Sequencia : UC-10



Fluxo de Compra dos Flashcards

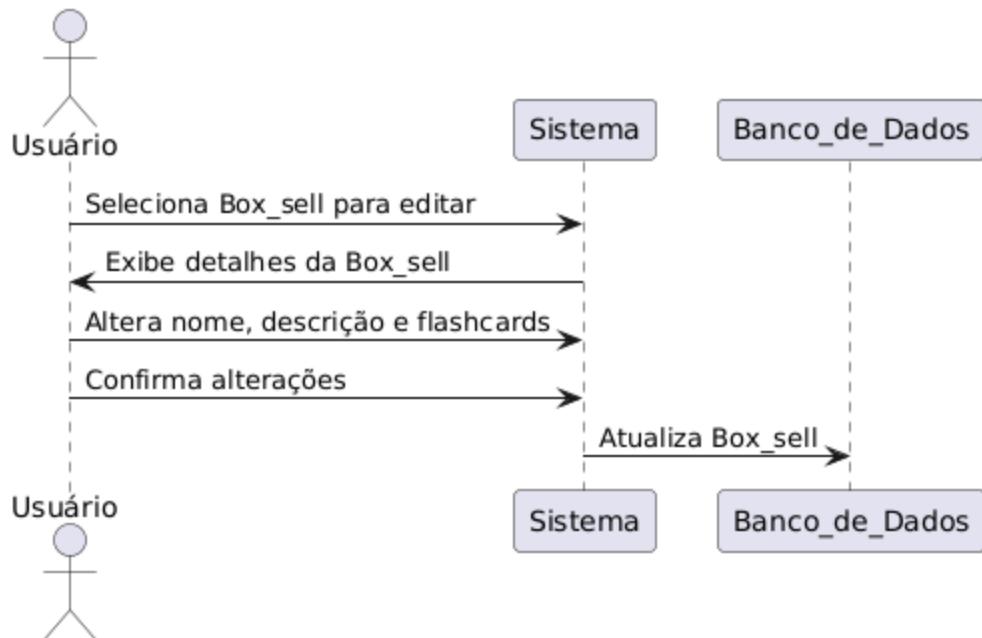
Diagrama de Sequencia : UC-11

Criação das Box Sells



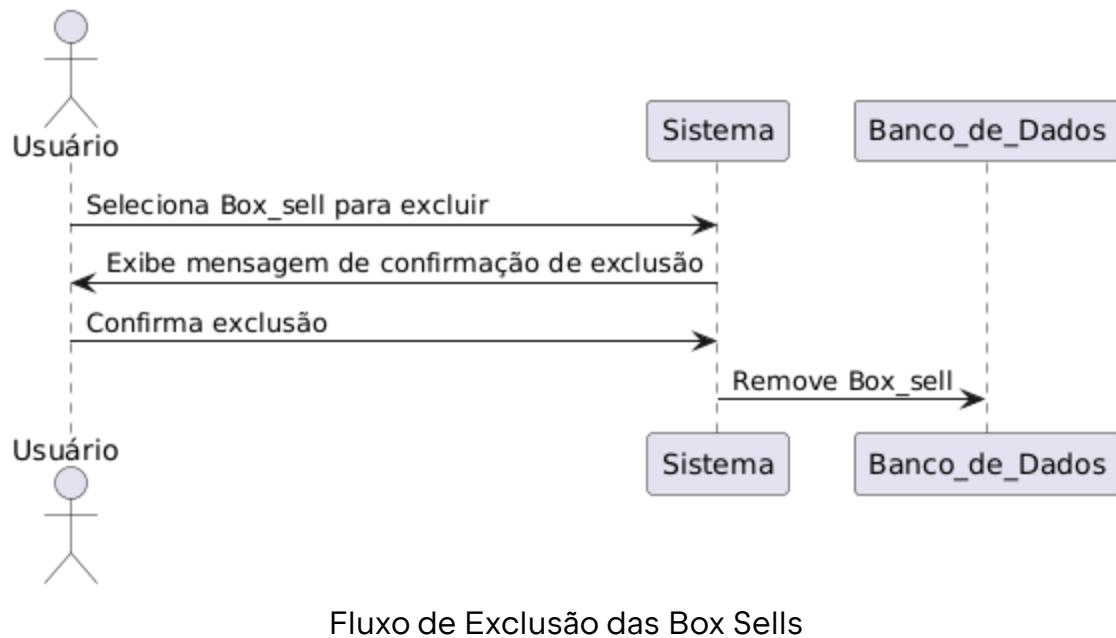
Fluxo de Criação das Box Sells

Edição das Box Sells



Fluxo de Edição das Box Sells

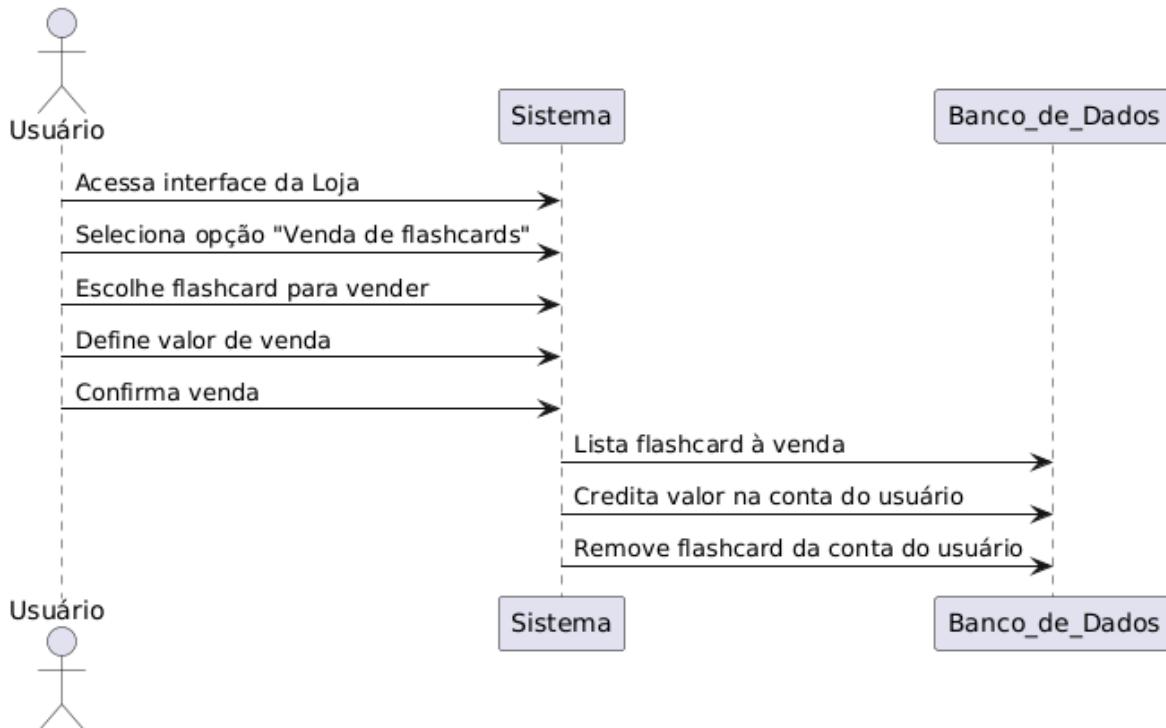
Exclusão das Box Sells



Fluxo de Exclusão das Box Sells

Diagrama de Sequencia : UC-12

Venda das Box Sells



Fluxo de Venda dos Flashcards

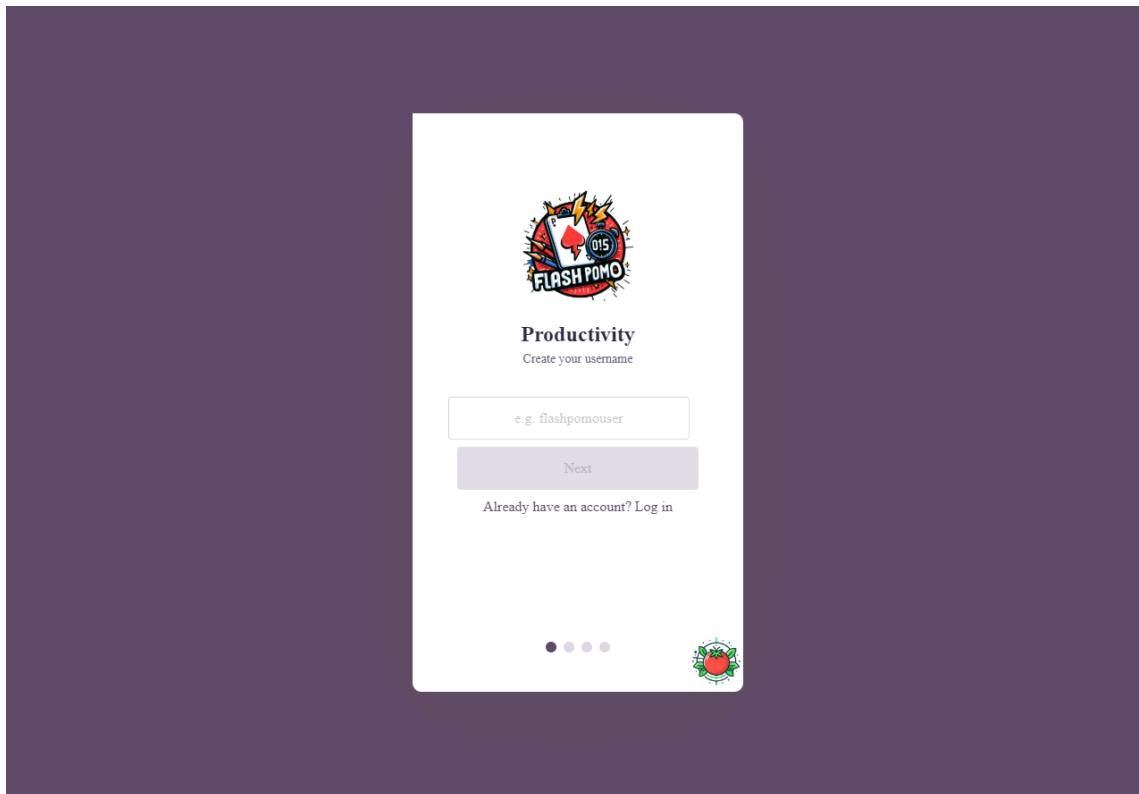
Manual do Usuário

O manual do usuário tem como objetivo orientar o usuário final do sistema a utilizá-lo de forma adequada e eficiente, contendo informações sobre as funcionalidades, recursos, instruções de uso, solução de problemas e demais informações relevantes ao usuário.

Cadastro de Usuários e Login

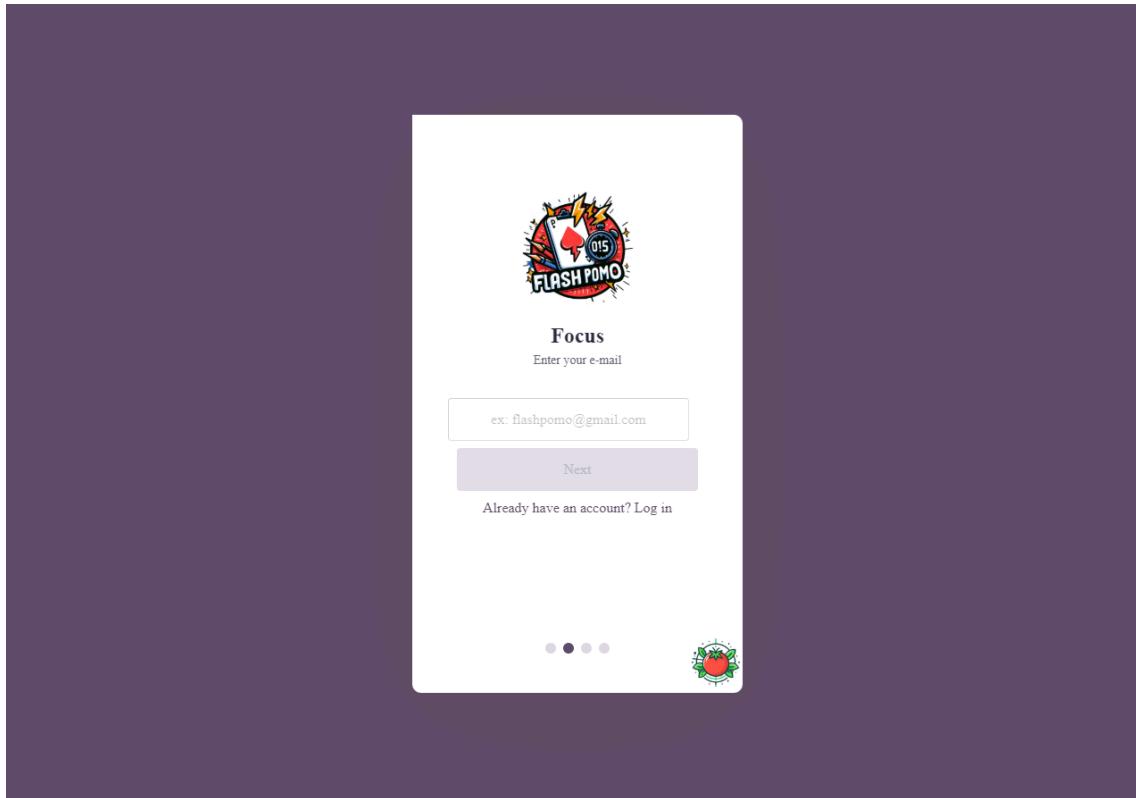
Para realizar o cadastro, o usuário deve informar os dados corretamente, preenchendo os campos necessários, cada campo devidamente preenchido deve ser confirmado pressionando o botão de continuar.

1. Primeiro o usuário criará o nome de usuário.



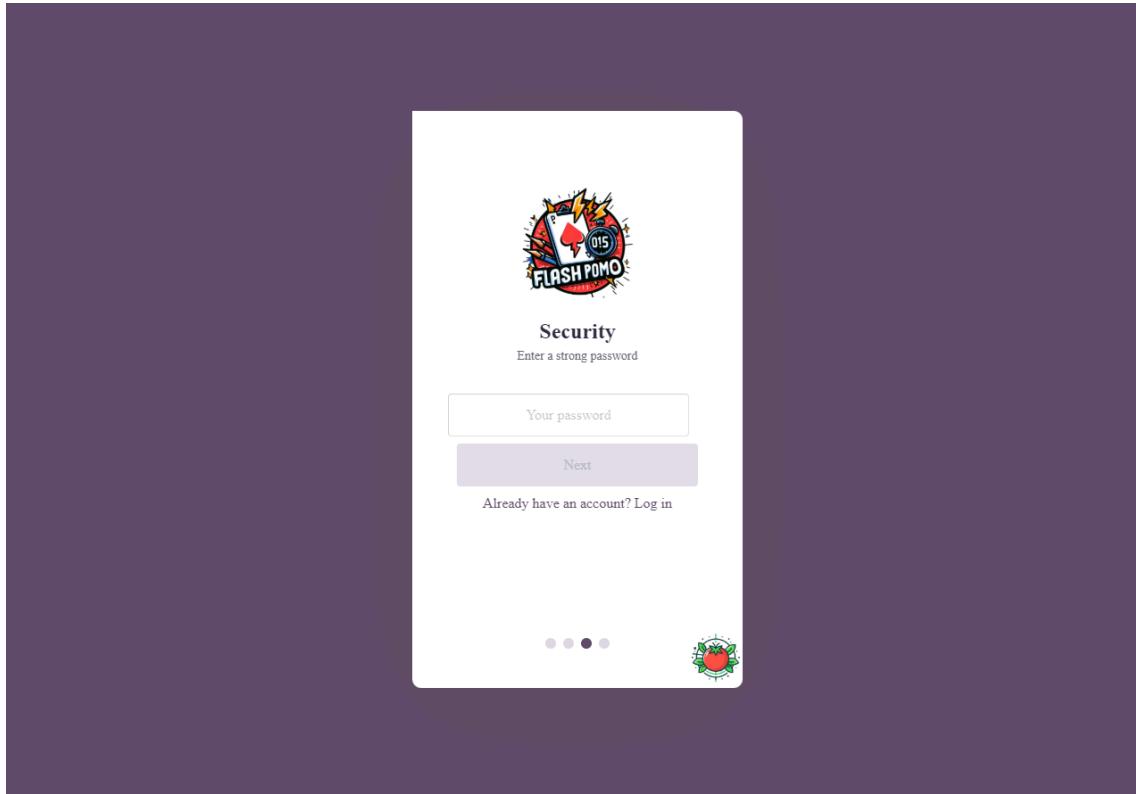
Cadastro nome de usuário

2. Em seguida o usuário informará o email.



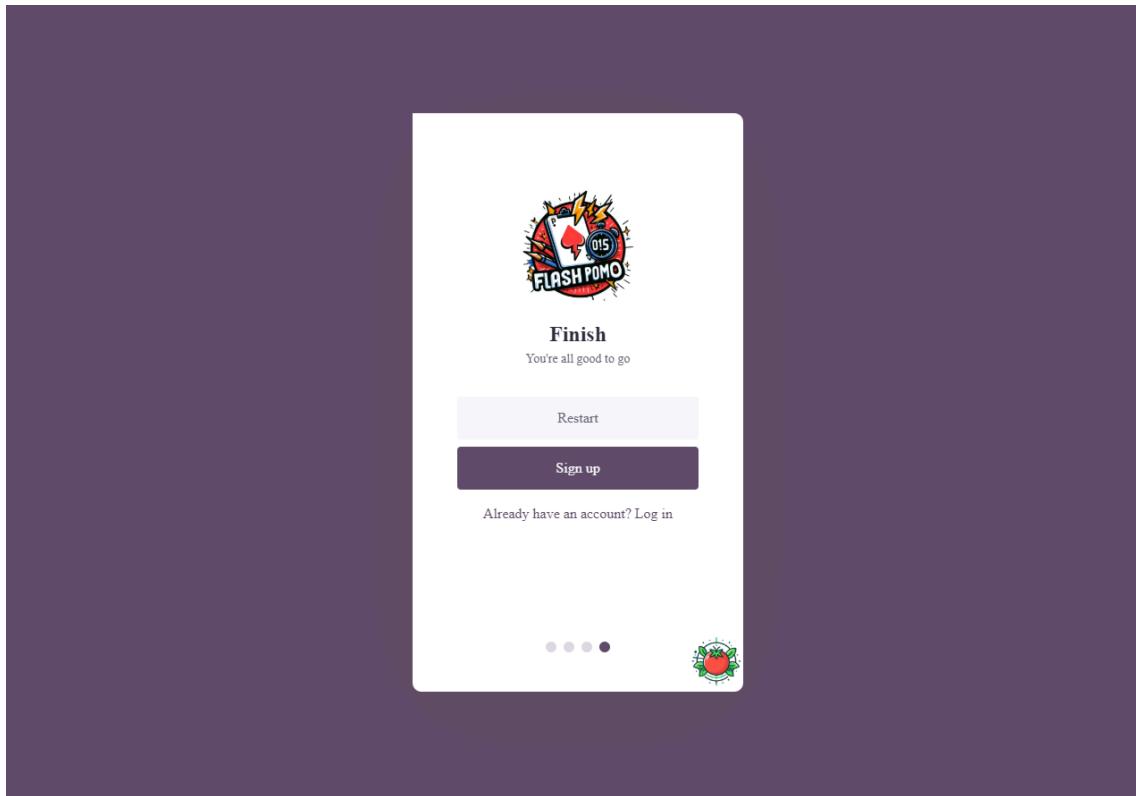
Cadastro email

3. O usuário criará sua senha.



Cadastro senha de usuário

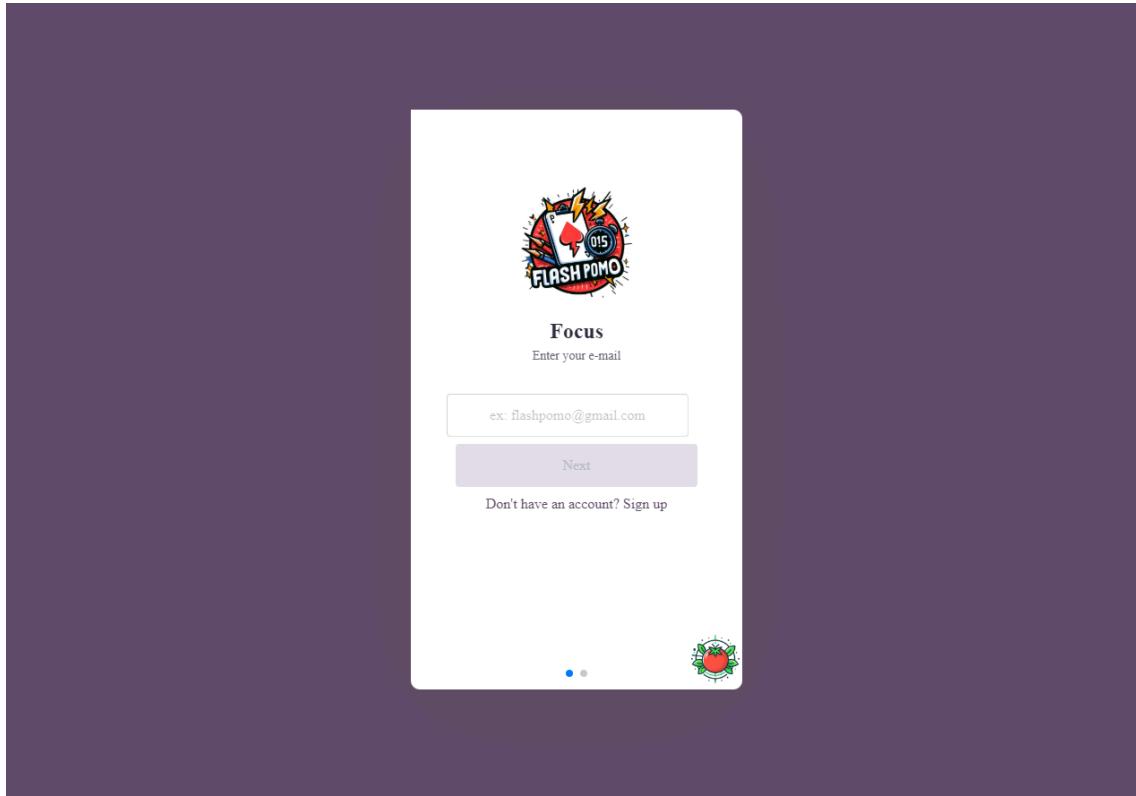
4. Após informar os devidos dados, o usuário confirma a criação de conta.



Signup4

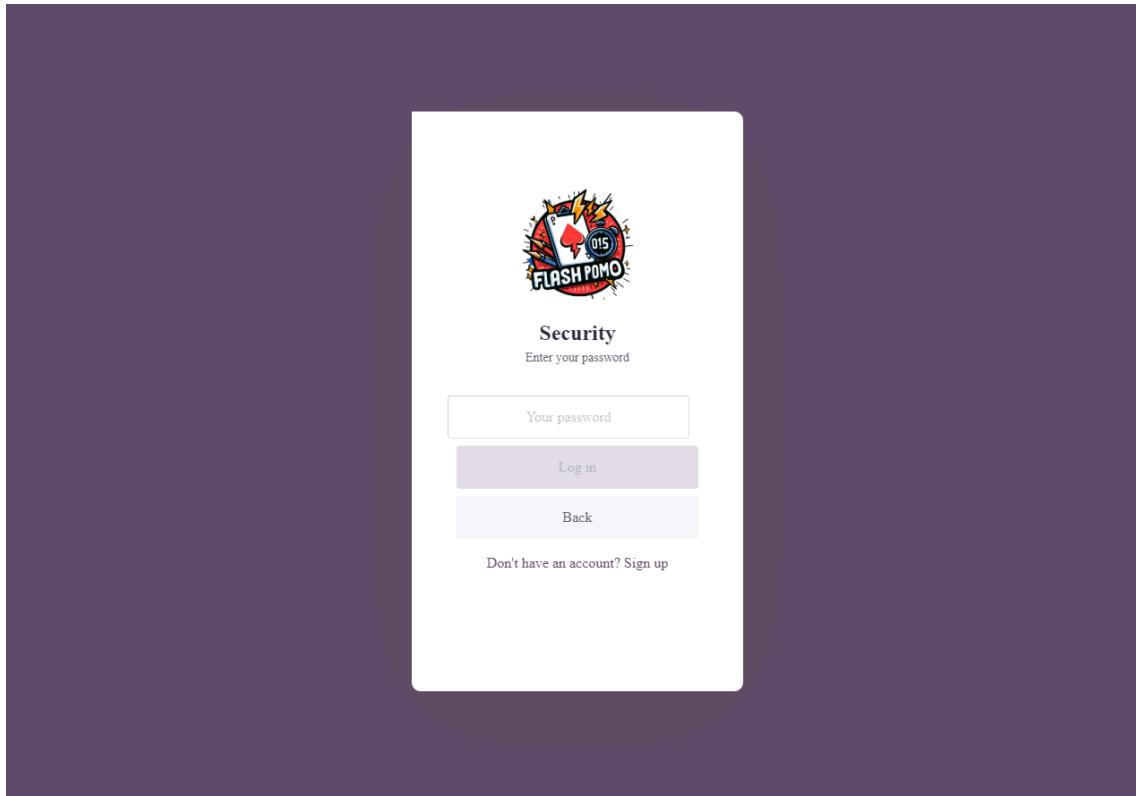
Pronto, a conta do usuário foi criada! Agora o usuário pode fazer log in.

5. O usuário na página de log in, informa o email.



Login1

6. O usuário digita sua senha e confirma o log in

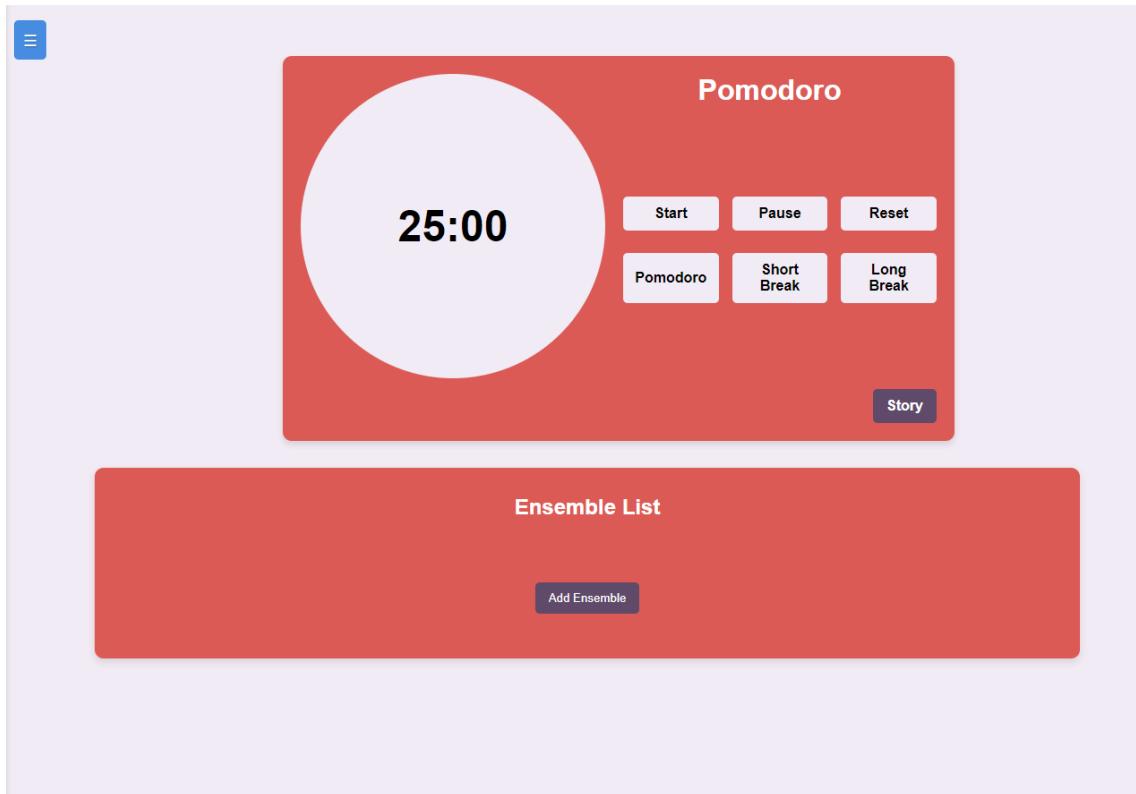


Login2

Criar FlashCard

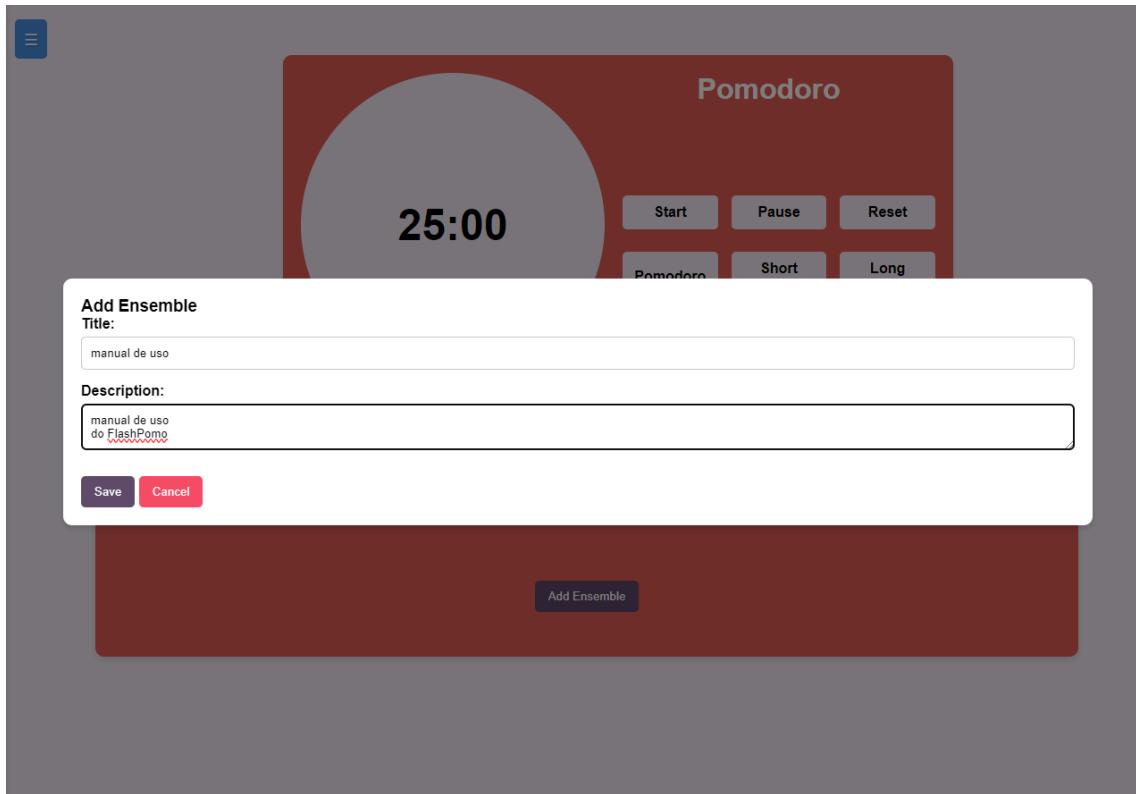
Na página de Dashboard o usuário precisará criar primeiramente o ensemble que será o conjunto presente dos flashcards, para tal selecionará o botão add ensemble, o sistema pedirá o título e a descrição do ensemble.

1. Na seção Ensemble List.



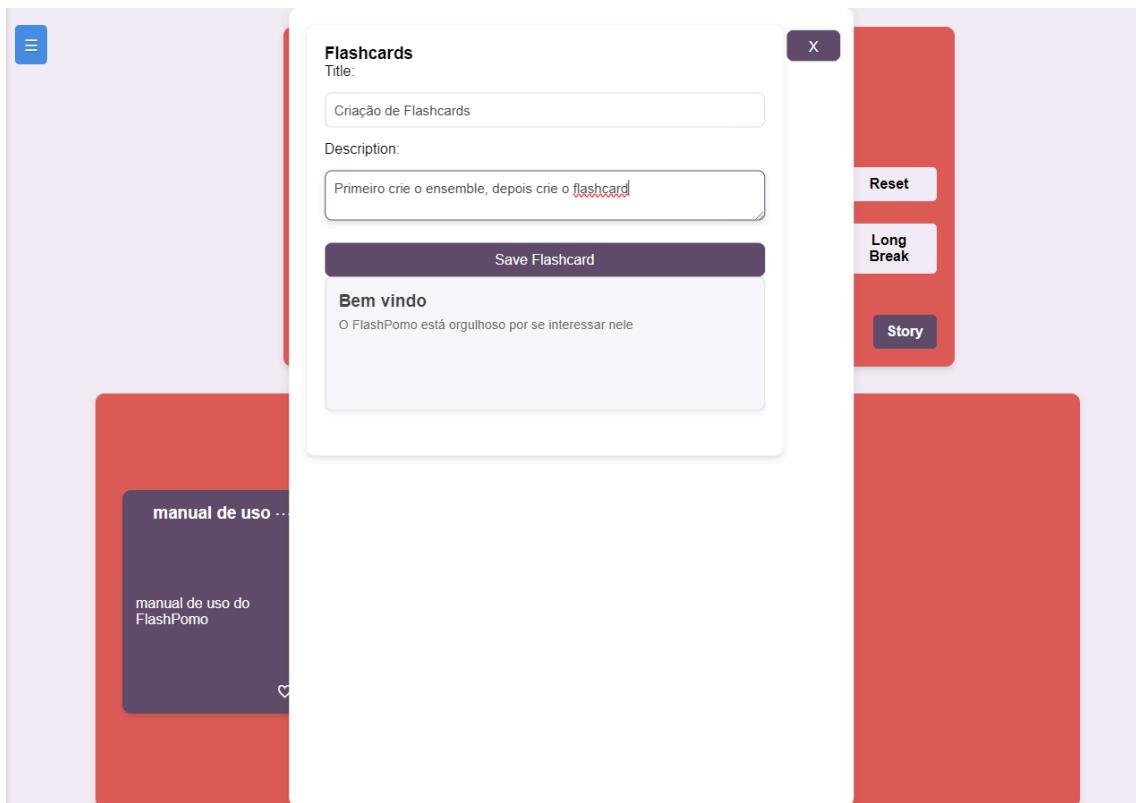
Dashboard1

2. Crie o ensemble, adicione o título e descrição, e salve.



Add ensemble

3. Pressione no ensemble e crie os flashcards, o título e descrição, depois é só salvar

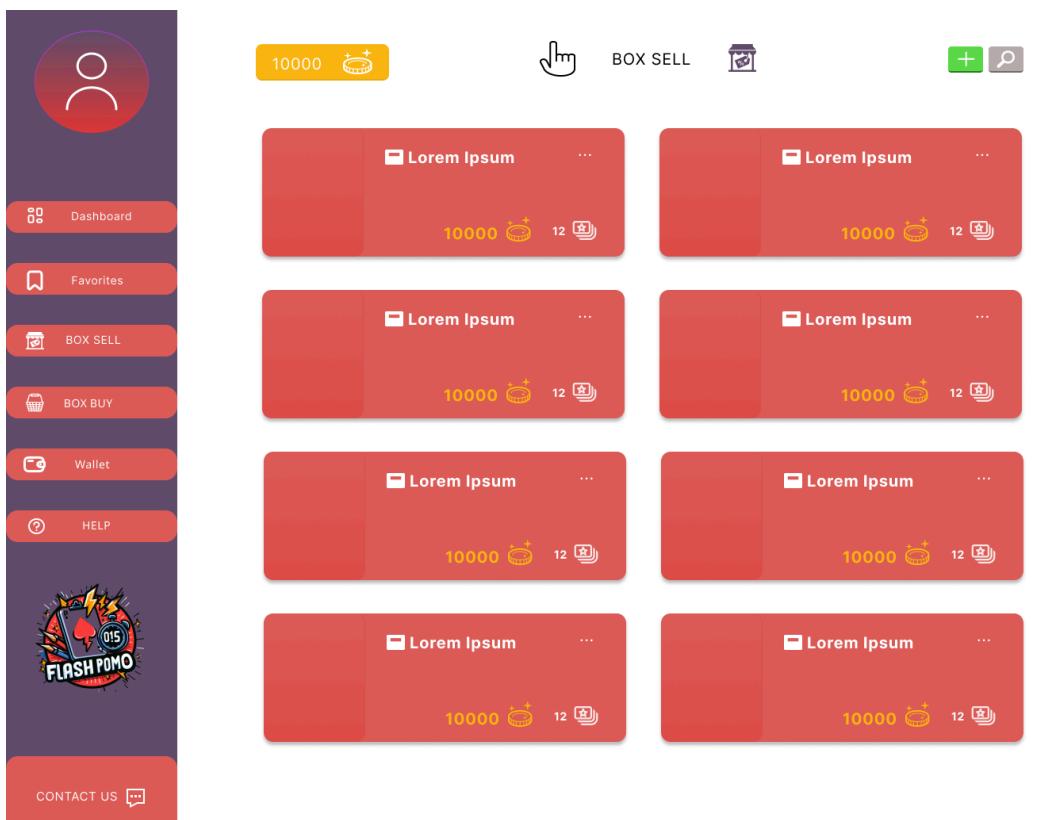


Add flashcard

Criar Box e Vender

Para vender uma box, primeiro o usuário precisa, criá-la.

1. No menu, o usuário entrará na página Box Sell.



Desktop user box shell

2. Nela precisará selecionar o botão de adicionar um "+" ao lado do botão de pesquisa. Em seguida deverá criar a box com os flashcards já existentes e, os quais podem pertencer a vários ensembles, e logo após selecionar o preço.

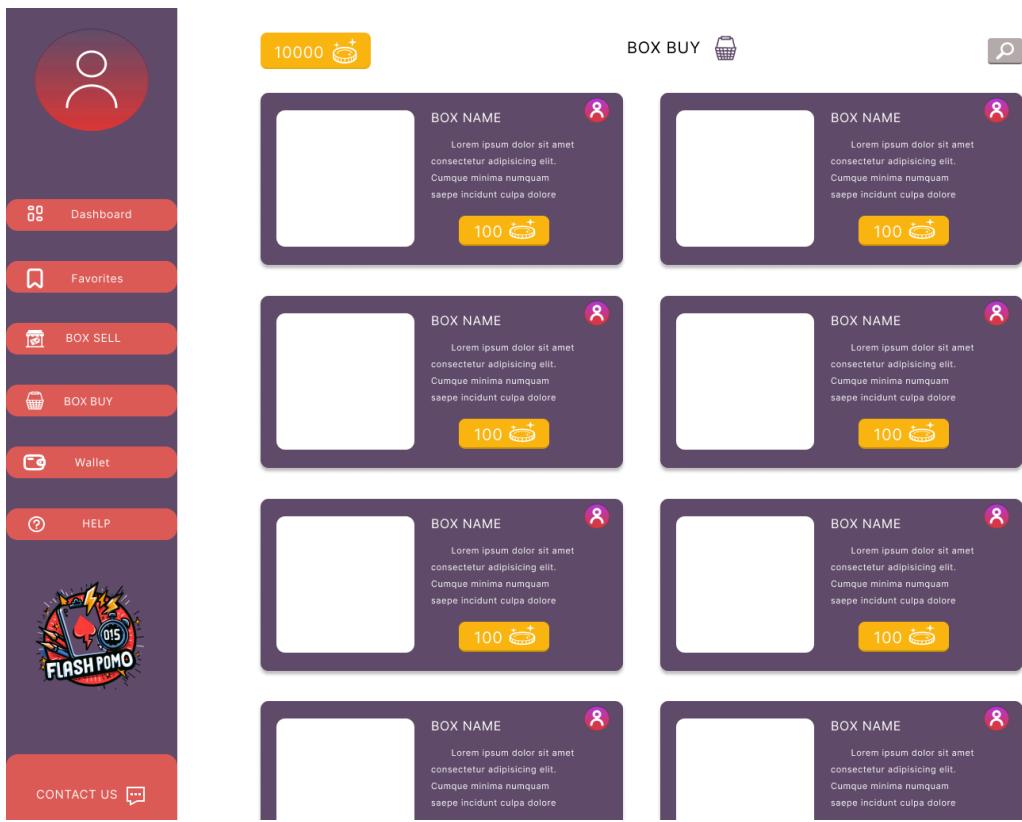


Desktop user creating box shell

Comprar Box

Para comprar uma box, o usuário deverá entrar na página Box Buy, e pesquisar pela box que ele deseja.

1. O usuário selecionará o botão presente do valor da box, e em seguida confirmará a compra.

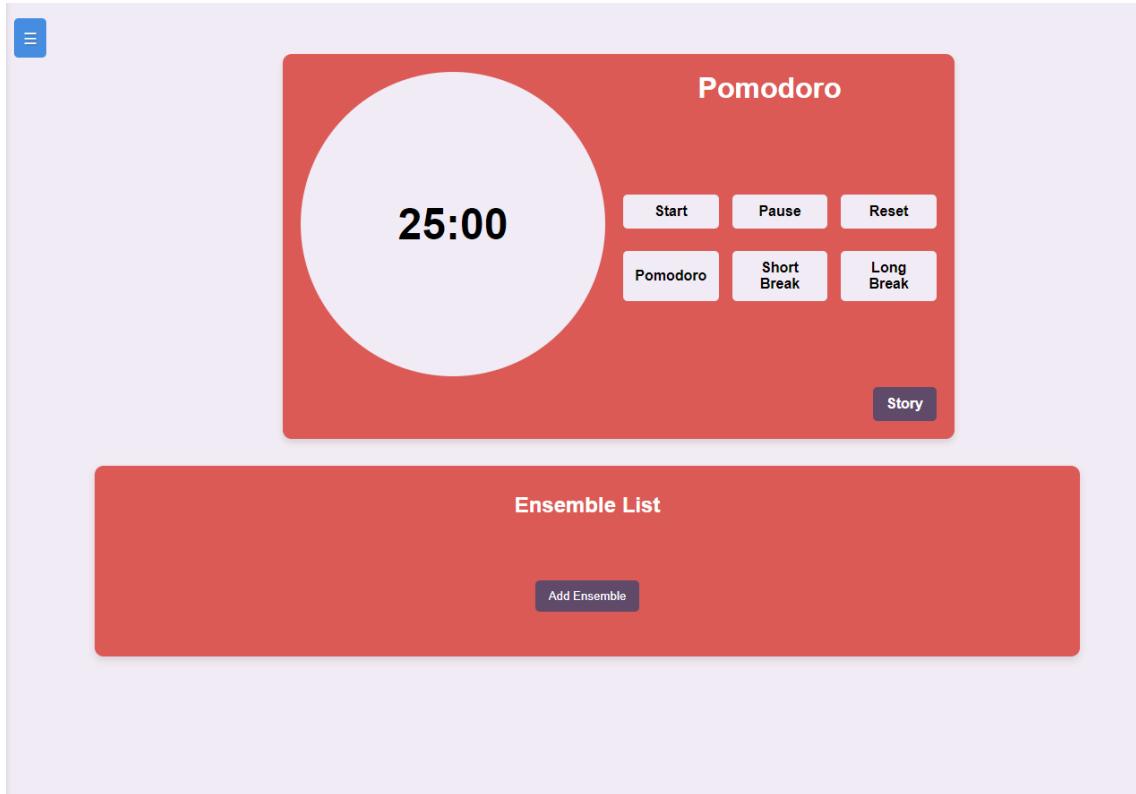


Interface box buy

Usar Pomodoro

Para o usuário usar o pomodoro, ele deverá primeiro estar no Dashboard.

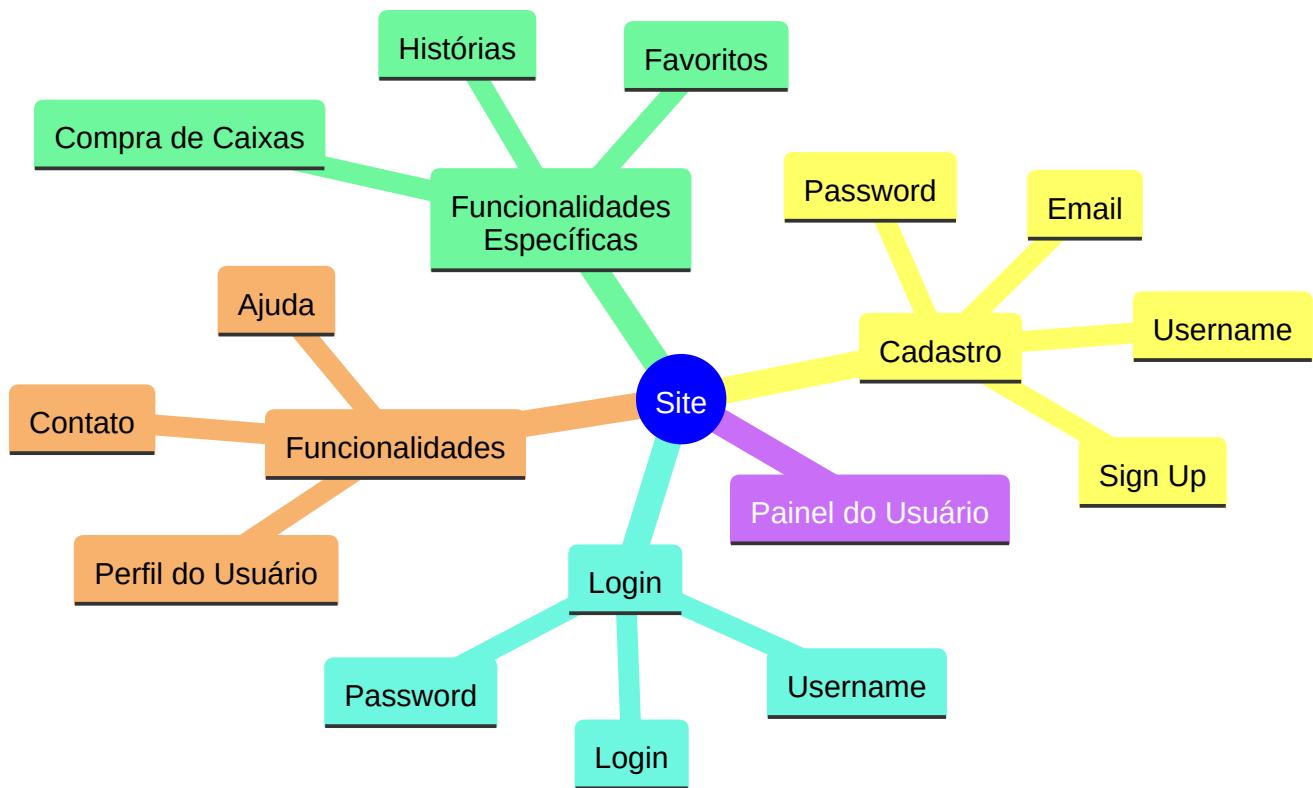
1. O usuário poderá escolher no pomodoro as funções Pomodoro, de 25min, short break de 5min, e long break de 15min. também poderá usar as funções de start para começar, pause para pausar e restart para recomeçar.



Dashboard1

Mapa do Site

A seção **Mapa do Site** apresenta uma visão estruturada das principais áreas e funcionalidades do site, organizada em formato de mapa mental. Este diagrama facilita a compreensão das seções e fluxos disponíveis, desde o cadastro e login até as funcionalidades específicas oferecidas ao usuário.



API Reference

Esta seção fornece uma visão geral detalhada dos endpoints da API disponíveis em nosso aplicativo de software. Ela inclui informações sobre os formatos de solicitação e resposta, bem como exemplos e diretrizes de uso.

Schemas

A seção de Schemas da API é uma parte essencial da documentação, pois ela define os formatos de dados esperados pelos diferentes endpoints da nossa API.

Os schemas descrevem, em detalhes, a estrutura e as regras de validação dos objetos JSON que são utilizados nas solicitações e respostas da API. Essa padronização é fundamental para garantir a consistência e a interoperabilidade entre o cliente (aplicação que consome a API) e o servidor (nossa aplicação que fornece a API).

ResponseType

O schema `ResponseType` é utilizado para representar uma resposta HTTP padrão, contendo um código de status, um corpo de resposta e, opcionalmente, headers adicionais. Esse schema é usado como a estrutura base para as respostas de todos os endpoints da API, garantindo uma consistência no formato das respostas.

ResponseType

Ensemble Schemas

O schema `Ensemble` define a estrutura de um conjunto de flashcards (ensemble) no sistema, incluindo informações como título, descrição, data de criação, entre outras.

Ensemble

O schema `EnsembleRequestDTO` é utilizado para representar os dados necessários para criar ou atualizar um ensemble.

EnsembleRequestDTO

Purchase Schemas

O schema `PurchaseRequestDTO` é usado para representar os dados necessários para realizar a compra de um ensemble de flashcards.

PurchaseRequestDTO

PomodoroStory Schemas

O schema `PomodoroStoryRequestDTO` define a estrutura de uma história de Pomodoro, contendo informações como título, descrição, status e outras propriedades relevantes.

PomodoroStoryRequestDTO

Favorite Schemas

O schema `FavoriteRequestDTO` é usado para representar os dados necessários para criar ou atualizar um favorito do usuário.

FavoriteRequestDTO

Box Schemas

O schema `BoxRequestDTO` define a estrutura de uma caixa (box) no sistema, que é utilizada para organizar os flashcards do usuário.

BoxRequestDTO

GetByIdResponse

O schema `GetByIdResponse` é usado para representar a estrutura de uma resposta que retorna um objeto identificado por um ID.

GetByIdResponse

ResponseCommonDTO

O schema `ResponseCommonDTO` define uma estrutura de resposta padrão, contendo informações comuns, como código de status, mensagem e dados.

ResponseCommonDTO

Flashcard Schemas

O schema `Flashcard` define a estrutura de um flashcard no sistema, incluindo informações como pergunta, resposta, data de criação e outras propriedades relevantes.

Flashcard

O schema `FlashcardRequestEditDTO` é usado para representar os dados necessários para editar um flashcard existente.

FlashcardRequestEditDTO

O schema `FlashcardRequestDTO` é usado para representar os dados necessários para criar um novo flashcard.

FlashcardRequestDTO

User Schemas

O schema `User` define a estrutura de um usuário no sistema, incluindo informações como nome, email, senha e outras propriedades relevantes.

User

O schema `UserEditRequestDTO` é usado para representar os dados necessários para atualizar as informações de um usuário.

UserEditRequestDTO

O schema `ResponseUserEditDTO` é usado para representar a resposta após a atualização das informações de um usuário.

ResponseUserEditDTO

Auth Schemas

RegisterRequestDTO

O schema `'RegisterRequestDTO'` é utilizado para representar os dados necessários para o registro de um novo usuário.

LoginRequestDTO

O schema `LoginRequestDTO` é utilizado para representar os dados necessários para o login de um usuário.

Auth Endpoints

A autenticação é um aspecto crucial de aplicações web, permitindo que os usuários acessem suas contas de forma segura e executem diversas ações. Os endpoints de autenticação mais comuns incluem registro, login, logout e gerenciamento de senha.

O endpoint de registro permite que os usuários criem uma nova conta, fornecendo informações como nome de usuário, email e senha. Após um registro bem-sucedido, o usuário normalmente recebe um token de autenticação, que pode ser usado em solicitações autenticadas posteriores.

O endpoint de login é usado para autenticar os usuários e obter um token de autenticação, que é então usado para autorizar as ações do usuário. Esse token é tipicamente um JSON Web Token (JWT) que contém as informações do usuário e é assinado pelo servidor.

POST /auth/login

Este endpoint permite que os usuários realizem o login no sistema e obtenham um token de autenticação.

POST /auth/login

Request parameters

Body

application/json LoginRequestDTO **required**

Child attributes

email object

Child attributes

password object

Child attributes

JSON example

```
{  
    "email": {},  
    "password": {}  
}
```

Responses

200 OK

Content type */*

OK

 / ResponseEntity

 Child attributes

headers HttpHeaders **required**

 Child attributes

POST /auth/register

Este endpoint permite que os usuários se registrem no sistema, criando uma nova conta.

POST /auth/register

Request parameters

Body

application/json RegisterRequestDTO **required**

 Child attributes

name object

 Child attributes

email object

Child attributes

password object

Child attributes

avatar object

Child attributes

createdAt object

Child attributes

status object

Child attributes

role object

Child attributes

JSON example

```
{  
  "name": {},  
  "email": {},  
  "password": {},  
  "avatar": {},  
  "createdAt": {},  
  "status": {},  
  "role": {}  
}
```

Responses

200 OK

Content type */*

OK

/* ResponseEntity

Child attributes

headers HttpHeaders required

Child attributes

User Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de usuários no sistema.

GET /user

Este endpoint retorna todos os usuários cadastrados no sistema.

GET /user

Responses

200 OK

Content type */*

OK

/ object

Child attributes

GET /user/{userId}

Este endpoint retorna os detalhes de um usuário específico, identificado pelo seu ID.

GET /user/{userId}

Request parameters

Path

userId object required

Child attributes

Responses

200 OK

Content type */*

OK

/ GetByIdResponse

Child attributes

name object

Child attributes

email object

Child attributes

PUT /user/

Este endpoint permite atualizar as informações de um usuário existente, identificado pelo seu ID.

PUT /user/{userId}

Request parameters

Path

userId object **required**

Child attributes

Body

application/json UserEditRequestDTO **required**

Child attributes

name object

Child attributes

email object

Child attributes

avatar object

Child attributes

password object

Child attributes

role object

Child attributes

updatedAt object

Child attributes

JSON example

```
{  
  "name": {},  
  "email": {},  
  "avatar": {},  
  "password": {},  
  "role": {},  
  "updatedAt": {}  
}
```

Responses

200 OK

Content type */*

OK

/ ResponseUserEditDTO

Child attributes

name object

Child attributes

email object

Child attributes

password object

Child attributes

avatar object

Child attributes

role object

Child attributes

DELETE /user/

Este endpoint permite excluir um usuário do sistema, identificado pelo seu ID.

DELETE

/user/{userId}

Request parameters

Path

userId object required

Child attributes

Responses

200 OK

Content type */*

OK

/ ResponseCommonDTO

Child attributes

message object

Child attributes

Ensemble Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de ensembles no sistema.

GET /ensemble/{ensembleId}

Este endpoint retorna os detalhes de um ensemble específico, identificado pelo seu ID.

GET /ensemble/{ensembleId}

Request parameters

Path

ensembleId object **required**

Child attributes

Responses

200 OK

Content type `*/*`

OK

`*/*` `ResponseEntity`

Child attributes

headers `HttpHeaders` **required**

Child attributes

PUT /ensemble/

Este endpoint permite atualizar as informações de um ensemble existente, identificado pelo seu ID.

PUT /ensemble/{ensembleId}

Request parameters

Path

ensembleId object **required**

Child attributes

Body

application/json EnsembleRequestDTO **required**

Child attributes

name object

Child attributes

description object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "name": {},  
  "description": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

 / ResponseEntity

 Child attributes

headers HttpHeaders required

 Child attributes

DELETE /ensemble/

Este endpoint permite excluir um ensemble do sistema, identificado pelo seu ID.

DELETE /ensemble/{ensembleId}

Request parameters

Path

ensembleId object required

 Child attributes

Responses

200 OK

Content type */*

OK

 / ResponseEntity

 Child attributes

headers HttpHeaders required

Child attributes

GET /ensemble/

Este endpoint retorna todos os ensembles cadastrados no sistema.

GET /ensemble/

Responses

200 OK

Content type */*

OK

/* object

Child attributes

POST /ensemble/

Este endpoint permite criar um novo ensemble no sistema.

POST /ensemble/

Request parameters

Body

application/json EnsembleRequestDTO required

Child attributes

name object

Child attributes

description object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "name": {},  
  "description": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders required

Child attributes

Flashcard Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de flashcards no sistema.

GET /flashcard/{flashcardId}

Este endpoint retorna os detalhes de um flashcard específico, identificado pelo seu ID.

`GET /flashcard/{flashcardId}`

Request parameters

Path

flashcardId object `required`

Child attributes

Responses

`200 OK`

`Content type` `*/*`

OK

`*/* ResponseEntity`

Child attributes

headers `HttpHeaders required`

Child attributes

PUT /flashcard/

Este endpoint permite atualizar as informações de um flashcard existente, identificado pelo seu ID.

PUT /flashcard/{flashcardId}

Request parameters

Path

flashcardId object **required**

Child attributes

Body

application/json FlashcardRequestEditDTO **required**

Child attributes

title object

Child attributes

content object

Child attributes

createdAt object

Child attributes

updatedAt object

Child attributes

ensembleId object

Child attributes

status object

Child attributes

JSON example

```
{  
    "title": {},  
    "content": {},  
    "createdAt": {},  
    "updatedAt": {},  
    "ensembleId": {},  
    "status": {}  
}
```

Responses

200 OK

Content type */*

OK

 */ ResponseEntity

 Child attributes

headers HttpHeaders **required**

 Child attributes

DELETE /flashcard/

Este endpoint permite excluir um flashcard do sistema, identificado pelo seu ID.

DELETE /flashcard/{flashcardId}

Request parameters

Path

flashcardId object [required](#)

Child attributes

Responses

200 OK

Content type [*/*](#)

OK

[*/*](#) **ResponseEntity**

Child attributes

headers [HttpHeaders](#) [required](#)

Child attributes

GET /flashcard/

Este endpoint retorna todos os flashcards cadastrados no sistema.

GET /flashcard/

Responses

200 OK

Content type [*/*](#)

OK

[*/*](#) **object**

Child attributes

POST /flashcard/

Este endpoint permite criar um novo flashcard no sistema.

POST /flashcard/

Request parameters

Body

application/json FlashcardRequestDTO required

Child attributes

title object

Child attributes

content object

Child attributes

createdAt object

Child attributes

updatedAt object

Child attributes

ensembleId object

Child attributes

JSON example

```
{  
  "title": {},  
  "content": {},  
  "createdAt": {},  
  "updatedAt": {},  
  "ensembleId": {}  
}
```

```
"ensembleId": {}  
}
```

Responses

200 OK

Content type */*

OK

 / ResponseEntity

 Child attributes

headers HttpHeaders **required**

 Child attributes

GET /flashcard/ensemble/

Este endpoint retorna todos os flashcards associados a um ensemble específico, identificado pelo seu ID.

GET /flashcard/ensemble/{ensemble}

Request parameters

Path

ensemble object **required**

 Child attributes

Responses

200 OK

Content type */*

OK

/ object

Child attributes

PomodoroStory Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de histórias de Pomodoro no sistema.

GET /pomodoro-story/

Este endpoint retorna os detalhes de uma história de Pomodoro específica, identificada pelo seu ID.

GET /pomodoro-story/{pomodoroStoryId}

Request parameters

Path

pomodoroStoryId object **required**

Child attributes

Responses

200 OK

Content type ***/***

OK

/ ResponseEntity

Child attributes

headers HttpHeaders **required**

Child attributes

PUT /pomodoro-story/

Este endpoint permite atualizar as informações de uma história de Pomodoro existente, identificada pelo seu ID.

PUT /pomodoro-story/{pomodoroStoryId}

Request parameters

Path

pomodoroStoryId object **required**

Child attributes

Body

application/json PomodoroStoryRequestDTO **required**

Child attributes

pomodoroStoryId object

Child attributes

startDateTimestamp object

Child attributes

endDateTimestamp object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "pomodoroStoryId": {} ,
```

```
"startDateTimestamp": {} ,  
"endDateTimestamp": {} ,  
"userId": {}  
}
```

Responses

200 OK

Content type `*/*`

OK

`*/*` ResponseEntity

Child attributes

headers HttpHeaders `required`

Child attributes

POST /pomodoro-story/

Este endpoint permite criar uma nova história de Pomodoro no sistema.

`POST` `/pomodoro-story/{pomodoroStoryId}`

Request parameters

Path

pomodoroStoryId object `required`

Child attributes

Body

application/json PomodoroStoryRequestDTO `required`

Child attributes

pomodoroStoryId object

Child attributes

startDateTimestamp object

Child attributes

EndDateTimestamp object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "pomodoroStoryId": {},  
  "startDateTimestamp": {},  
  "EndDateTimestamp": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

 // ResponseEntity

 Child attributes

headers HttpHeaders **required**

Child attributes

GET /pomodoro-story/

Este endpoint retorna todas as histórias de Pomodoro cadastradas no sistema.

GET /pomodoro-story/

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders required

Child attributes

Favorite Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de favoritos do usuário no sistema.

GET /favorite/

Este endpoint retorna os detalhes de um favorito específico, identificado pelo seu ID.

`GET /favorite/{favoriteld}`

Request parameters

Path

favoriteld object `required`

Child attributes

Responses

`200 OK`

`Content type` `*/*`

OK

`*/* ResponseEntity`

Child attributes

headers `HttpHeaders required`

Child attributes

PUT /favorite/

Este endpoint permite atualizar as informações de um favorito existente, identificado pelo seu ID.

PUT /favorite/{favoriteld}

Request parameters

Path

favoriteld object **required**

Child attributes

Body

application/json FavoriteRequestDTO **required**

Child attributes

farvoriteld object

Child attributes

favoritationDateTimestamp object

Child attributes

status object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "farvoriteId": {},  
  "favoritationDateTimestamp": {},  
  "status": {},  
  "userId": {}  
}
```

```
"userId": {}  
}
```

Responses

200 OK

Content type */*

OK

 / ResponseEntity

 Child attributes

headers HttpHeaders required

 Child attributes

DELETE /favorite/

Este endpoint permite excluir um favorito do sistema, identificado pelo seu ID.

DELETE /favorite/{favoritoid}

Request parameters

Path

favoritoid object required

 Child attributes

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders [required](#)

Child attributes

GET /favorite/

Este endpoint retorna todos os favoritos cadastrados no sistema.

[GET](#) /favorite/

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders [required](#)

Child attributes

POST /favorite/

Este endpoint permite criar um novo favorito no sistema.

[POST](#) /favorite/

Request parameters

Body

application/json FavoriteRequestDTO [required](#)

Child attributes

farvoriteId object

Child attributes

favoritactionDateTimestamp object

Child attributes

status object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "farvoriteId": {},  
  "favoritactionDateTimestamp": {},  
  "status": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type [*/*](#)

OK

[*/*](#) ResponseEntity

Child attributes

headers HttpHeaders required

Child attributes

Olha a fimose do papasito aqui!

Box Endpoints

Esta seção descreve os endpoints relacionados ao gerenciamento de caixas (boxes) no sistema.

PUT /box/

Este endpoint permite atualizar uma caixa existente, identificada pelo ID do favorito associado.

PUT /box/{favoriteld}

Request parameters

Path

favoriteld null

Body

application/json BoxRequestDTO **required**

Child attributes

boxId object

Child attributes

price object

Child attributes

title object

Child attributes

cover object

Child attributes

description object

Child attributes

status object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "boxId": {},  
  "price": {},  
  "title": {},  
  "cover": {},  
  "description": {},  
  "status": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders [required](#)

Child attributes

DELETE /box/

Este endpoint permite excluir uma caixa do sistema, identificada pelo ID do favorito associado.

[DELETE](#) /box/{favoritId}

Request parameters

Path

favoritId null

Responses

[200](#) OK

Content type [*/*](#)

OK

[*/*](#) ResponseEntity

Child attributes

headers HttpHeaders [required](#)

Child attributes

GET /box/

Este endpoint retorna todas as caixas cadastradas no sistema.

GET /box/

Responses

200 OK

Content type */*

OK

/* ResponseEntity

Child attributes

headers HttpHeaders required

Child attributes

POST /box/

Este endpoint permite criar uma nova caixa no sistema.

POST /box/

Request parameters

Body

application/json BoxRequestDTO required

Child attributes

boxId object

Child attributes

price object

Child attributes

title object

Child attributes

cover object

Child attributes

description object

Child attributes

status object

Child attributes

userId object

Child attributes

JSON example

```
{  
  "boxId": {},  
  "price": {},  
  "title": {},  
  "cover": {},  
  "description": {},  
  "status": {},  
  "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders **required**

Child attributes

GET /box/

Este endpoint retorna os detalhes de uma caixa específica, identificada pelo seu ID.

GET

/box/{boxId}

Request parameters

Path

boxId object **required**

Child attributes

Responses

200 OK

Content type ***/***

OK

/ ResponseEntity

Child attributes

headers HttpHeaders **required**

Child attributes

Olha a fimose do papasito aqui!

Purchase Endpoints

Endpoints para Purchase

GET purchase/

GET /purchase/{purchaseId}

Request parameters

Path

purchaseId object **required**

Child attributes

Responses

200 OK

Content type */*

OK

/ ResponseEntity

Child attributes

headers HttpHeaders **required**

Child attributes

PUT purchase/

PUT /purchase/{purchaseId}

Request parameters

Path

purchasId object [required](#)

Child attributes

Body

application/json PurchaseRequestDTO [required](#)

Child attributes

purchasId object

Child attributes

email object

Child attributes

rule object

Child attributes

avatar object

Child attributes

purschaseDate object

Child attributes

status object

Child attributes

amount object

Child attributes

userId object

Child attributes

JSON example

```
{  
    "purchaseId": {},  
    "email": {},  
    "rule": {},  
    "avatar": {},  
    "purschaseDate": {},  
    "status": {},  
    "amount": {},  
    "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

 */ ResponseEntity

Child attributes

headers HttpHeaders **required**

Child attributes

POST purchase/

POST /purchase/{purchaseId}

Request parameters

Path

purchaseld object required

Child attributes

Body

application/json PurchaseRequestDTO required

Child attributes

purchaseld object

Child attributes

email object

Child attributes

rule object

Child attributes

avatar object

Child attributes

purschaseDate object

Child attributes

status object

Child attributes

amount object

Child attributes

userId object

Child attributes

JSON example

```
{  
    "purchaseId": {},  
    "email": {},  
    "rule": {},  
    "avatar": {},  
    "purschaseDate": {},  
    "status": {},  
    "amount": {},  
    "userId": {}  
}
```

Responses

200 OK

Content type */*

OK

 */ ResponseEntity

Child attributes

headers HttpHeaders [required](#)

Child attributes

DELETE purchase/

DELETE /purchase/{purchaseId}

Request parameters

Path

purchaseId object **required**

Child attributes

Responses

200 OK

Content type ***/***

OK

/ **ResponseType**

Child attributes

headers **HttpHeaders** **required**

Child attributes

GET purchase/

GET /purchase/

Responses

200 OK

Content type ***/***

OK

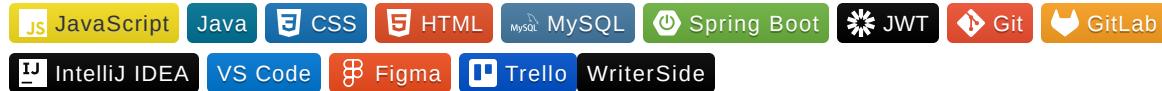
/ **ResponseType**

Child attributes

headers HttpHeaders [required](#)

Child attributes

Tecnologias Utilizadas



Linguagens e Tecnologias

- **JavaScript**: Utilizado para a construção da interface web interativa do sistema, incluindo funcionalidades do lado do cliente.
- **Java**: Empregado no desenvolvimento do backend da aplicação, utilizando o framework Spring Boot para a construção dos serviços e APIs.
- **CSS**: Responsável pela estilização visual dos componentes e layout da interface do usuário.
- **HTML**: Usado para a estruturação semântica das páginas web que compõem a aplicação.
- **Spring Boot**: Framework Java utilizado para a construção rápida e eficiente do backend, fornecendo recursos de injeção de dependência, segurança, persistência de dados, entre outros.
- **JWT (JSON Web Tokens)**: Utilizado para a implementação de autenticação e autorização no sistema, garantindo a segurança das interações entre cliente e servidor.

Banco de Dados

- **MySQL**: Banco de dados relacional escolhido para armazenar e persistir os dados da aplicação de forma estruturada.

Controle de Versão

- **Git**: Sistema de controle de versão adotado para o gerenciamento do código-fonte do projeto.

- **GitLab:** Plataforma de hospedagem e colaboração de projetos Git utilizada para o versionamento do código.

IDEs

- **IntelliJ IDEA:** Ambiente de desenvolvimento integrado (IDE) escolhido para a codificação e debug do backend em Java.
- **Visual Studio Code:** IDE utilizada no desenvolvimento do frontend, provendo ferramentas e extensões para trabalhar com JavaScript, HTML e CSS.

Prototipagem

- **Figma:** Ferramenta de design e prototipagem utilizada para criar os mockups e designs da interface do usuário.

Gerenciamento de Projeto

- **Trello:** Plataforma de gerenciamento de projetos e tarefas empregada para organizar e acompanhar o andamento do desenvolvimento.

Documentação

- **WriterSide:** Ferramenta selecionada para a criação e estruturação da documentação técnica do projeto, visando a facilitar a compreensão e manutenção do sistema.

Conclusão

O projeto atingiu uma parte significativa dos objetivos definidos inicialmente, com a implementação completa de 5 requisitos funcionais (RF01, RF02, RF03, RF05 e RF08) e 7 requisitos não funcionais (RNF01, RNF02, RNF03, RNF04, RNF05, RNF07 e RNF09). Além disso, conseguimos desenvolver a interface front-end e integrar a funcionalidade de criação de conjuntos (ensembles) com o back-end, o que representa um importante avanço na entrega do produto.

Apesar das limitações de tempo e das dificuldades técnicas enfrentadas, como a curva de aprendizado em ferramentas e tecnologias, a equipe demonstrou resiliência e colaboração ao superar esses desafios. O uso do modelo ágil Scrum foi fundamental para o sucesso do projeto, permitindo uma boa divisão de tarefas, revisões constantes, e melhorias contínuas ao longo de cada sprint. A metodologia também nos ajudou a lidar com prazos e promover um ambiente de trabalho estruturado e produtivo.

Durante a execução do projeto, adquirimos habilidades valiosas, como o trabalho em equipe, o hábito de realizar reuniões diárias (dailys) e a capacidade de lidar com prazos e adversidades. Esses aprendizados serão essenciais para futuros projetos e carreiras profissionais.

Em resumo, embora algumas funcionalidades ainda estejam pendentes, o projeto demonstrou um grande progresso, entregando um produto funcional e alinhado aos requisitos priorizados. Este trabalho serviu não apenas como um exercício técnico, mas também como uma experiência de crescimento e aprendizado para todos os envolvidos.

Referencias Bibliográficas

- OMG. (2017). OMG Unified Modeling Language (OMG UML), Version 2.5.1. Object Management Group. <https://www.omg.org/spec/UML/2.5.1/> (<https://www.omg.org/spec/UML/2.5.1/>)
- Asana. (2024). Business Requirements Document Template: 7 Components. Recuperado de <https://asana.com/resources/business-requirements-document-template> (<https://asana.com/resources/business-requirements-document-template>)
- Mozilla Developer Network. MDN Web Docs - HTML (<https://developer.mozilla.org/en-US/docs/Web/HTML>) e CSS (<https://developer.mozilla.org/en-US/docs/Web/CSS>).
- Spring Team. Spring Boot Documentation (<https://docs.spring.io/spring-boot/docs/current/reference/html/>).
- W3schools Java. Java Tutorial (<https://www.w3schools.com/java/>)
- W3Schools. Git Tutorial (<https://www.w3schools.com/git/>).
- Pressman, R. S. (2009). *Ingeniería de Software* 9. Pearson.
- Cris Rúa. Ágil Es - Introdução às Metodologias Ágeis (https://www.youtube.com/watch?v=16loziutCZs&ab_channel=%C3%81gilEs-PorCrisR%C3%BAa).
- Sommerville, I. (2011). *Engenharia de Software*. 10^a edição. OCR.
- AEVO. Metodologia Ágil (<https://blog.aevo.com.br/metodologia-agil/>).
- Pontotel. Metodologia Scrum (<https://www.pontotel.com.br/metodologia-scrum/>).
- Lucidchart. Diagrama de Caso de Uso UML (<https://www.lucidchart.com/pages/pt/diagrama-de-caso-de-uso-uml#discoveryTop>).
- Lucidchart. O Que é Diagrama de Sequência UML (<https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-sequencia-uml>).

- IBM. Sequence Diagrams - UML (<https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=uml-sequence-diagrams>).
- MySQL. MySQL Tutorial Excerpt (<https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.0/en/>).
- W3Schools. JavaScript Tutorial (<https://www.w3schools.com/js/>).