

**CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA
SOUZA**

FATEC PRAIA GRANDE

Desenvolvimento de Software Multiplataforma

Anthony Fernandes do Vale Passos

Erick Nascimento dos Santos

Guilherme Pereira Cardoso de Andrade

SISTEMA DE LOCALIZAÇÃO PARA FEIRAS

Praia Grande – SP

2025

Anthony Fernandes do Vale Passos

Erick Nascimento dos Santos

Guilherme Pereira Cardoso de Andrade

SISTEMA DE LOCALIZAÇÃO PARA FEIRAS

Projeto de Curricularização da Extensão apresentado ao Curso Superior de Tecnologia em Desenvolvimento de Software Multiplataforma da Fatec de Praia Grande, orientado pela Prof.^a Me. Eulaliane Gonçalves, como requisito para aprovação da disciplina de Laboratório de Desenvolvimento Web.

Praia Grande – SP

2025

SUMÁRIO

1 INTRODUÇÃO.....	5
1.1 Tema.....	5
1.2 Objetivos.....	5
1.3 Problematização.....	5
1.4 Justificativa.....	6
1.5 Metodologia.....	6
1.6 Organização do Trabalho.....	6
2 DESCRIÇÃO GERAL DO SISTEMA.....	7
2.1. Descrição do Problema.....	7
2.2. Principais Envolvidos e suas Características.....	8
2.3. Regras de Negócio.....	10
2.4 Ferramentas de Apoio.....	10
3 REQUISITOS DO SISTEMA.....	11
3.1 Requisitos Funcionais.....	12
3.2 Requisitos Não Funcionais.....	13
3.3. Protótipo.....	14
3.3.1. Diagrama de Navegação.....	27
4. ARQUITETURA DO SISTEMA.....	28
4.1. Componentes do Sistema.....	28
4.2 Diagrama de Arquitetura.....	28
4.3. Segurança.....	31
4.4. Escalabilidade.....	32
4.5. Manutenção.....	32
4.6. Integração.....	32
4.7. Limitações.....	33
5. ARQUITETURA DE DADOS.....	33
5.1. Modelo Lógico da Base de Dados.....	33
5.2. Criação Física do Modelo de Dados.....	36
5.3. Dicionário de Dados.....	39

6. PROJETO DE SOFTWARE.....	42
6.1. Design dos Componentes.....	43
6.2. Diagrama de Classes.....	46
6.3. Diagrama de Sequência ou Fluxo de Interações.....	50
6.4. Regras de Negócio Detalhadas.....	54
6.5. Estratégias de Tratamento de Erros.....	57
7 IMPLEMENTAÇÃO.....	58
7.1 Estrutura Geral do Projeto.....	58
7.2 Modelos.....	59
7.2.1 Modelo Feira.....	59
7.2.2 Modelo Barraca.....	60
7.2.3. Modelo PresencaBarracaFeira.....	60
7.2.4. Modelo Usuario.....	60
7.3. Views.....	60
7.3.1. Autenticação com JWT.....	60
7.3.2. Views HTML (Frontend).....	61
7.4. Templates.....	61
7.4.1. Template dashboard.html.....	61
7.4.2. Template feira_detalhes.html.....	61
7.4.3. Template base.html.....	61
7.4.4. Template login.html.....	61
7.4.5. Template register.html.....	62
7.5. Segurança e Criptografia.....	62

1. INTRODUÇÃO

O presente documento tem como objetivo apresentar a documentação do projeto de software desenvolvido no âmbito da disciplina, seguindo as normas da ABNT. O projeto consiste em um site que permite aos usuários localizarem feiras próximas. O documento seguirá uma estrutura clara e organizada para detalhar todas as fases do desenvolvimento do produto.

1.1 Tema

O projeto tem como tema a criação de uma plataforma online para facilitar o acesso dos consumidores a feiras locais, permitindo a visualização da localização delas.

1.2 Objetivos

O objetivo geral do projeto é proporcionar uma solução digital que otimize a experiência dos consumidores ao frequentarem feiras locais, reduzindo tempo de busca e aumentando a conveniência.

Os objetivos específicos incluem:

- Desenvolver um site responsivo e intuitivo;
- Implementar um sistema de geolocalização para identificar feiras próximas.

1.3 Problematização

O problema identificado é a dificuldade que consumidores enfrentam para encontrar feiras locais devido à falta de informações centralizadas sobre suas localizações e horários de funcionamento.

1.4 Justificativa

A escolha desse projeto se justifica pela necessidade de modernização do acesso às feiras livres, proporcionando maior praticidade para os consumidores. A implementação de uma plataforma digital pode facilitar a busca por feiras próximas, incentivando sua frequência e contribuindo para o fortalecimento do comércio local.

1.5 Metodologia

O desenvolvimento do projeto seguirá a metodologia ágil Scrum, um framework iterativo e incremental que permite a entrega contínua de funcionalidades por meio de ciclos curtos chamados sprints.

No Scrum, há três papéis principais:

Product Owner (PO): responsável por definir as prioridades do backlog do produto e garantir que o sistema atenda às necessidades dos usuários.

Scrum Master: atua como facilitador, removendo impedimentos e garantindo que a equipe siga os princípios do Scrum.

Time de Desenvolvimento: equipe multidisciplinar responsável pela implementação das funcionalidades dentro dos sprints.

A modelagem será orientada a objetos, empregando UML para representação do sistema. O site será desenvolvido utilizando tecnologias web modernas, garantindo responsividade e segurança.

1.6 Organização do Trabalho

Este documento está estruturado da seguinte forma:

Capítulo 1 - Introdução: apresenta a introdução ao projeto, abordando tema, objetivos, problematização, justificativa, metodologia e organização do trabalho;

Capítulo 2 - Descrição Geral do Sistema: Detalha o escopo do sistema, os problemas que ele resolve, os principais envolvidos e as regras de negócio.

Capítulo 3 - Navegação entre Telas: apresenta o diagrama de navegação entre as telas da aplicação, demonstrando visualmente como o usuário interage com o sistema e como as transições ocorrem entre as interfaces;

Capítulo 4 - Arquitetura do Sistema: descreve a arquitetura escolhida para o desenvolvimento do software, incluindo os componentes do sistema, diagrama de arquitetura, considerações de segurança, escalabilidade, manutenção, integração e limitações da arquitetura.

Capítulo 5 - Arquitetura de Dados: Este capítulo tem como objetivo descrever os fundamentos e diretrizes para a estruturação e gerenciamento dos dados dentro de um software.

Capítulo 6 - Projeto de Software: Este capítulo tem como objetivo detalhar o design interno dos componentes do sistema com base na arquitetura já definida, preparando a equipe para a fase de implementação. Ele detalha como as partes do sistema interagem, quais responsabilidades cada componente tem e como a lógica será distribuída.

Capítulo 7 - Implementação: apresenta como as funcionalidades do sistema foram implementadas utilizando o framework Django, destacando a estrutura modular, os modelos principais do sistema, a separação entre views REST e HTML, a integração com autenticação JWT, além da construção dos templates e práticas de segurança aplicadas.

2. DESCRIÇÃO GERAL DO SISTEMA

Este capítulo descreve de forma geral o sistema, seu escopo e suas principais funções. A descrição geral do sistema abrange os itens a seguir.

2.1 Descrição do Problema

O sistema visa resolver a dificuldade de acesso a informações sobre a localização de feiras livres, proporcionando aos usuários uma ferramenta digital acessível e eficiente. Consumidores que frequentam feiras livres frequentemente enfrentam desafios para encontrar feiras próximas de maneira rápida e prática, especialmente devido à falta de uma plataforma centralizada com essas informações. A implementação desse sistema terá um impacto positivo, otimizando a experiência dos usuários ao reduzir o tempo gasto na busca por feiras, além de incentivar a

economia regional ao promover o acesso facilitado às feiras locais. Para solucionar esse problema, a proposta consiste no desenvolvimento de uma plataforma responsiva e intuitiva, que utilize tecnologia de geolocalização para fornecer informações atualizadas sobre a localização e funcionamento das feiras.

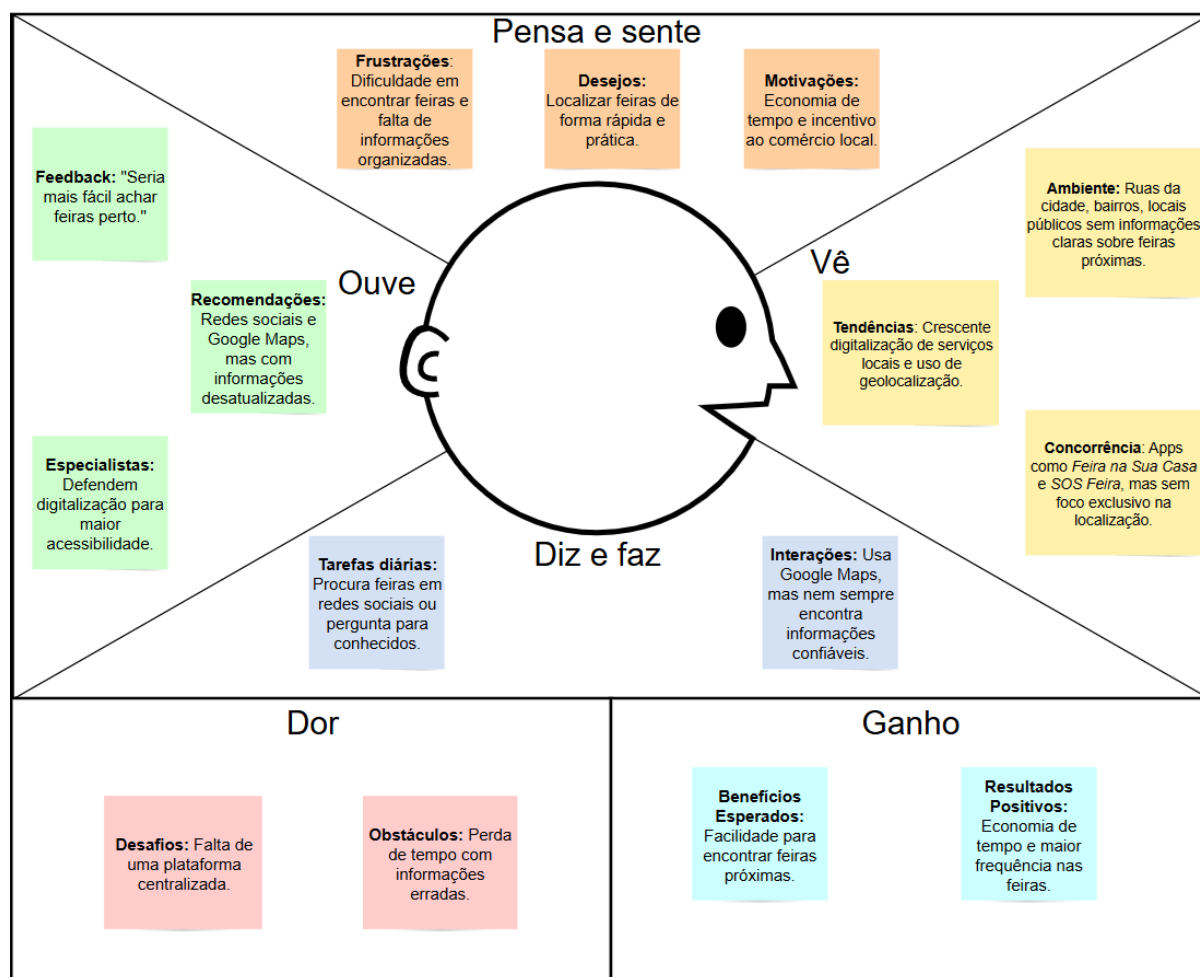
2.2 Principais Envolvidos e suas Características

Usuários do Sistema:

O sistema se destina a consumidores que frequentam feiras livres e buscam um meio digital para localizá-las. A plataforma será voltada para usuários da cidade de **Praia Grande - SP**, permitindo que residentes e visitantes encontrem feiras próximas de forma prática e eficiente.

O Mapa da empatia identifica as necessidades dos usuários, revelando a dificuldade de encontrar feiras livres devido à falta de informações confiáveis. Atualmente, eles dependem de redes sociais e Google Maps, perdendo tempo com dados desatualizados. O sistema proposto visa oferecer uma solução intuitiva e precisa, usando geolocalização para facilitar o acesso a feiras próximas e incentivar o comércio local.

Mapa de Empatia: Consumidores



Desenvolvedores Back-End e Front-End são responsáveis por diferentes áreas do projeto, garantindo uma abordagem estruturada e eficiente, os membros são:

Anthony Fernandes - Documentação/banco de dados

Erick Santos – Front-end/banco de dados

Guilherme Pereira – Back-end/banco de dados

2.3 Regras de Negócio

O sistema seguirá as seguintes regras de negócio:

Restrições de negócio: O sistema deve ser de acesso gratuito aos usuários.

Restrições de desempenho: O tempo de resposta para exibição das feiras não deve ser demorado.

Tolerância a falhas: O sistema deve ter um mecanismo de recuperação para erros inesperados.

2.4 Ferramentas de Apoio

Python com Django: Python é uma linguagem de programação versátil e amplamente utilizada no desenvolvimento web. Django, por sua vez, é um framework que permite a criação de aplicações robustas e seguras, oferecendo recursos como ORM para manipulação do banco de dados, autenticação integrada e um sistema eficiente de templates.

SQLite: Um banco de dados relacional leve e incorporado, ideal para aplicações de pequeno a médio porte. No projeto, o SQLite será utilizado para armazenar informações sobre as feiras, como localização, horários de funcionamento e dados dos usuários. Ele se integra facilmente com o Django, tornando o desenvolvimento mais ágil e prático.

Bootstrap: Um framework CSS que facilita a criação de interfaces modernas e responsivas, garantindo que o site seja acessível e visualmente agradável em diferentes dispositivos.

API Nominatim: fornecida pelo projeto **OpenStreetMap**, é uma ferramenta de geocodificação gratuita que permite converter endereços em coordenadas geográficas (latitude e longitude) e vice-versa. Ela é amplamente utilizada em aplicações que necessitam de localização geográfica sem depender de serviços pagos.

PythonAnywhere: Serviço de hospedagem em nuvem voltado para aplicações Python. Permite rodar projetos Django diretamente no servidor, com suporte nativo ao SQLite e fácil publicação via navegador. No projeto, será utilizado como **serviço de hospedagem**, viabilizando o acesso online da aplicação. Oferece também

terminal, agendador de tarefas e painel de administração, sendo ideal para projetos acadêmicos e de pequeno porte.

Figma: será utilizado para criar e prototipar as telas do sistema, permitindo a visualização prévia do design e facilitando a colaboração entre os desenvolvedores em tempo real.

HTML: HTML (HyperText Markup Language) é a linguagem de marcação utilizada para estruturar o conteúdo de páginas web. No projeto, o HTML será responsável por definir os elementos que compõem a interface do usuário, como formulários, botões, campos de entrada, menus e divisões de conteúdo. Sua função é organizar semanticamente a informação exibida no navegador, garantindo que o sistema seja compreendido tanto por humanos quanto por máquinas, como navegadores e mecanismos de busca.

CSS: CSS (Cascading Style Sheets) é a linguagem de estilo utilizada para definir a aparência visual das páginas web. No projeto, o CSS será responsável por aplicar cores, fontes, tamanhos, espaçamentos, alinhamentos e demais aspectos estéticos aos elementos HTML. Ele permite o desenvolvimento de interfaces atraentes, responsivas e consistentes, sendo também a base para a utilização de frameworks visuais como o Bootstrap.

3. REQUISITOS DO SISTEMA: este capítulo tem como objetivo descrever os requisitos do sistema.

Níveis de Prioridade:

Cada requisito é classificado com um nível de prioridade, para guiar o desenvolvimento com base em sua importância para o funcionamento do sistema. As prioridades adotadas são:

- **Crítico:** Essencial para o funcionamento mínimo do sistema. Sem esse requisito, a aplicação não atinge seu propósito principal.
- **Desejável:** Requisitos que não são essenciais, mas melhoram significativamente a experiência do usuário ou a eficiência do sistema.

- **Opcional:** Funcionalidades que agregam valor, mas que podem ser implementadas futuramente, caso haja tempo ou recursos disponíveis.

3.1 Requisitos Funcionais:

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer. Em outras palavras, representam tudo aquilo que o sistema deve fazer para atender às necessidades dos usuários e cumprir os objetivos propostos. São ações, comportamentos e reações esperadas da aplicação frente a determinadas interações.

ID	Descrição	Prioridade
RF01	O sistema deve permitir que os usuários pesquisem feiras locais por localização.	Crítico (1)
RF02	O sistema deve exibir a localização das feiras em um mapa interativo.	Crítico (1)
RF03	O sistema deve permitir a filtragem de feiras por dia da semana e horário de funcionamento.	Desejável (2)
RF04	O sistema deve fornecer detalhes das feiras, como endereço, horário.	Crítico (1)
RF05	O sistema deve ser acessível por diferentes dispositivos (desktop e mobile).	Desejável (2)
RF06	O sistema deve oferecer uma interface intuitiva para navegação e pesquisa.	Desejável (2)
RF07	O sistema deve possibilitar que usuários favoritem feiras para acesso rápido.	Opcional (3)
RF08	O sistema deve permitir avaliações e comentários dos usuários sobre as feiras.	Opcional (3)
RF09	O sistema deve enviar notificações aos usuários sobre novas feiras cadastradas próximas à sua localização.	Opcional (3)
RF10	O sistema deve permitir o cadastro de novos usuários com nome, endereço, e-mail e senha.	Crítico (1)
RF11	O sistema deve validar e-mails antes de concluir o cadastro.	Desejável (2)
RF12	O sistema deve permitir que usuários façam login com e-mail e senha.	Crítico (1)

RF14	O sistema deve permitir a recuperação de senha via e-mail.	Desejável (2)
------	--	----------------------

3.2 Requisitos Não Funcionais:

Os requisitos não funcionais referem-se a restrições ou qualidades que o sistema deve possuir. Isso inclui aspectos como desempenho, segurança, usabilidade, compatibilidade com dispositivos e navegadores, entre outros. Diferente dos requisitos funcionais, os não funcionais não descrevem "o que" o sistema faz, mas sim "como" ele deve se comportar.

ID	Descrição	Prioridade
RNF01	O sistema deve ter um tempo de resposta inferior a 2 segundos para buscas de feiras.	Crítico (1)
RNF02	O sistema deve ser compatível com os navegadores mais populares (Chrome, Firefox, Edge).	Desejável (2)
RNF03	O sistema deve criptografar senhas dos usuários utilizando bcrypt ou Django's PBKDF2.	Crítico (1)
RNF04	O sistema deve armazenar os dados em um banco de dados MongoDB, utilizando Django com Djongo ou MongoEngine.	Desejável (2)
RNF05	O sistema deve seguir as diretrizes de acessibilidade WCAG 2.1 para garantir inclusão.	Desejável (2)
RNF06	O sistema deve utilizar autenticação baseada em tokens JWT para login e persistência de sessão.	Crítico (1)
RNF07	O sistema deve permitir integração com serviços de mapas como Google Maps ou OpenStreetMap.	Desejável (2)

RNF08	O sistema deve ser responsivo e adaptável para dispositivos móveis e desktops.	Crítico (1)
-------	--	--------------------

3.3. Protótipo:

Nesta seção, serão apresentadas as telas que compõem o protótipo do sistema proposto. Cada tela foi projetada com o objetivo de garantir uma navegação simples, eficiente e acessível, atendendo às necessidades dos usuários e às funcionalidades estabelecidas nos requisitos.

Tela de Cadastro:



O protótipo da tela de cadastro é apresentado em um formato de wireframe. No topo, há uma barra de navegação escura com um ícone de menu hambúrguer. Abaixo, o formulário é centralizado em um fundo verde claro com cantos arredondados. O formulário possui o título 'Cadastro' no topo. Seguem os campos de entrada: 'Usuário' (com o placeholder 'Digite seu nome de usuário'), 'E-mail' (com o placeholder 'Digite seu e-mail'), 'Senha' (com o placeholder 'Digite sua senha') e 'Confirmar Senha' (com o placeholder 'Digite sua senha novamente'). Abaixo dos campos, há um link azul 'já possui uma conta? clique aqui!'. No final, há um botão amarelo com o texto 'Cadastrar'.

Objetivo da Tela:

A tela de cadastro permite que novos usuários criem uma conta no sistema de localização de feiras. Essa conta será necessária para acessar funcionalidades personalizadas, como visualização de feiras próximas com base na localização do usuário.

Como chegar até ela:

O usuário pode acessar a tela de cadastro a partir da tela de login, clicando no link "Clique aqui" presente abaixo dos campos de login, ou diretamente pelo menu lateral do sistema.

Componentes e Regras da Tela:

Campo	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações
Usuário	Texto	30 caracteres	Sim	Vazio	Deve ser único no sistema.
E-mail	E-mail	100 caracteres	Sim	Vazio	Deve conter formato válido de e-mail.
Senha	Senha	20 caracteres	Sim	Vazio	Mínimo 6 caracteres.
Confirmar Senha	Senha	20 caracteres	Sim	Vazio	Deve ser igual ao campo "Senha".

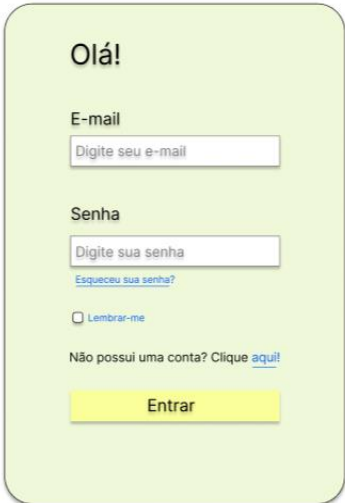
Ações:

- Botão **Cadastrar**: Envia os dados informados para validação e criação da conta.
- Link “**já possui uma conta? clique aqui!**”: Redireciona o usuário para a tela de login.

Usuários que podem acessar:

Essa tela pode ser acessada por qualquer visitante do sistema, ou seja, por usuários ainda não autenticados que desejam se registrar.

Tela de Login:



A login form mockup with a light green background. At the top, it says "Olá!". Below that is the "E-mail" section with a text input field containing the placeholder "Digite seu e-mail". The "Senha" section has a text input field with the placeholder "Digite sua senha", a blue link "Esqueceu sua senha?", and a checkbox labeled "Lembrar-me". Below these is the text "Não possui uma conta? Clique [aqui!](#)". At the bottom is a yellow "Entrar" button.

Objetivo da Tela:

A tela de login permite que usuários previamente cadastrados acessem o sistema de localização de feiras, autenticando-se com suas credenciais. Após o login, o usuário poderá visualizar feiras próximas, acessar o perfil e utilizar outras funcionalidades restritas.

Como chegar até ela:

É a tela padrão inicial do sistema para usuários não autenticados. Também pode ser acessada a partir de outras telas públicas, como a tela de cadastro, por meio do link "Clique aqui" para quem já possui uma conta.

Componentes e Regras da Tela:

Campo	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações
E-mail	E-mail	100 caracteres	Sim	Vazio	Deve conter um formato válido.
Senha	Senha	20 caracteres	Sim	Vazio	Deve corresponder à senha cadastrada.
Lembrar-me	Checkbox	-	Não	Desmarcado	Salva as credenciais localmente (opcional).

Ações:

- **Entrar:** Verifica as credenciais e, se válidas, redireciona o usuário à tela principal.
- **Esqueceu sua senha?:** Link que direciona para o processo de recuperação de senha.
- **Não possui uma conta? Clique aqui!:** Redireciona para a tela de cadastro.

Usuários que podem acessar:

Todos os usuários do sistema. Visitantes que ainda não possuem uma conta devem utilizar o link para realizar o cadastro.

Tela recuperação de senha:

Redefinição de senha

Informe um email e enviaremos um link de recuperação de senha.

E-mail

Digite seu e-mail

Enviar link de recuperação

Objetivo da Tela:

Permitir que o usuário recupere o acesso ao sistema em caso de esquecimento da senha. Através desta tela, é possível solicitar o envio de um link para redefinição da senha ao e-mail previamente cadastrado.

Como chegar até ela:

Esta tela é acessada por meio do link "Esqueceu sua senha?" disponível na tela de login.

Componentes e Regras da Tela:

Campo	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações
E-mail	E-mail	100 caracteres	Sim	Vazio	Deve ser um e-mail válido e já cadastrado no sistema.

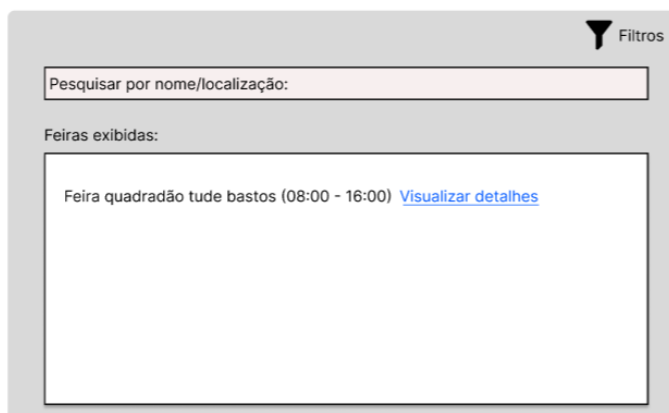
Ações:

- **Enviar link de recuperação:** Ao clicar, o sistema valida o e-mail informado e, se for válido, envia uma mensagem com o link de redefinição de senha.

Usuários que podem acessar:

Todos os usuários que esqueceram a senha e possuem um e-mail cadastrado no sistema.

Tela de Pesquisa:



Objetivo da Tela:

Permitir que o usuário pesquise feiras próximas ou específicas por nome ou localização, exibindo os resultados de forma clara e acessível.

Como chegar até ela:

A tela pode ser acessada através do menu lateral após o login no sistema.

Componentes e Regras da Tela:

Campo	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações
Campo de pesquisa (texto)	Texto	100 caracteres	Não	Vazio	Realiza a busca por nome ou localização da feira.

Elementos e Funcionalidades:

- **Botão de Filtros:** (ícone de funil) permite ao usuário aplicar filtros específicos à busca (ex: por dia da semana, horário, bairro, etc.).
- **Resultado da pesquisa:** Lista com nome da feira, horário de funcionamento e link para **Visualizar detalhes** da feira.

- **Ícone de perfil (canto superior direito):** Acesso a configurações ou informações do usuário logado.

Usuários que podem acessar:

Todos os usuários autenticados no sistema.

Tela perfil do usuário:



Objetivo da Tela:

Exibir as informações pessoais do usuário logado, bem como a lista de feiras marcadas como favoritas, permitindo também desfavoritá-las.

Como chegar até ela:

Clicando no ícone de perfil (canto superior direito da tela).

Componentes e Regras da Tela:

Campo/Elemento	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações / Observações
Nome do usuário	Texto	100 caracteres	Sim	Nome do usuário logado	Informativo, não editável
Email	Texto (e-mail)	100 caracteres	Sim	E-mail do usuário logado	Informativo, não editável
Lista de feiras favoritas	Lista de textos	-	Não	-	Lista dinâmica de feiras marcadas como favoritas
Link "Desfavoritar"	Link (ação)	-	Não	-	Remove a feira da lista de favoritas do usuário

Usuários que podem acessar:

Apenas o usuário autenticado (dados próprios).

Tela de Detalhes da Feira:



[Voltar](#)[Ver avaliações](#)

Feira quadradão tude bastos (08:00 - 16:00)[Favoritar feira](#)

Detalhes da feira:

Possui barracas de venda de frutas, pastel.....

[Clique aqui para abrir o mapa](#)

Objetivo da Tela:

Exibir informações detalhadas sobre uma feira específica, incluindo localização, descrição e funcionalidades extras como visualização no mapa, avaliações e possibilidade de favoritar.

Como chegar até ela:

Através da opção “**Visualizar detalhes**” na **Tela de Pesquisa**.

Componentes e Regras da Tela:

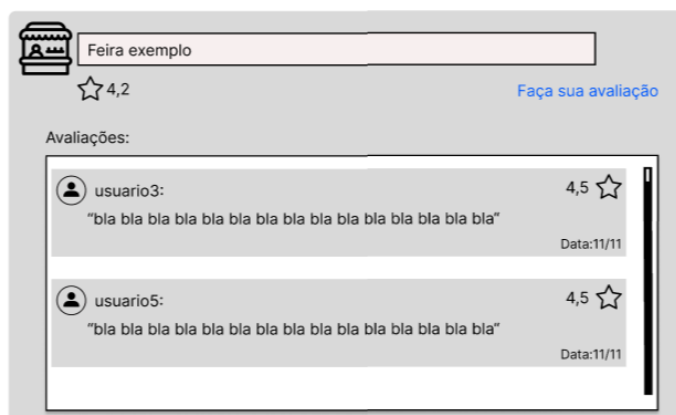
Campo/Elemento	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações / Observações
Nome da feira e horário	Texto	100 caracteres	Sim	Nome da feira selecionada	Campo informativo, não editável

Descrição da feira	Texto longo	500 caracteres	Não	Texto com os tipos de barracas	Ex: frutas, verduras, pastel, etc.
Link "Clique aqui para abrir o mapa"	Link	-	Não	-	Redireciona para a geolocalização da feira no mapa
Botão "Favoritar feira"	Botão	-	Não	-	Adiciona feira aos favoritos do usuário
Link "Ver avaliações"	Link	-	Não	-	Redireciona para a tela de avaliações da feira
Botão "Voltar"	Botão	-	Não	-	Retorna à tela anterior (Pesquisa)

Usuários que podem acessar:

Todos os usuários autenticados.

Tela avaliação:



Objetivo da Tela:

Exibir as avaliações feitas por usuários sobre uma feira específica, incluindo nota, comentário e data, além de permitir o redirecionamento para avaliar a feira.

Como chegar até ela:

Através do link **“Ver avaliações”** na **Tela de Detalhes da Feira**.

Componentes e Regras da Tela:

Campo/Elemento	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações / Observações
Nome da feira	Texto	100 caracteres	Sim	Nome da feira visualizada	Campo informativo, não editável
Nota média da feira	Número (float)	3 caracteres	Sim	Ex: 4.2	Exibida ao lado da estrela e do botão "Favoritar feira"

Botão "Favoritar feira"	Botão	-	Não	-	Adiciona feira aos favoritos
Link "Faça sua avaliação"	Link	-	Não	-	Redireciona para a tela de avaliação
Lista de avaliações	Lista (scroll)	-	Não	Avaliações feitas	Cada item inclui nome do usuário, nota, comentário e data
Nome do usuário (na avaliação)	Texto	50 caracte res	Sim	Nome cadastrado	Exibido junto ao comentário
Nota da avaliação	Numé rico (float)	3 caracte res	Sim	De 0.0 a 5.0	Exibida com estrela
Comentário da avaliação	Texto	200 caracte res	Não	-	Texto livre
Data da avaliação	Data	-	Sim	Formato dd/mm	Exibido no canto inferior direito de cada avaliação

Usuários que podem acessar:

Todos os usuários autenticados.

Tela avaliação (usuário):



Objetivo da Tela:

Permitir que o usuário escreva uma avaliação e dê uma nota de 0 a 5 estrelas para uma feira específica.

Como chegar até ela:

Através do link **“Faça sua avaliação”** na **Tela de Avaliações da Feira**.

Componentes e Regras da Tela:

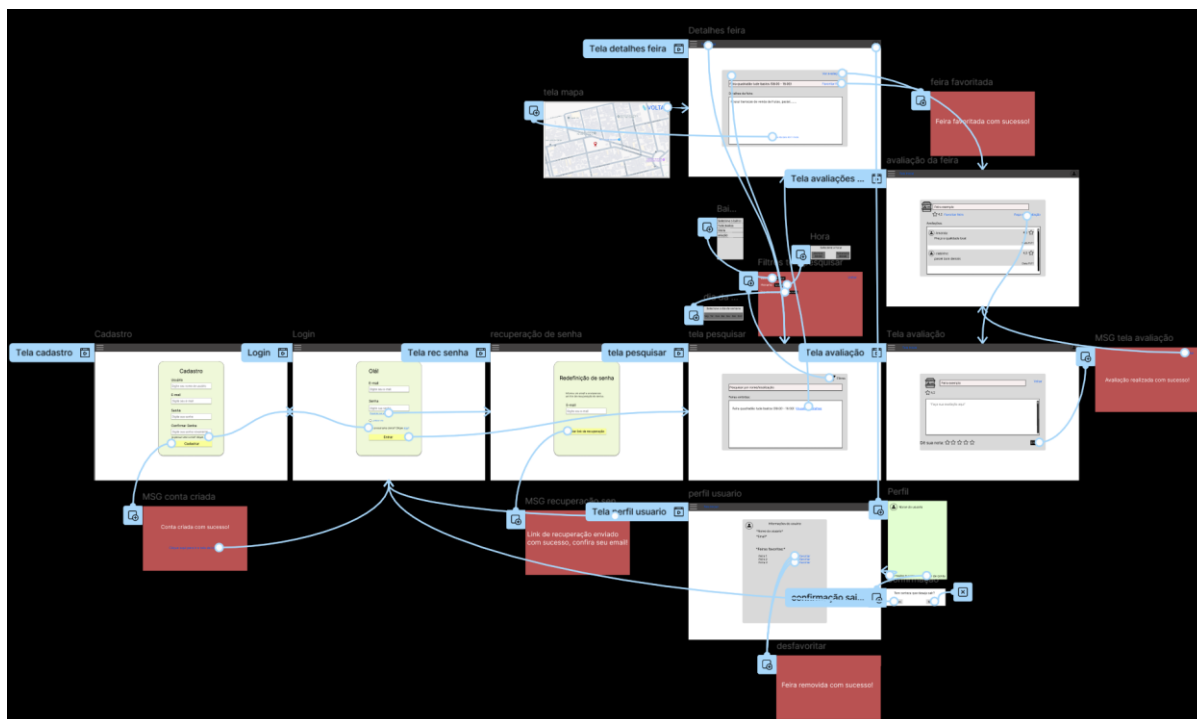
Campo/Elemento	Tipo de dado	Tamanho máximo	Obrigatório	Valor default	Validações / Observações
Nome da feira	Texto	100 caracteres	Sim	Nome da feira visualizada	Campo informativo, não editável
Nota média da feira atual	Numérico (float)	3 caracteres	Sim	Ex: 4.2	Informativa, não editável

Campo de avaliação	Texto multilinha	200 caracteres	Não	Placeholder: "Faça sua avaliação aqui"	Comentário opcional, mas limitado
Campo de nota (estrelas)	Númerico (1 a 5)	1 caractere	Sim	Nenhum selecionado	Obrigatório selecionar pelo menos 1 estrela
Botão "Enviar"	Botão	-	Sim	-	Valida se a nota foi dada antes de enviar
Link "Voltar"	Link	-	Não	-	Retorna para a tela anterior (Avaliações da feira)

Usuários que podem acessar:

Usuários autenticados.

3.3.1. Diagrama de Navegação:



4. ARQUITETURA DO SISTEMA

A arquitetura escolhida para o sistema foi a **arquitetura em camadas**, uma abordagem tradicional e amplamente utilizada no desenvolvimento de aplicações web, que promove a separação de responsabilidades entre os diferentes componentes da aplicação. Essa escolha visa facilitar a manutenção, a escalabilidade e o desenvolvimento colaborativo da solução.

A arquitetura foi dividida nas seguintes camadas principais: **apresentação (front-end)**, **lógica de negócio (back-end)**, e **persistência de dados (banco de dados)**. A comunicação entre essas camadas é feita por meio de requisições HTTP e APIs REST.

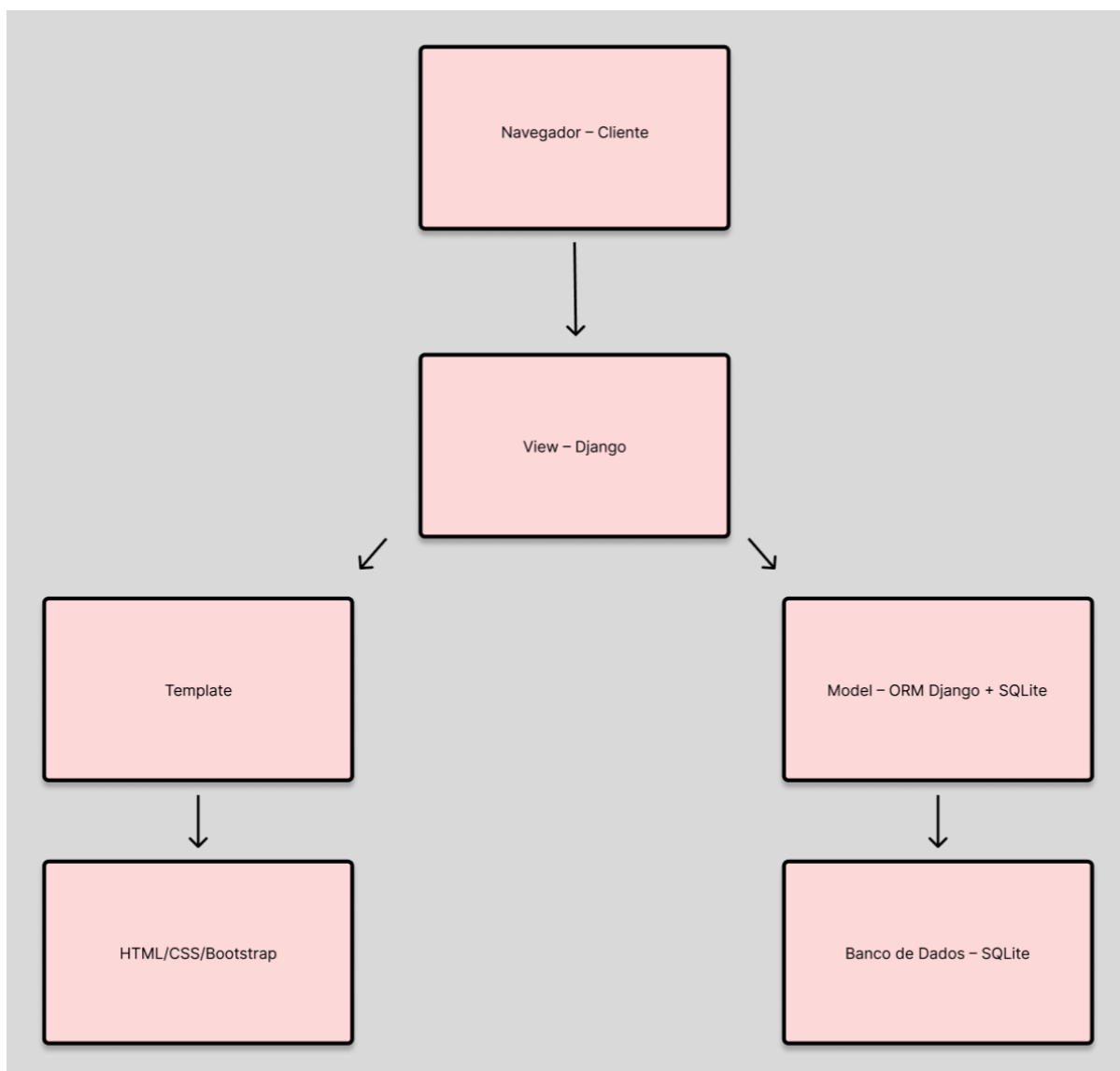
4.1. Componentes do Sistema

- **Front-end (Apresentação):**
Desenvolvido com HTML, CSS e JavaScript, com apoio de frameworks como Bootstrap. Essa camada é responsável pela interface com o usuário, possibilitando ações como login, cadastro, visualização de feiras no mapa e avaliação das mesmas.
- **Back-end (Lógica de Negócio):**
Implementado em **Django (Python)**, o back-end lida com toda a lógica do sistema, incluindo autenticação, manipulação de dados, regras de negócio e comunicação com a base de dados.
- **Banco de Dados:**
Utiliza-se o **SQLite**, um banco de dados relacional leve e embutido, que oferece integração nativa com o Django. É ideal para aplicações de pequeno porte e protótipos funcionais, facilitando a configuração e o desenvolvimento local sem a necessidade de servidores externos.
- **APIs e Integrações:**
Serão utilizadas APIs de geolocalização para detectar a posição atual do usuário, como a API Nominatim do projeto OpenStreetMap, além de possíveis integrações futuras com serviços públicos de cadastro de feiras.

4.2. Diagrama de Arquitetura

Título: Diagrama de Arquitetura em Camadas do Sistema Feira Livre

A arquitetura do sistema segue o padrão MVT (Model-View-Template) utilizado pelo Django. Essa abordagem promove a separação de responsabilidades entre a camada de dados (Model), a lógica de controle (View) e a apresentação (Template), favorecendo a reutilização de código, organização e facilidade de manutenção do sistema.



Acesso Uniforme:

- **Função:** Serve como ponto único de entrada para o sistema, seja por desktop ou mobile.
- **Responsabilidade:** Centralizar o login e a navegação inicial, garantindo uma experiência consistente ao usuário.

- **Importância:** Ajuda na padronização da interface e facilita a autenticação e controle de acesso.

Portal Web (via navegador), compatível com mobile e desktop:

- **Função:** Representa o meio pelo qual os usuários interagem com o sistema.
- **Responsabilidade:** Adaptar-se a diferentes tamanhos de tela e garantir acessibilidade.
- **Tecnologia associada:** Navegadores modernos.

Acesso ao Sistema:

- **Função:** Indica a entrada na aplicação após o login.
- **Responsabilidade:** Permitir que o usuário navegue pelas funcionalidades internas do sistema (buscar feiras, favoritar, avaliar etc).

Interface do Usuário:

- **Componentes:** HTML, CSS, Bootstrap
- **Função:** Estruturar (HTML), estilizar (CSS) e tornar responsiva (Bootstrap) a interface do sistema.
- **Responsabilidade:** Exibir os elementos visuais como formulários, listas de feiras, botões, etc.
- **Papel na arquitetura:** Apresentação – primeira camada que interage com o usuário.

Lógica de Negócio:

- **Componente:** Django (Python)
- **Função:** Controlar e processar regras do sistema, como autenticação, validações, interações com o banco de dados.
- **Responsabilidade:** Gerenciar o fluxo de dados entre o front-end e o banco de dados.
- **Papel na arquitetura:** Coração da aplicação, trata tudo que não é visual.

APIs e Serviços:

- **Componente:** API de Geolocalização
- **Função:** Identificar a localização do usuário.

- **Responsabilidade:** Fornecer dados para exibir feiras próximas com base na posição geográfica.
- **Observação:** Pode usar a API de geolocalização do navegador (HTML5) ou serviços como OpenStreetMap/Google Maps.

Banco de Dados:

- **Componente:** SQLite
- **Função:** Armazenar dados da aplicação como usuários, feiras, avaliações e favoritos.
- **Responsabilidade:** Garantir persistência e consistência dos dados.
- **Justificativa:** Leve, integrado ao Django, ideal para projetos pequenos e médios.

Segurança:

- **Componentes:**
 - **Autenticação com e-mail e senha:** Garante que apenas usuários registrados tenham acesso.
 - **Autorização para ações restritas:** Exemplo: favoritar feiras, avaliar.
 - **HTTPS para comunicação segura:** Protege os dados trafegados entre cliente e servidor.
- **Responsabilidade:** Assegurar privacidade, integridade e controle de acesso.

4.3. Segurança

A arquitetura contempla os seguintes mecanismos de segurança:

- **Autenticação:** Sistema de login com validação de e-mail e senha.
- **Autorização:** Controle de acesso a funcionalidades restritas a usuários logados (ex: favoritar feiras, avaliar).
- **Proteção de Dados:** As senhas dos usuários são armazenadas de forma criptografada no banco de dados utilizando algoritmos de hash (como PBKDF2).

- **Comunicação Segura:** O sistema será hospedado com suporte a HTTPS para garantir comunicação criptografada entre cliente e servidor.

4.4. Escalabilidade

- A escolha por Django e SQLite atende bem às necessidades de aplicações de pequeno porte ou protótipos. A escalabilidade poderá ser aprimorada futuramente com substituição do banco de dados, caso necessário:
- O back-end Django permite separação modular dos serviços, o que facilita a criação de APIs independentes e integração com bancos de dados mais robustos, como PostgreSQL ou MySQL, em caso de crescimento da aplicação.
- A arquitetura pode ser migrada facilmente para serviços em nuvem (como AWS, Render ou Railway) com possibilidade de uso de bancos de dados externos, balanceamento de carga e múltiplas instâncias.

4.5. Manutenção

O sistema foi projetado para facilitar atualizações por meio de:

- Separação clara entre camadas de front-end, back-end e banco de dados;
- Uso de **repositórios versionados** (ex: GitHub), o que facilita controle de versão e colaboração;
- Utilização de **componentes reutilizáveis** na interface;
- Planejamento de testes unitários e de integração com Django.

4.6. Integração

- O sistema utilizará a **API de geolocalização do navegador** para identificar a posição do usuário;
- Há planejamento para futura integração com **sistemas de dados públicos**, como APIs municipais ou estaduais de feiras livres;

- A arquitetura está preparada para integração com **gateways de notificações** (ex: envio de lembretes por e-mail) se necessário.

4.7. Limitações

- O sistema requer **acesso à internet** e permissões de geolocalização para funcionamento completo;
- Há dependência de **navegadores modernos** para o uso da API de localização;
- A hospedagem gratuita inicial pode impor **limites de uso** (como número de requisições ou espaço de armazenamento);
- Restrições de privacidade e **regulamentações sobre proteção de dados** (ex: LGPD) devem ser observadas no armazenamento de dados pessoais.

5. ARQUITETURA DE DADOS

Este capítulo apresenta a estrutura de dados do sistema proposto para localização de feiras livres, com o objetivo de garantir a organização, integridade e eficiência no armazenamento e manipulação das informações. A arquitetura de dados foi planejada de forma a atender aos requisitos funcionais do sistema, considerando a tecnologia utilizada no desenvolvimento (Django + SQLite) e os princípios de normalização de banco de dados.

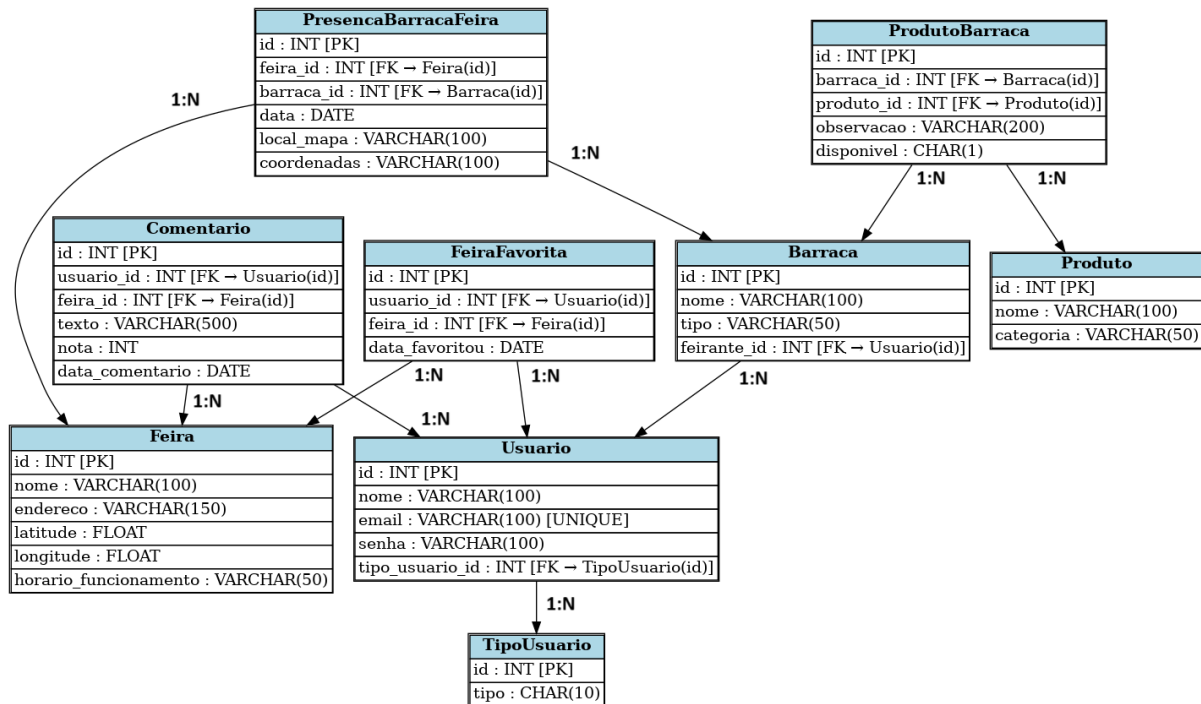
5.1. Modelo Lógico da Base de Dados

O modelo lógico foi desenvolvido com base no modelo Entidade-Relacionamento (ER), representando de forma clara as entidades, seus atributos e os relacionamentos entre elas. As entidades principais do sistema são:

- **Usuário:** armazena dados de autenticação e identificação dos usuários.
- **Feira:** representa cada feira cadastrada, incluindo nome, localização e horários.
- **FavoritoFeira:** registra as feiras que foram marcadas como favoritas por um usuário.

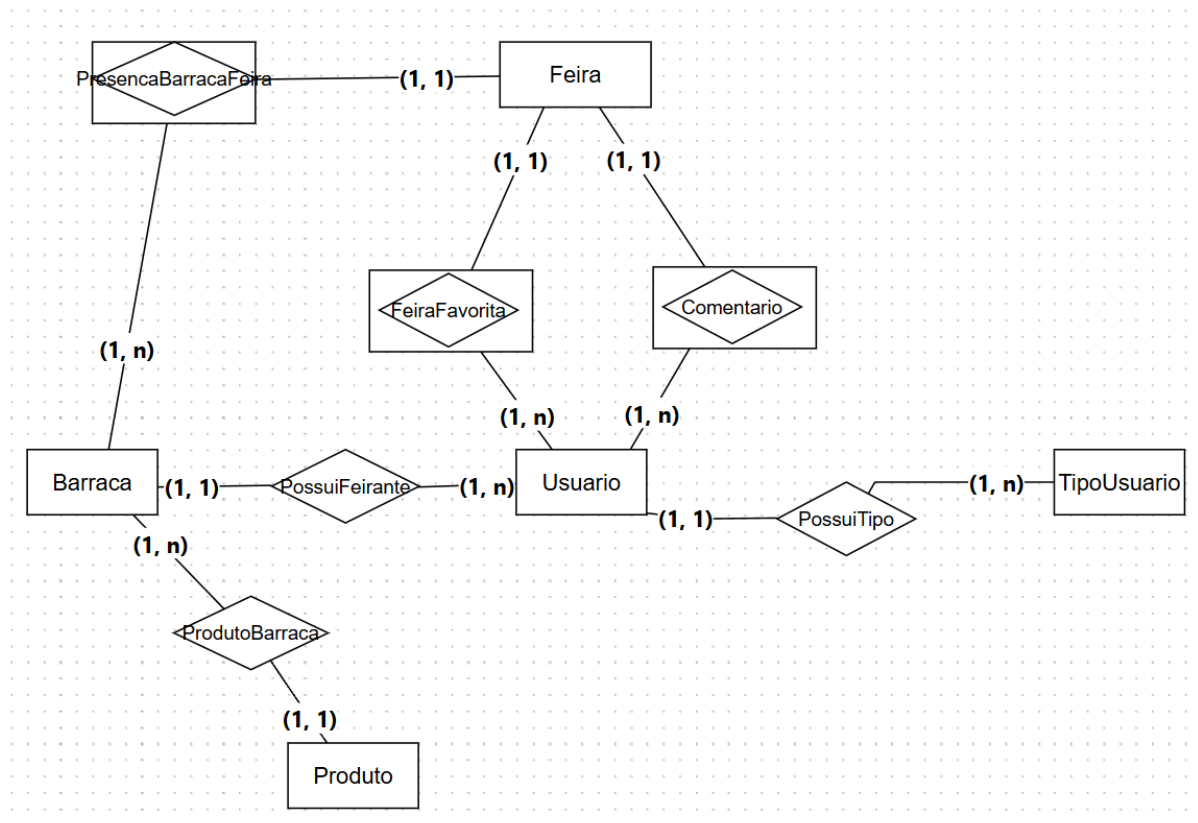
- **ComentarioFeira:** armazena as avaliações e notas que os usuários atribuem às feiras.
- **Produto:** armazena os dados dos produtos disponíveis para venda nas barracas, como nome e categoria. Essa entidade permite o gerenciamento e categorização dos itens comercializados nas feiras.
- **Barraca:** representa cada barraca registrada por um feirante. Contém informações como nome e tipo, e está associada a um usuário do tipo “feirante”, permitindo a identificação do responsável.
- **ProdutoBarraca:** é uma entidade associativa que vincula os produtos às barracas. Além de representar a disponibilidade de determinado produto em uma barraca específica, armazena observações adicionais e se o item está disponível.
- **PresencaBarracaFeira:** registra a presença de uma barraca em uma feira em determinada data, bem como sua posição no mapa. Essa entidade permite controlar dinamicamente a participação dos feirantes em diferentes feiras ao longo do tempo.
- **TipoUsuario:** define os diferentes perfis de acesso do sistema, como “admin”, “usuario” e “feirante”. Está ligada à entidade Usuário e é essencial para controle de permissões e funcionalidades específicas.

O modelo passou pelas três formas normais de normalização para garantir integridade e evitar redundância. O relacionamento entre as entidades é predominantemente do tipo 1:N (um para muitos), o que favorece o controle e a escalabilidade da estrutura de dados.



Modelo conceitual

O modelo conceitual foi elaborado utilizando o modelo Entidade-Relacionamento (ER), com o objetivo de representar graficamente e de forma abstrata os principais elementos de dados do sistema e seus relacionamentos. Este modelo permite uma compreensão geral da estrutura da base de dados antes da implementação física.



5.2. Criação Física do Modelo de Dados

A criação física do banco de dados foi realizada utilizando o ORM (Object-Relational Mapping) do Django, que permite mapear as classes Python diretamente em tabelas SQLite, sem a necessidade de escrever manualmente scripts SQL.

```
-- Tabela de usuários
CREATE TABLE Usuario (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL,
    senha TEXT NOT NULL,
    tipo_usuario_id INTEGER NOT NULL,
    FOREIGN KEY (tipo_usuario_id) REFERENCES TipoUsuario(id) ON DELETE CASCADE
);

CREATE TABLE TipoUsuario (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    tipo TEXT UNIQUE NOT NULL CHECK (tipo IN ('admin', 'usuario', 'feirante'))
);

-- Tabela de Feiras
CREATE TABLE Feira (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    endereco TEXT,
    latitude DECIMAL(9, 6),
    longitude DECIMAL(9, 6),
    horario_funcionamento TEXT
);

-- Feira Favorita
CREATE TABLE FeiraFavorita (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario_id INT NOT NULL,
    feira_id INT NOT NULL,
    data_favoritou TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id) ON DELETE CASCADE,
    FOREIGN KEY (feira_id) REFERENCES Feira(id) ON DELETE CASCADE,
    UNIQUE (usuario_id, feira_id)
);
```

```
-- Comentários
CREATE TABLE Comentario (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    usuario_id INT NOT NULL,
    feira_id INT NOT NULL,
    texto TEXT NOT NULL,
    nota INT CHECK (nota >= 1 AND nota <= 5),
    data_comentario TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id) ON DELETE CASCADE,
    FOREIGN KEY (feira_id) REFERENCES Feira(id) ON DELETE CASCADE
);

-- Barraca
CREATE TABLE Barraca (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    tipo TEXT NOT NULL,
    feirante_id INT NOT NULL,
    FOREIGN KEY (feirante_id) REFERENCES Usuario(id) ON DELETE CASCADE
);

-- Produto
CREATE TABLE Produto (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    categoria TEXT NOT NULL
);

-- Produto-Barraca
CREATE TABLE ProdutoBarraca (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    barraca_id INT NOT NULL,
    produto_id INT NOT NULL,
    observacao TEXT,
    disponivel BOOLEAN DEFAULT 1,
    FOREIGN KEY (barraca_id) REFERENCES Barraca(id) ON DELETE CASCADE,
    FOREIGN KEY (produto_id) REFERENCES Produto(id) ON DELETE CASCADE,
    UNIQUE (barraca_id, produto_id)
);
```

```
-- Presença Barraca na Feira
CREATE TABLE PresencaBarracaFeira (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    feira_id INT NOT NULL,
    barraca_id INT NOT NULL,
    data DATE NOT NULL,
    local_mapa TEXT,
    coordenadas TEXT,
    FOREIGN KEY (feira_id) REFERENCES Feira(id) ON DELETE CASCADE,
    FOREIGN KEY (barraca_id) REFERENCES Barraca(id) ON DELETE CASCADE,
    UNIQUE (feira_id, barraca_id, data)
);
```

5.3. Dicionário de Dados

O dicionário de dados do sistema tem como objetivo documentar todas as tabelas e atributos utilizados no banco de dados. Abaixo estão descritas as principais tabelas:

Tabela: Usuario

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador único do usuário
nome	TEXT	NOT NULL	Nome completo do usuário
email	TEXT	UNIQUE, NOT NULL	Endereço de e-mail (único)
senha	TEXT	NOT NULL	Senha criptografada
tipo_usuario_id	INTEGER	NOT NULL, FK para TipoUsuario(id)	Tipo de usuário (admin, usuário, feirante)

Tabela: TipoUsuario

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador do tipo de usuário
tipo	TEXT	UNIQUE, NOT NULL, CHECK (admin, usuario...)	Tipo do usuário permitido

Tabela: Feira

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador da feira
nome	TEXT	NOT NULL	Nome da feira
endereco	TEXT	—	Endereço físico da feira
latitude	DECIMAL(9,6)	—	Coordenada geográfica - latitude
longitude	DECIMAL(9,6)	—	Coordenada geográfica - longitude
horario_funcionamento	TEXT	—	Horário/dia de funcionamento da feira

Tabela: FeiraFavorita

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador do favorito
usuario_id	INTEGER	NOT NULL, FK para Usuario(id)	Usuário que favoritou
feira_id	INTEGER	NOT NULL, FK para Feira(id)	Feira favoritada
data_favoritou	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data e hora que a feira foi favoritada
(UNIQUE)	—	UNIQUE (usuario_id, feira_id)	Impede favoritar a mesma feira mais de uma vez

Tabela: Comentario

Atributo	Tipo	Restrições	Descrição
----------	------	------------	-----------

id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador do comentário
usuario_id	INTEGER	NOT NULL, FK para Usuario(id)	Usuário que comentou
feira_id	INTEGER	NOT NULL, FK para Feira(id)	Feira comentada
texto	TEXT	NOT NULL	Texto da avaliação
nota	INTEGER	CHECK (nota entre 1 e 5)	Nota atribuída à feira
data_comentario	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Data e hora da avaliação

Tabela: Barraca

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador da barraca
nome	TEXT	NOT NULL	Nome da barraca
tipo	TEXT	NOT NULL	Tipo da barraca (ex: alimentação)
feirante_id	INTEGER	NOT NULL, FK para Usuario(id)	Proprietário da barraca

Tabela: Produto

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador do produto
nome	TEXT	NOT NULL	Nome do produto
categoria	TEXT	NOT NULL	Categoria (hortifrúti, bebida etc)

Tabela: ProdutoBarraca

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador do vínculo
barraca_id	INTEGER	NOT NULL, FK para Barraca(id)	Barraca que vende o produto
produto_id	INTEGER	NOT NULL, FK para Produto(id)	Produto oferecido
observacao	TEXT	—	Observações (ex: promoção, orgânico)
disponivel	BOOLEAN	DEFAULT 1	Indica se está disponível para venda
(UNIQUE)	—	UNIQUE (barraca_id, produto_id)	Impede duplicidade do mesmo produto na barraca

Tabela: PresencaBarracaFeira

Atributo	Tipo	Restrições	Descrição
id	INTEGER	PRIMARY KEY, AUTOINCREMENT	Identificador da presença
feira_id	INTEGER	NOT NULL, FK para Feira(id)	Feira onde a barraca estará presente
barraca_id	INTEGER	NOT NULL, FK para Barraca(id)	Barraca presente na feira
data	DATE	NOT NULL	Data da presença
local_mapa	TEXT	—	Localização textual no mapa
coordenadas	TEXT	—	Coordenadas da barraca no mapa
(UNIQUE)	—	UNIQUE (feira_id, barraca_id, data)	Impede registros duplicados no mesmo dia

6. PROJETO DE SOFTWARE

Este capítulo detalha o design interno dos componentes do sistema com base na arquitetura em MVT (Model-View-Template) implementada com o framework Django, conforme os modelos conceitual, lógico e físico. O objetivo é fornecer um panorama da estrutura interna da aplicação, evidenciando a interação entre seus

módulos, a lógica implementada e os mecanismos de controle e validação dos dados.

6.1. Design dos Componentes

O sistema foi desenvolvido seguindo a arquitetura MVT (Model-View-Template) do framework **Django**, estruturado de forma modular com uso de múltiplos apps. Cada app é responsável por um conjunto específico de funcionalidades, o que promove a organização, a reutilização de código e facilita a manutenção. A seguir, são descritos os principais componentes (apps) do sistema e suas responsabilidades:

1. App usuarios

- **Responsabilidades:**
 - Gerenciamento de autenticação (login, logout, registro).
 - Definição de tipos de usuário: admin, feirante, usuario.
 - Armazenamento e recuperação de dados pessoais dos usuários.
 - Exibição do perfil do usuário e suas feiras favoritas.
- **Modelos:**
 - Usuario
 - TipoUsuario
- **Relacionamentos:**
 - Um Usuario possui um TipoUsuario (1:N).
 - Um Usuario pode ter várias feiras favoritas, comentários e barracas (quando feirante).
 - App dependente nos apps feiras, comentarios, barracas, favoritos.

2. App feiras

- **Responsabilidades:**
 - Cadastro e gerenciamento de feiras.
 - Armazenamento de localização, horários e dados complementares.
 - Exibição das feiras no mapa com filtros de busca.
- **Modelos:**
 - Feira
- **Relacionamentos:**
 - Uma Feira pode conter várias Barracas, Presencas, Comentarios e ser favorita por vários usuários.
 - App utilizado em comentarios, favoritos, presencas, barracas.

3. App barracas

- **Responsabilidades:**
 - Cadastro de barracas por feirantes.
 - Vínculo com produtos disponíveis.
 - Registro de presença da barraca em diferentes feiras.
- **Modelos:**
 - Barraca
- **Relacionamentos:**
 - Uma Barraca pertence a um Usuario com tipo feirante.
 - Uma Barraca pode ter vários produtos e comparecer a várias feiras (PresencaBarracaFeira).
 - Está associada a ProdutoBarraca e PresencaBarracaFeira.

4. App produtos

- **Responsabilidades:**
 - Gerenciamento dos produtos ofertados nas barracas.
 - Categorização e descrição dos produtos.
- **Modelos:**
 - Produto
 - ProdutoBarraca (associativa entre Produto e Barraca)
- **Relacionamentos:**
 - Uma Barraca pode oferecer vários Produtos.
 - Um Produto pode ser ofertado por várias Barracas.

5. App comentarios

- **Responsabilidades:**
 - Permitir que usuários comentem e avaliem as feiras.
 - Armazenar notas (de 1 a 5) e textos.
 - Apresentar comentários com data, nome do usuário e média de avaliações.
- **Modelos:**
 - Comentario
- **Relacionamentos:**
 - Um Usuario pode fazer vários comentários.
 - Uma Feira pode receber diversos comentários.

6. App favoritos

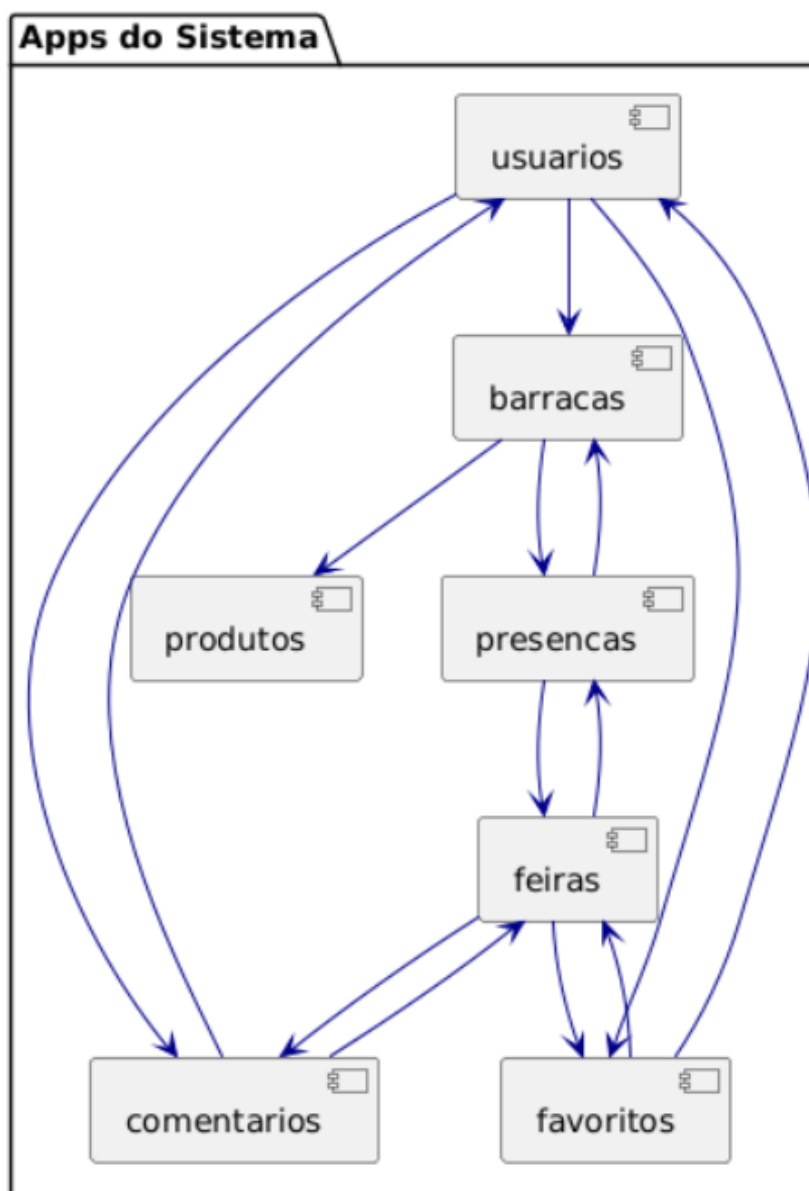
- **Responsabilidades:**
 - Permitir que usuários marquem feiras como favoritas.
 - Exibir lista de feiras favoritas no perfil do usuário.
- **Modelos:**
 - FeiraFavorita (entidade associativa)
- **Relacionamentos:**
 - Cada registro relaciona um Usuario com uma Feira.
 - Restrição de unicidade garante que um usuário só favorite uma feira uma vez.

7. App presencas

- **Responsabilidades:**
 - Registrar a presença de barracas em feiras específicas e datas.
 - Armazenar dados geográficos (local e coordenadas no mapa).
- **Modelos:**
 - PresencaBarracaFeira (entidade associativa)
- **Relacionamentos:**
 - Associa uma Barraca a uma Feira em determinada data.
 - Permite múltiplas presenças ao longo do tempo.

App	Depende de	Motivo da Dependência
usuarios	feiras, comentarios, favoritos, barracas	Usuário pode favoritar feiras, comentar, e ser feirante
feiras	comentarios, favoritos, presencas, barracas	Feiras possuem comentários, favoritos e presenças
barracas	usuarios, produtos, presencas	Barraca pertence a um feirante, vende produtos, está presente em feiras
produtos	barracas	Produtos estão associados a barracas
comentarios	usuarios, feiras	Comentários são feitos por usuários em feiras
favoritos	usuarios, feiras	Favoritos associam usuários e feiras
presencas	barracas, feiras	Presenças registram barracas em feiras

Mapa de dependência entre os apps do sistema



6.2. Diagrama de Classes

O diagrama de classes do sistema representa as estruturas fundamentais do projeto, com base nas classes definidas nos models do Django. Cada classe corresponde a uma entidade da base de dados, e os relacionamentos entre elas refletem as ligações por chaves estrangeiras e entidades associativas.

Classe Usuario

- **Atributos:**
 - id (int, PK)
 - nome (string)
 - email (string, único)
 - senha (string)
 - tipo_usuario (FK para TipoUsuario)
- **Métodos** (possíveis):
 - autenticar()
 - favoritar_feira(feira)
 - comentar(feira, texto, nota)
- **Relacionamentos:**
 - Agregação com TipoUsuario
 - Um para muitos com Comentario, FeiraFavorita, Barraca

Classe TipoUsuario

- **Atributos:**
 - id (int, PK)
 - tipo (char, 'admin', 'usuario', 'feirante')
- **Relacionamentos:**
 - 1:N com Usuario

Classe Feira

- **Atributos:**
 - id (int, PK)
 - nome (string)
 - endereco (string)
 - latitude (float)
 - longitude (float)
 - horario_funcionamento (string)
- **Métodos** (possíveis):
 - listar_barracas()

- obter_avaliacoes()
- **Relacionamentos:**
 - 1:N com Comentario, FeiraFavorita, PresencaBarracaFeira

Classe FeiraFavorita (Entidade Associativa)

- **Atributos:**
 - id (int, PK)
 - usuario (FK para Usuario)
 - feira (FK para Feira)
 - data_favoritou (timestamp)
- **Relacionamentos:**
 - Composição de Usuario e Feira
 - 1:1 com ambas as entidades no contexto da chave composta (usuario, feira)

Classe Comentario (Entidade Associativa com atributos)

- **Atributos:**
 - id (int, PK)
 - usuario (FK)
 - feira (FK)
 - texto (string)
 - nota (int)
 - data_comentario (timestamp)
- **Relacionamentos:**
 - Composição de Usuario e Feira
 - 1:N para ambas

Classe Barraca

- **Atributos:**
 - id (int, PK)
 - nome (string)
 - tipo (string)
 - feirante (FK para Usuario)
- **Relacionamentos:**
 - 1:N com ProdutoBarraca
 - 1:N com PresencaBarracaFeira
 - 1:1 com Usuario (feirante)

Classe Produto

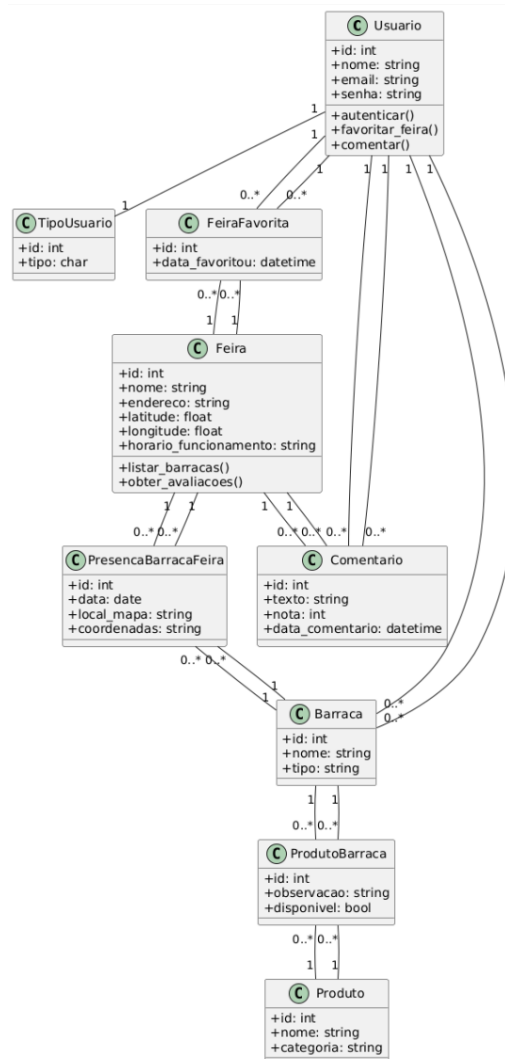
- **Atributos:**
 - id (int, PK)
 - nome (string)
 - categoria (string)
- **Relacionamentos:**
 - 1:N com ProdutoBarraca

Classe ProdutoBarraca (Entidade Associativa)

- **Atributos:**
 - id (int, PK)
 - barraca (FK)
 - produto (FK)
 - observacao (string, opcional)
 - disponivel (boolean)
- **Relacionamentos:**
 - Composição de Barraca e Produto
 - 1:N com ambos os lados

Classe PresencaBarracaFeira (Entidade Associativa)

- **Atributos:**
 - id (int, PK)
 - feira (FK)
 - barraca (FK)
 - data (date)
 - local_mapa (string, opcional)
 - coordenadas (string, opcional)
- **Relacionamentos:**
 - Composição entre Feira e Barraca
 - 1:N com ambos os lados



6.3. Diagrama de Sequência ou Fluxo de Interações

Esta seção descreve o comportamento dinâmico do sistema, representando o fluxo de comunicação entre os objetos e componentes do sistema durante a execução de funcionalidades específicas. Os diagramas de sequência evidenciam a ordem das chamadas e as mensagens trocadas entre os elementos.

Caso de Uso 1 – Cadastro de Usuário

Objetivo: Permitir que um novo usuário se cadastre no sistema.

Participantes:

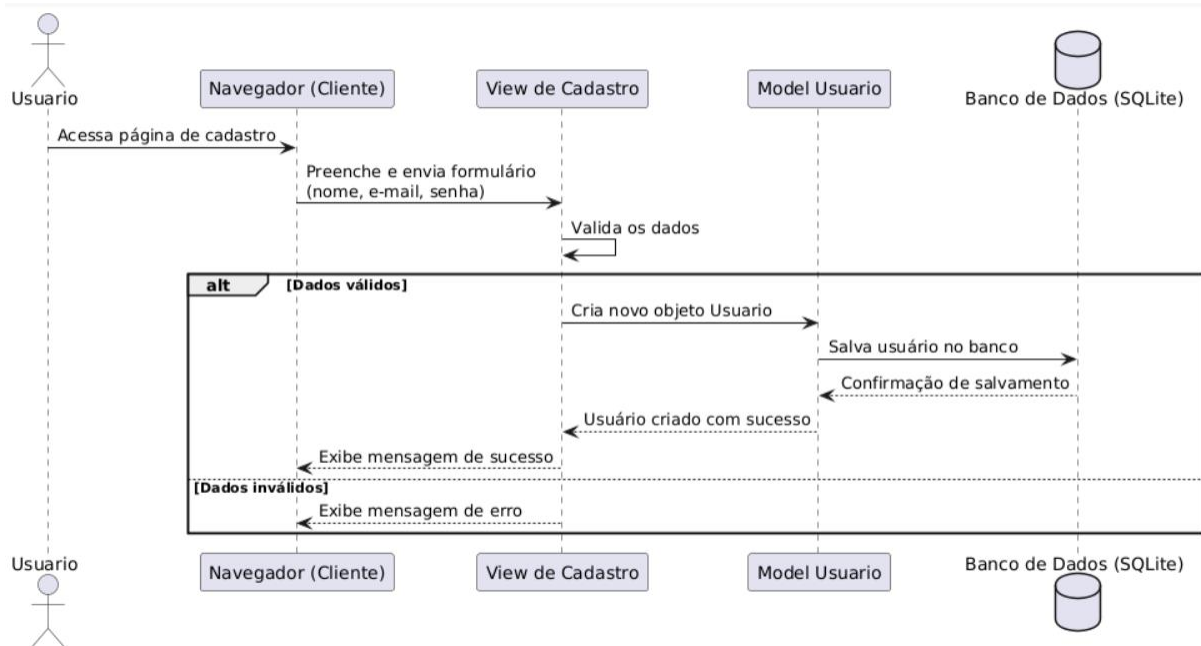
- Navegador (cliente)
- View de cadastro
- Model Usuario

- Banco de dados (SQLite)

Fluxo:

1. Usuário preenche o formulário de cadastro (nome, e-mail, senha).
2. A view recebe os dados e realiza validações.
3. Se os dados forem válidos, cria-se um novo objeto Usuario.
4. O objeto é salvo no banco de dados.
5. O sistema confirma o cadastro com uma mensagem de sucesso.

UML caso de uso – Cadastro de Usuário



Caso de Uso 2 – Login de Usuário

Objetivo: Autenticar o usuário e iniciar uma sessão.

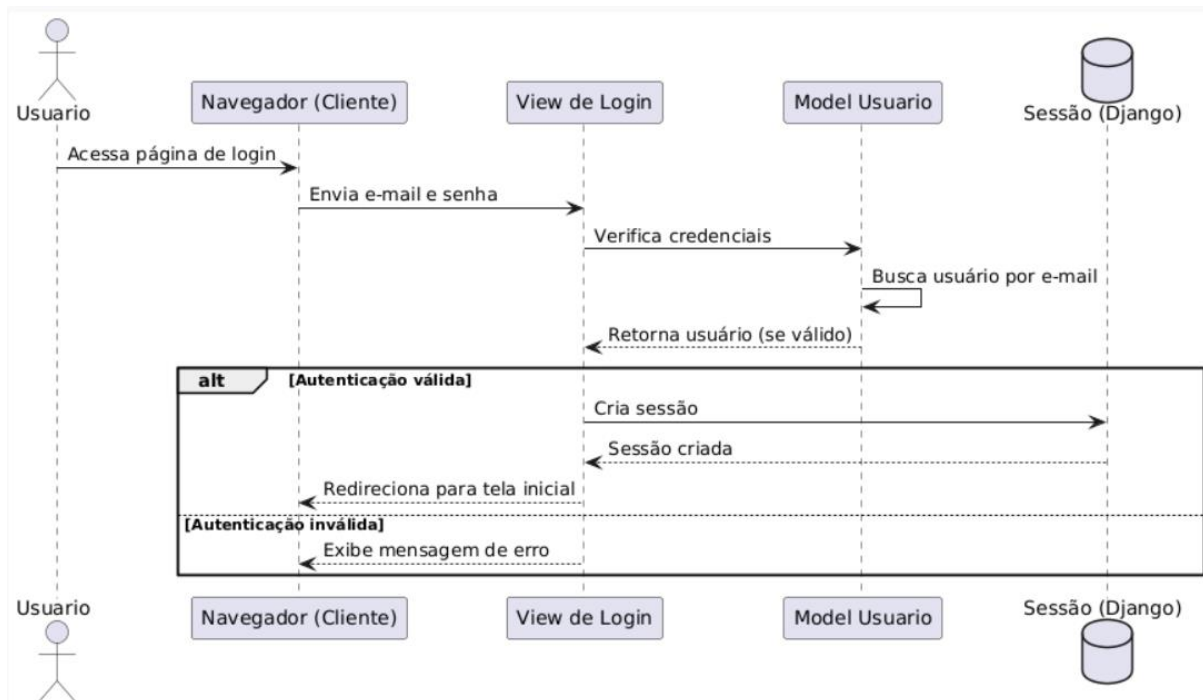
Participantes:

- Navegador (cliente)
- View de login
- Model Usuario
- Sessão do Django

Fluxo:

1. Usuário insere e-mail e senha.
2. A view verifica se há um usuário correspondente no banco.
3. Se a autenticação for bem-sucedida, cria-se a sessão.
4. Usuário é redirecionado à tela inicial/logado.

UML caso de uso - Login de Usuário



Caso de Uso 3 – Favoritar Feira

Objetivo: Permitir que o usuário favorite uma feira para acesso rápido posterior.

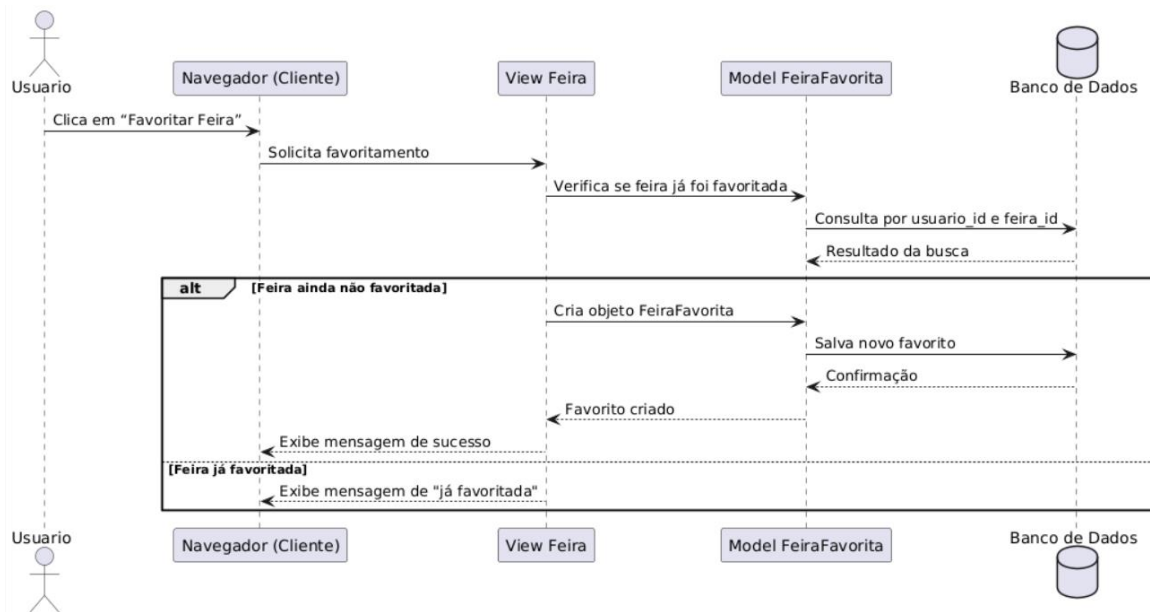
Participantes:

- Navegador (cliente)
- View de detalhes da feira
- Model FeiraFavorita
- Banco de dados

Fluxo:

1. Usuário clica em “Favoritar feira”.
2. A view verifica se a feira já está favoritada.
3. Se não estiver, cria-se um objeto FeiraFavorita com o usuario_id e feira_id.
4. O objeto é salvo no banco.
5. Uma mensagem de sucesso é exibida.

UML caso de uso - Favoritar Feira



Caso de Uso 4 – Adicionar Comentário/Avaliação

Objetivo: Permitir que o usuário registre sua opinião e nota sobre uma feira.

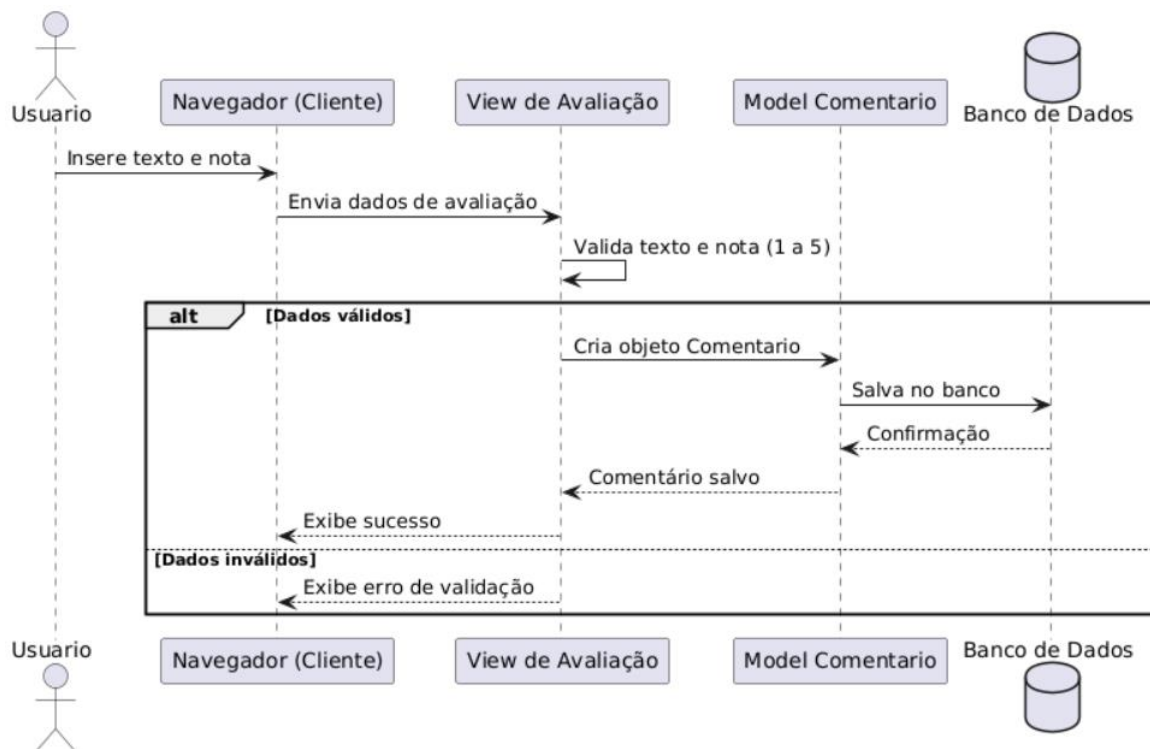
Participantes:

- Navegador (cliente)
- View de avaliação
- Model Comentario
- Banco de dados

Fluxo:

1. Usuário insere texto e nota da avaliação.
2. A view valida os dados (ex: nota entre 1 e 5).
3. Cria-se um objeto Comentario com as informações e referências do usuário e da feira.
4. O comentário é salvo no banco.
5. A avaliação aparece listada na página da feira.

UML caso de uso – Adicionar Comentário/Avaliação



6.4. Regras de Negócio Detalhadas

As regras de negócio do sistema foram definidas para garantir a integridade dos dados, o comportamento esperado das funcionalidades e a segurança do sistema. A seguir, estão detalhadas as principais regras implementadas nos modelos (models.py) e nas views (views.py) do Django.

Usuário

- O **e-mail deve ser único** (validação no modelo e no formulário).
- O **tipo de usuário** deve ser obrigatoriamente admin, usuario ou feirante (validação no modelo e via chave estrangeira para TipoUsuario).
- Apenas **usuários autenticados** podem:
 - Favoritar feiras
 - Comentar feiras
 - Cadastrar barracas (apenas usuários do tipo feirante) – validação na view.
- A **senha deve ser armazenada de forma segura** com hash (implementação via sistema de autenticação padrão do Django).

Feira

- Deve ter **nome obrigatório**.
- O campo **horário de funcionamento** também é obrigatório.
- Os campos **latitude e longitude** são opcionais, mas utilizados na exibição no mapa.
- Apenas **usuários administradores** podem cadastrar, editar ou excluir feiras (controle de permissão na view).

FeiraFavorita (relação entre Usuário e Feira)

- Um usuário **pode favoritar uma feira apenas uma vez** (validação via `unique_together` no modelo).
- Só usuários autenticados podem favoritar (validação na view).
- Caso o favorito já exista, a operação é ignorada ou substituída.

Comentário

- Somente **usuários autenticados** podem comentar (validação na view).
- Um **usuário só pode comentar uma vez por feira por dia** (validação na view, com consulta prévia).
- A **nota** deve ser um valor **entre 1 e 5** (validação no modelo).
- O **texto** do comentário é obrigatório e limitado a 500 caracteres.

Barraca

- Só pode ser **criada por usuários com tipo feirante** (verificação na view).
- Cada barraca tem **nome, tipo e um feirante obrigatório**.
- O feirante **não pode ter duas barracas com o mesmo nome** em uma mesma feira (validação na view ou constraint no banco).

Produto

- Deve conter **nome e categoria obrigatórios**.
- Os nomes podem ser repetidos entre categorias distintas.
- Pode ser reutilizado por várias barracas (relacionamento N:N via tabela associativa ProdutoBarraca).

ProdutoBarraca (associação)

- Um mesmo produto **não pode ser associado duas vezes à mesma barraca** (validação no modelo com `unique_together`).
- O campo `disponivel` indica se o produto está em estoque (usado para lógica de exibição).
- O campo `observacao` é opcional, e serve para descrição adicional ou promoções.

PresencaBarracaFeira (associação)

- Registra a **presença de uma barraca em uma feira em uma data específica**.
- A combinação de feira, barraca e data deve ser **única** (validação no modelo).
- O campo `local_mapa` pode armazenar a descrição textual da posição da barraca no mapa.
- `coordenadas` é um campo opcional que pode ser usado em integrações com mapas.

TipoUsuario

- Define os três tipos possíveis: `admin`, `usuario`, `feirante`.
- Apenas administradores podem cadastrar ou alterar os tipos (geralmente fixos no banco).
- Um tipo não pode ser excluído se ainda estiver sendo utilizado por algum usuário (restrição de integridade referencial).

Autorização e Consistência

- Toda **ação sensível ou restrita** é protegida com `@login_required` e, quando necessário, validação do tipo de usuário na view.
- As **ações de criação, edição e exclusão** (CRUD) são restritas conforme os perfis:
 - `admin`: acesso completo
 - `usuario`: ações de consumo (avaliar, favoritar, consultar)
 - `feirante`: pode gerenciar barracas e produtos próprios
- **Mensagens de erro amigáveis** são exibidas sempre que uma regra é violada (tratamento com mensagens e validações em formulários).

6.5. Estratégias de Tratamento de Erros

O tratamento de erros é uma parte essencial do projeto, pois visa garantir que o sistema continue funcionando de maneira segura e controlada, mesmo diante de falhas ou situações inesperadas. As estratégias adotadas no sistema foram pensadas para oferecer **robustez, clareza e uma boa experiência ao usuário final**.

1. Validação de Entradas

- Todos os dados inseridos pelo usuário, como formulários de cadastro, login, comentários, entre outros, são validados tanto no **lado do cliente** (front-end) quanto no **lado do servidor** (back-end).
- Campos obrigatórios são verificados antes do envio e exibem mensagens amigáveis caso estejam em branco ou inválidos.
- No Django, são utilizados **Formulários** e **ModelForm** para aplicar regras de validação com mensagens personalizadas.

2. Tratamento de Erros nas Views

- Nas views Django, erros como tentativas de acesso a recursos não autorizados, falhas de autenticação ou dados inválidos são capturados com **estruturas try/except**.
- O sistema exibe **mensagens de erro informativas**, sem revelar detalhes internos do servidor.
- Exemplo: ao tentar favoritar uma feira já favorita, uma mensagem do tipo “Você já adicionou esta feira aos seus favoritos” será exibida.

3. Redirecionamentos em Caso de Falha

- Caso uma ação falhe (ex: login com senha incorreta), o sistema redireciona o usuário para a mesma página com a informação do erro.
- Utiliza-se o sistema de mensagens do Django (`django.contrib.messages`) para exibir feedback como:
 - “Senha incorreta.”
 - “Campos obrigatórios não preenchidos.”
 - “Acesso negado. Permissão insuficiente.”

4. Respostas para Requisições Inválidas

- Para páginas ou rotas não encontradas, o sistema exibe uma **página personalizada de erro 404**, informando de forma amigável que o recurso não existe.
- Em caso de erros de servidor, como exceções não tratadas, uma **página 500 customizada** será apresentada, preservando a confidencialidade do erro técnico.

5. Ações Proibidas ou Restritas

- O sistema bloqueia tentativas de manipulação indevida ou acesso não autorizado.
- Verificações adicionais são feitas em todas as views protegidas por autenticação (@login_required) e verificação do tipo de usuário (admin, feirante, usuario).
- A tentativa de acessar ações restritas sem as devidas permissões gera uma resposta controlada, como:
 - “Você não tem permissão para realizar esta ação.”

6. Logs e Debugging (Ambiente de Desenvolvimento)

- Em ambiente de desenvolvimento, os erros são exibidos com detalhes para facilitar o processo de correção.
- Em produção, erros críticos são **registrados em log** e não exibidos ao usuário final.
- O sistema poderá ser configurado para **enviar alertas de erro** ou salvar logs detalhados com timestamp e traceback.

7. Padrões e Consistência

- Todas as mensagens seguem uma padronização de **linguagem clara, objetiva e acessível**.
- A exibição é feita preferencialmente por **alerts Bootstrap** ou áreas visuais dedicadas na interface.
- Mensagens de sucesso, erro, aviso e informação são estilizadas de forma distinta para facilitar a compreensão.

7. IMPLEMENTAÇÃO

Este capítulo documenta como as funcionalidades do sistema foram implementadas utilizando o framework Django, destacando a estrutura modularizada, a aplicação de princípios de Programação Orientada a Objetos (POO) e a integração com autenticação baseada em tokens JWT.

7.1. Estrutura Geral do Projeto

O projeto está organizado da seguinte forma:

feiras_livres/

└─ core/ # Aplicação principal com models de feira, barraca e presença

```

|   ├── models.py    # Modelos Feira, Barraca, PresencaBarracaFeira
|   ├── views.py
|   ├── urls.py
|   └── templates/
└── usuarios/        # Aplicação de autenticação e usuários
    ├── models.py    # Modelo customizado Usuario
    ├── serializers.py # Serializador com validação de senha e criação com hash
    ├── views.py
    ├── urls.py
    └── forms.py
└── feiras_livres/   # Configurações globais do Django
    ├── settings.py
    ├── urls.py
    ├── asgi.py / wsgi.py
    ├── db.sqlite3
    └── manage.py

```

7.2. Modelos

A seguir estão os modelos principais utilizados no sistema:

7.2.1. Modelo Feira

Representa uma feira livre cadastrada no sistema. Inclui dados como nome, localização, dias de funcionamento e horários.

Campos principais:

- nome: Nome da feira
- endereco, bairro, cidade
- latitude, longitude
- dias_funcionamento: Ex: "seg,qua,sex"

- `horario_abertura` e `horario_fechamento`

7.2.2. Modelo Barraca

Relaciona uma barraca a uma feira específica e ao usuário feirante proprietário.

Relacionamentos:

- `ForeignKey(Feira)`
- `ForeignKey(settings.AUTH_USER_MODEL)`

7.2.3. Modelo PresencaBarracaFeira

Registra a presença de barracas em feiras em dias específicos.

Campos principais:

- `feira`, `barraca` (FKs)
- `data da presença`
- `local_mapa` e coordenadas para posicionamento

7.2.4. Modelo Usuario

Substitui o modelo padrão de usuário do Django com campos adicionais como endereço completo e telefone.

Destaques:

- Campo email como `USERNAME_FIELD`
- Autenticação com PBKDF2 via `set_password()`
- Manager customizado (`UsuarioManager`) para criação de usuários e superusuários

7.3. Views

As views são separadas em dois contextos: API REST (usando `APIView`, `generics`) e views baseadas em função para o frontend.

7.3.1. Autenticação com JWT

A autenticação de usuários via API utiliza `SimpleJWT`. As rotas de login e refresh são:

- `api/token/` para login e obtenção de access e refresh tokens
- `api/token/refresh/` para renovação do token

Além disso, o endpoint `/me/` retorna os dados do usuário autenticado via token.

7.3.2. Views HTML (Frontend)

- `cadastro_view`: Formulário com preenchimento automático de endereço via CEP
- `login_view`: Autentica com `authenticate()` e redireciona para `/dashboard/`
- `dashboard_view`: Requer login com `@login_required`
- `feira_detalhes_view`: Mostra detalhes da feira em página com mapa, usando Leaflet.js

7.4. Templates

Os templates utilizam herança com `base.html`, campos dinâmicos com `{{ user.first_name }}` e interações com o backend via JavaScript (`fetch`, `geolocalização`, `autocomplete`).

7.4.1. Template `dashboard.html`

- Input com `autocomplete` e API de endereço
- Mapa dinâmico com Leaflet.js
- Lista de feiras buscadas com redirecionamento para detalhes da feira
- Proteção via JWT: redireciona para login se token estiver ausente

7.4.2. Template `feira_detalhes.html`

- Exibe os dados da feira (nome, bairro, horários, localização)
- Integração com mapa via Leaflet, marcando a feira

7.4.3. Template `base.html`

- Estrutura base com header, navegação e rodapé
- Rodapé com redes sociais e contatos

7.4.4. Template `login.html`

- Formulário de login com campos de email e senha
- Link para redirecionar ao cadastro

7.4.5. Template register.html

- Formulário de cadastro completo com preenchimento automático por CEP (ViaCEP)
- Campos para nome completo, endereço, telefone e senha

7.5. Segurança e Criptografia

- Todas as senhas são armazenadas com hash usando PBKDF2, implementado automaticamente por `set_password()`
- Login via API utiliza JWT (tokens de acesso e refresh)
- View protegida com `@login_required` e `IsAuthenticated`