FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Hadston Alexandre de Menezes Nunes

MEUS FIADOS

Americana, SP

2023

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Hadston Alexandre de Menezes Nunes

MEUS FIADOS

Projeto de Conclusão da disciplina Laboratório de Engenharia de Software do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Dr. Ivan Menerval da Silva

Área de concentração: Engenharia de Software

Americana, SP

2023

RESUMO

Este projeto tem como objetivo desenvolver um aplicativo de gerenciamento de fiados para alguns setores como serviços de alimentação, varejistas, prestadores de serviços e a comunidade em geral que disponha de um catálogo. O aplicativo terá dois ambientes, um para o vendedor dos produtos ou serviços e outra para o cliente marcar o produto desejado ou comprado. No ambiente do vendedor, ele poderá cadastrar clientes, gerenciar contas, verificar os saldos etc. No ambiente do cliente, ele poderá escanear um QR Code e registrar o produto que está adquirindo.

A metodologia usada para o desenvolvimento do projeto será baseada em Scrum, com reuniões periódicas, sprints e entregas incrementais. Os resultados serão avaliados com base na satisfação do cliente, eficiência do processo e facilidade de uso. As conclusões do trabalho serão apresentadas com base nos resultados obtidos e nos aprendizados adquiridos durante o desenvolvimento do projeto.

Palavras Chave: Gerenciamento de fiados; QR Code; Android.

1. LISTAS

1.1. Lista de Tabelas

1.2. Lista de Figuras

2. INTRODUÇÃO

O comércio com crediário informal, mesmo que haja riscos e incertezas, é encarado como uma forma de aumentar o faturamento de uma empresa (Silva, 2018), sempre foi muito comum as pessoas negociarem produtos de seus amigos de maneira informal. Sempre existe aquela pessoa que está vendendo alguma coisa para ser pago no pagamento. Se olha para o gerenciamento financeiro, muitas vezes se vê muitos papéis, planilhas ou outras anotações, gerando um trabalho maçante.

E o gerenciamento financeiro em venda de fiado é um problema comum em pequenos negócios, como os exemplos mencionados, onde a falta de uma ferramenta adequada pode levar a inúmeros problemas. Com o registro manual em papéis, há uma maior propensão a erros, como perda de informações importantes e anotações ilegíveis, além de dificuldades em gerenciar e controlar o crédito concedido, as dívidas em aberto e o fluxo de caixa. Além disso, a falta de organização e controle pode gerar desentendimentos entre vendedor e comprador, comprometendo a relação de amizade ou de negócio. Portanto, para automatizar esse processo poderia trazer inúmeros benefícios tanto para os vendedores quanto para os compradores, simplificando e otimizando a gestão de vendas e a relação comercial.

Com a evolução da tecnologia e a popularização dos smartphones, tornou-se possível gerenciar as transações financeiras de uma forma muito mais simples e eficiente. Uma das possibilidades é o uso de aplicativos de gerenciamento de finanças pessoais, mas e se pudéssemos aplicar essa mesma tecnologia para gerenciar as vendas e compras informais que as pessoas fazem com seus amigos? Neste contexto que surge a ideia deste projeto: um aplicativo para gerenciamento de vendas em fiado de maneira rápida, simples e intuitiva, que pode ser usado tanto pelo vendedor quanto pelo cliente.

Este trabalho tem como objetivo desenvolver um aplicativo para gestão dos fiados. Para este objetivo, será usado a metodologia Scrum para desenvolvimento do aplicativo e o ambiente de desenvolvimento, o Android Studio utilizando a linguagem Kotlin. O restante do trabalho será organizado em três capítulos conforme descrição a seguir: Capítulo 3 apresenta a documentação do sistema desenvolvido, o Capítulo 4 demonstra os elementos do projeto em diagramas, o Capítulo 5 descreve o desenvolvimento do projeto utilizando a metodologia de entregáveis, e por fim, no Capítulo 6, as considerações finais juntamente com as possibilidades de trabalhos futuros.

3. PROJETO DO SISTEMA

A etapa de projeto do sistema é um processo fundamental na engenharia de software. Para desenvolver um software que atenda às necessidades do usuário, é necessário realizar uma análise cuidadosa de outros softwares similares, para identificar seus pontos fortes e fracos e compreender as expectativas dos usuários. Esta etapa é fundamental porque tem como objetivo definir os elementos e componentes do software com base nas informações coletadas na etapa de levantamento de requisitos. Dentre os aspectos considerados na etapa de projeto estão a arquitetura de software, o design de suas interfaces, a escolha das tecnologias e plataformas, e a elaboração de um plano de testes e avaliação para verificar a qualidade e usabilidade do software. Assim, é essencial garantir que o produto atenda às expectativas do usuário e seja desenvolvido de forma eficiente e eficaz.

1.1. Softwares Similares

Atualmente existem alguns aplicativos voltados para gerenciamento de fiados com diferentes funcionalidades, foram selecionadas três aplicações relacionadas com o tema na Play Store, são eles:

- **Crediado**: o aplicativo ajuda empreendedores a gerenciar dívidas de seus clientes com um cadastro simples e integração com o PIX. Com um site exclusivo para o negócio, clientes podem visualizar suas contas. Tem 3,9 de nota na Google Play. (GOOGLE PLAY, 2023);
- **Fiado**: este app gerencia dívidas entre amigos, família e colegas, com interface simples e cria "mini contas" para cada pessoa que deve dinheiro, no smartphone. Visualize dívidas abertas facilmente, evitando conflitos. 4,4 na Google Play (GOOGLE PLAY, 2023).
- **Somei**: promete um fácil gerenciamento do seu negócio. O aplicativo oferece uma interface intuitiva e gráficos de estatísticas excelentes, além de permitir o cadastro de produtos com foto e diversas funcionalidades para controle de vendas. Gratuito, oferece a opção de adicionar o custo do produto antes de finalizar a venda, possibilitando uma melhor visualização dos resultados. No entanto, existem ressalvas feitas pelos usuários, como a falta de opção de

estoque e um sistema de busca problemático na adição de endereços e clientes. Apesar disso, tem uma nota alta de 4,7 na Google Play (GOOGLE PLAY, 2023);

Levando estes aspectos em consideração, foi elaborado a tabela abaixo (Tabela 1), que relaciona alguns diferenciais entre o Meus Fiados e os respectivos aplicativos:

Tabela 1 - Comparativo de funcionalidades da aplicação Meus Fiados em relação aos concorrentes.

Funcionalidades	Crediado	Fiado	Somei	Meus Fiados
Ambiente para o Vendedor	**X**	**-**	**-**	**X**
Ambiente para o Cliente	**X**	**-**	**-**	**X**
Login pelo Google	**X**	**-**	**-**	**X**
Cadastro de Clientes	**X**	**X**	**X**	**X**
Cadastrar Cliente pela Lista de Contatos	**X**	**-**	**-**	**X**
Cadastro de Produto	**X**	**-**	**X**	**X**
Controle dos Fiados	**X**	**X**	**X**	**X**
Comprar Fiado pelo QRCode	**-**	**-**	**-**	**X**
Envio de Cobrança Individual	**X**	**-**	**-**	**X**
Envio de Cobrança em Lote	**-**	**-**	**-**	**X**
Integração com o PIX	**X**	**-**	**-**	**X**
Registro dos Pagamentos	**-**	**-**	**-**	**X**
Vitrine Inicial com Indicadores de Resultado	**-**	**-**	**X**	**X**
Relatório de Produtos com QRCode	**-**	**-**	**-**	**X**
Relatórios de Contas (Vencido, A Vencer)	**-**	**-**	**-**	**X**
Relatório de Resultados por Período	**-**	**-**	**X**	**X**

Fonte: Elaborado pelo autor (2023).

1.2. Levantamento de Requisitos

A engenharia de requisitos (RE – Requirements Engineering) é o processo de descobrir, analisar, documentar e verificar requisitos de um sistema. Um requisito pode ser definido como uma descrição dos serviços fornecidos pelo sistema e suas restrições operacionais (SOMMERVILLE, 2007). Tradicionalmente, os requisitos são divididos em dois tipos: requisitos funcionais e requisitos não funcionais.

1.2.1. Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, definem a funcionalidade desejada do software (SOMMERVILLE, 2007). A Tabela 2 apresenta os requisitos funcionais deste projeto.

Tabela 2 – Requisitos funcionais do projeto.

Identificação	Requisito Funcional	Prioridade
RF001	Selecionar Ambiente (Vendedor / Cliente)	Essencial
RF002	Cadastrar Cliente	Essencial
RF003	Cadastrar Produto	Essencial
RF004	Gerenciar Fiados	Essencial
RF005	Comprar Produto (Manual / QR Code)	Essencial
RF006	Gerenciar Cobrança	Essencial
RF007	Visualizar Produtos (QR Code)	Essencial
RF008	Visualizar Contas (Vencidas, A Vencer)	Importante
RF009	Visualizar Resultados (Por Período)	Importante
RF010	Cadastrar Cliente pela Lista de Contato	Desejável
RF011	Enviar Cobrança (Individual / Lote)	Desejável
RF012	Efetuar Pagamento (Simples / Integração PIX)	Desejável

Fonte: Elaborado pelo autor (2023).

1.2.2. Requisitos Não Funcionais

Os requisitos não funcionais são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema (SOMMERVILLE, 2007). A Tabela 3 apresenta os requisitos não funcionais deste projeto.

Tabela 3 – Requisitos não funcionais do projeto.

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Autenticação	Segurança	Essencial
RNF002	Sistema de cache em SQLite	Desempenho	Essencial
RNF003	Sistema de recuperação de falhas	Confiabilidade	Essencial
RNF004	Práticas de usabilidade na UI	Usabilidade	Essencial
RNF005	Controlada por Convite	Distribuição	Essencial
RNF006	Padrão de projeto Clean Architecture	Padrões	Essencial
RNF007	1 GB de RAM / Android 8.0 ou superior	Hardware e Software	Essencial
RNF008	Necessário ter Internet para efetuar a Compra	Disponibilidade	Essencial

Fonte: Elaborado pelo autor (2018).

1.3. Recursos e Ferramentas

Esta seção contempla as ferramentas de programação e os conceitos necessários para o desenvolvimento do sistema:

- Plataforma: Android Studio {descrição da plataforma};
- Linguagem de Programação: Kotlin no Front {Descrição da linguagem e como será utilizada}, C# no Backend {Descrição da linguagem e como ela será utilizada};
- Markdown: Uma linguagem simples de marcação criada para escrever textos com o recurso rich-text e para ser humanamente legível sem precisar, necessariamente de interpretador como o HTML.
- Mermaid: É uma ferramenta de diagramação e gráficos baseada em JavaScript que renderiza comandos ou definições em texto inspiradas em Markdown para criar e modificar diagramas dinamicamente.

4. MODELAGEM

Na fase de modelagem é feita a documentação do aplicativo, se tratam de diagramas que facilitam na compreensão do projeto de forma padronizada.

A documentação deste trabalho utilizará a linguagem de modelagem “Unified Modeling Language 2” (UML) para modelar os casos de uso e os diagramas de classe, utilizando o Mermaid para criar e editar os diagramas.

4.1. Casos de Uso

Os diagramas de caso de uso descrevem um cenário de funcionalidades do ponto de vista do usuário, catalogando os requisitos funcionais do sistema. Dentro do diagrama são retratados os atores (representado pelos bonecos), as funcionalidades (representados pelos balões com a ação escrita por dentro) e as relações (representadas pelas linhas).

Os atores que interagem com o sistema são: o Usuário, API Google, Firebase, API Meus Fiados. O sistema é um caso de uso explícito e se trata do sistema em si em que os casos de uso acontecem.

4.1.1. Vendedor

É um dos principais atores do sistema, representa o usuário final que utiliza a aplicação e interagem com ela para realizar suas tarefas de venda;

A Figura 1 apresenta o caso de uso para os recursos disponíveis para o usuário vendedor no seu ambiente.

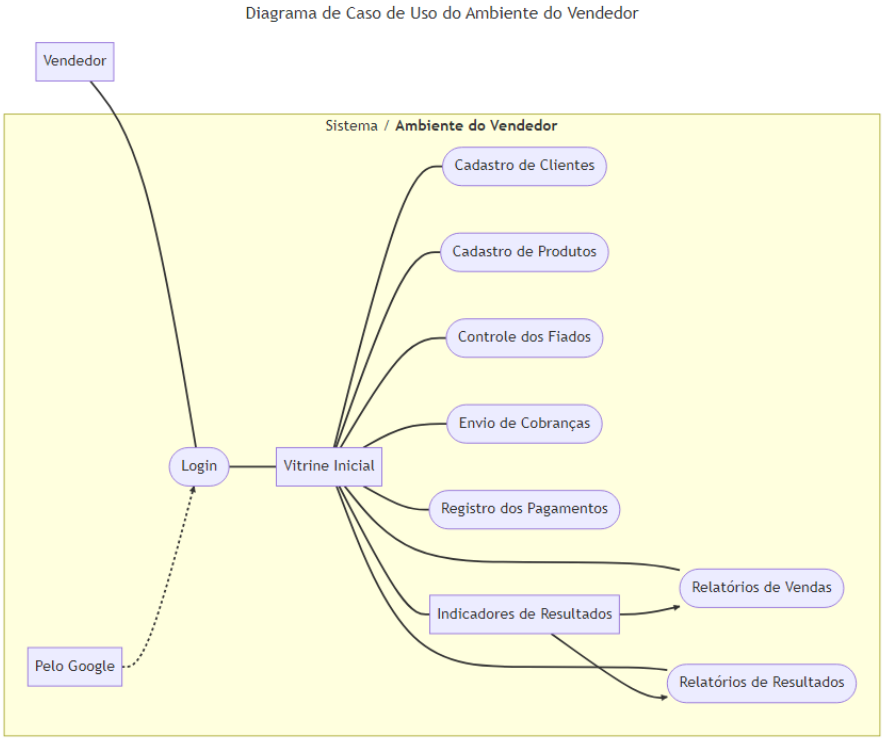

Figura 1 - Diagrama de Caso de Uso do Ambiente do Vendedor

4.1.2. Cliente

É um dos principais atores do sistema, representa o usuário final que utiliza a aplicação e interagem com ela para realizar suas tarefas de compra;

A Figura 2 demonstra o caso de uso para os recursos disponíveis para o usuário cliente no seu ambiente.

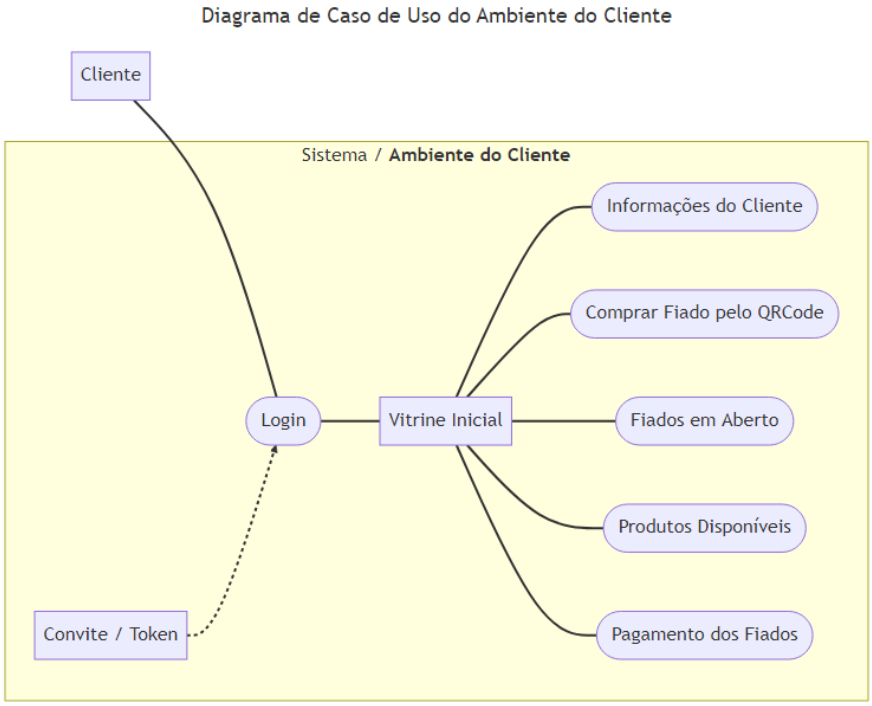

Figura 2 - Diagrama de Caso de Uso do Ambiente do Cliente

4.1.3. APIs

Este diagrama de caso de uso do Cadastro de Clientes exemplifica o uso de outro ator ainda não mencionado, o das APIs, neste contexto a API Meus Fiados é utilizado.

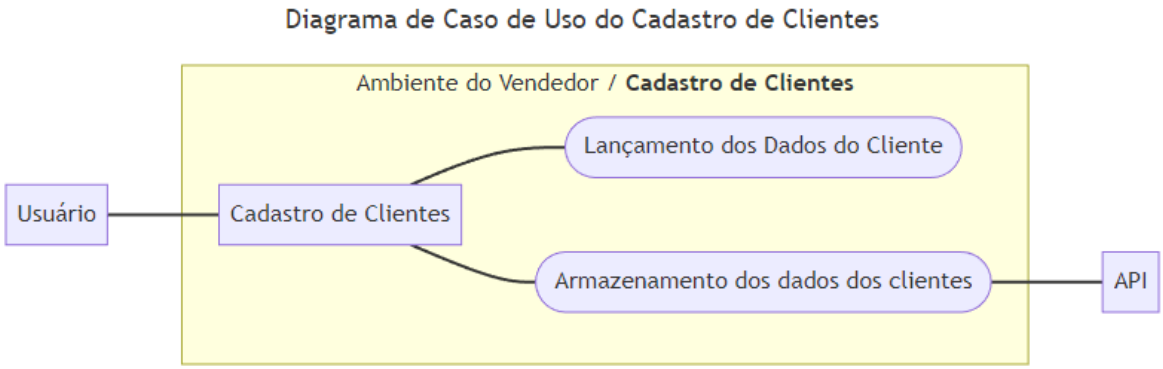

Figura 3 - Diagrama conectando um ator API

- **API do Google**: é um ator externo que fornece a interface de programação de aplicativos para o sistema, permitindo a integração com os serviços da plataforma Google, como a autenticação;
- **Firebase**: é um ator externo que fornece uma plataforma de desenvolvimento de aplicativos móveis e web baseada em nuvem, oferecendo serviços de back-end como armazenamento em nuvem e notificações push para o sistema;
- **API Meus Fiados**: é um ator externo que fornece uma interface de programação de aplicativos para o sistema, permitindo a integração com

serviços de fiados, como a sincronização da lista de produtos com QRCode e Pedidos realizados.

- **API do Pix**: é um ator externo que permite a integração de sistemas e serviços com o sistema de pagamentos instantâneos Pix, desenvolvido pelo Banco Central do Brasil. Por meio desta API é possível realizar operações de geração de cobranças, recebimento de pagamentos, consulta de transações e a integração com a plataforma de liquidação do Pix, permitindo automatizar e simplificar os processos de pagamento.

4.2. Classes

Diagrama de Classes é uma ferramenta da UML (Unified Modeling Language) utilizada para modelar a estrutura de um sistema orientado a objetos. Ele descreve as classes, atributos e métodos de um sistema, bem como as relações entre as classes.

4.2.1. Autenticação

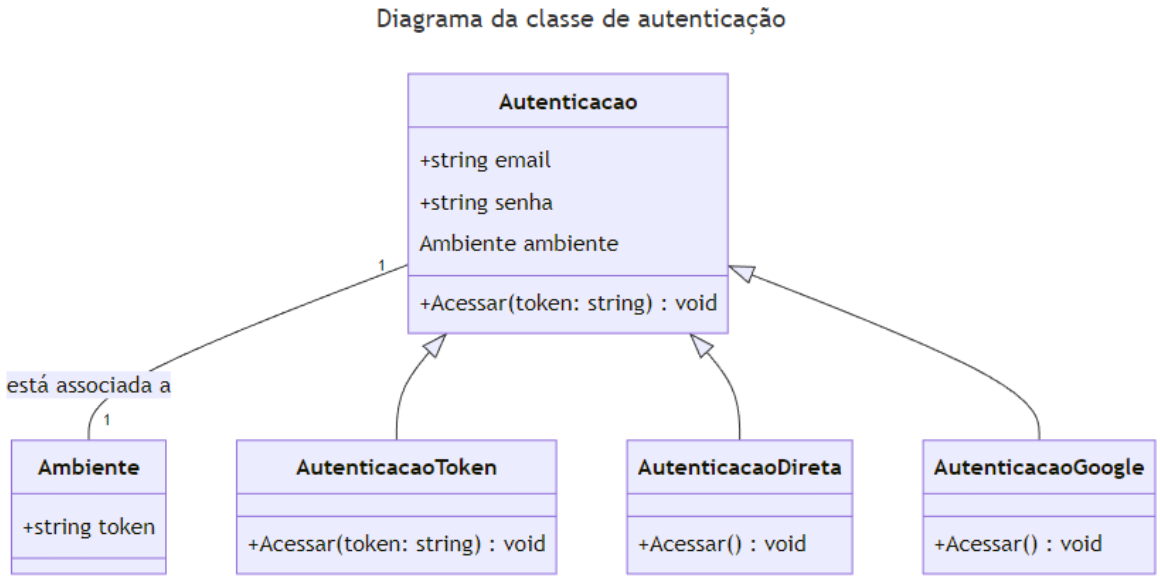

Figura 4 - Diagrama de classe Autenticação

A classe autenticação representa o processo de autenticação de usuário no sistema. Esta classe possui as propriedades “email” e “senha” para armazenar as credenciais do usuário e a propriedade “ambiente” que referencia a classe “Ambiente” e contém informações sobre o ambiente em que a autenticação está ocorrendo, incluindo o token de autenticação. A classe “Autenticacao” possui um método “Acessar” que recebe um token como parâmetro e realiza a autenticação. Além disso, existem outras classes que herdam da classe “Autenticacao”: “AutenticacaoToken”, “AutenticacaoDireta” e “AutenticacaoGoogle”, cada uma implementando seu próprio

método “Acessar” de acordo com a forma específica de autenticação. Essa estrutura de classes permite diferentes métodos de autenticação serem utilizados no sistema, proporcionando flexibilidade e modularidade na implementação do processo de autenticação.

4.2.2. Cadastro

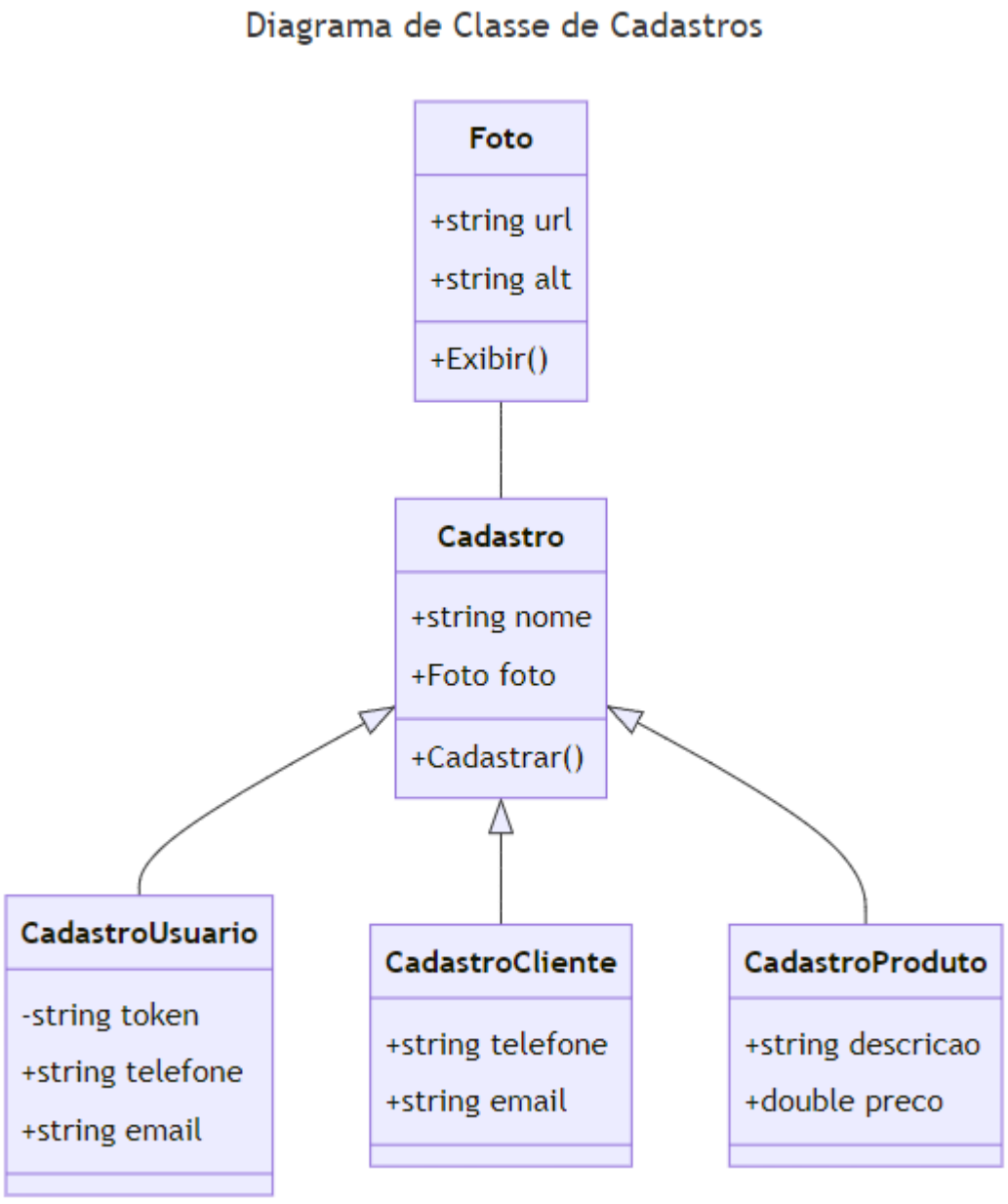

Figura 5 - Diagrama de classe Cadastro

Este diagrama representa o processo de cadastro de um elemento no sistema. Essa classe possui a propriedade "nome" para armazenar o nome do elemento a ser cadastrado e a propriedade "foto" que referencia a classe "Foto" e contém informações sobre a imagem relacionada ao cadastro, como a URL e o texto alternativo. A classe

"Cadastro" possui o método "Cadastrar" que realiza a ação de cadastrar o elemento no sistema. Além disso, existem outras classes que herdam da classe "Cadastro": "CadastroUsuario", "CadastroCliente" e "CadastroProduto".

A classe "CadastroUsuario" possui propriedades como "token", "telefone" e "email", representando informações específicas de um cadastro de usuário no sistema. A classe "CadastroCliente" também possui propriedades "telefone" e "email", indicando informações específicas de um cadastro de cliente. Por fim, a classe "CadastroProduto" possui propriedades como "descricao" e "preco", relacionadas a um cadastro de produto no sistema.

Essa estrutura de classes permite diferentes tipos de cadastro serem realizados no sistema, com propriedades específicas de cada tipo. Essa abordagem modular e hierárquica facilita a extensibilidade do sistema, permitindo a adição de novos tipos de cadastro com características distintas de forma simples e organizada.

4.2.3. Negociação

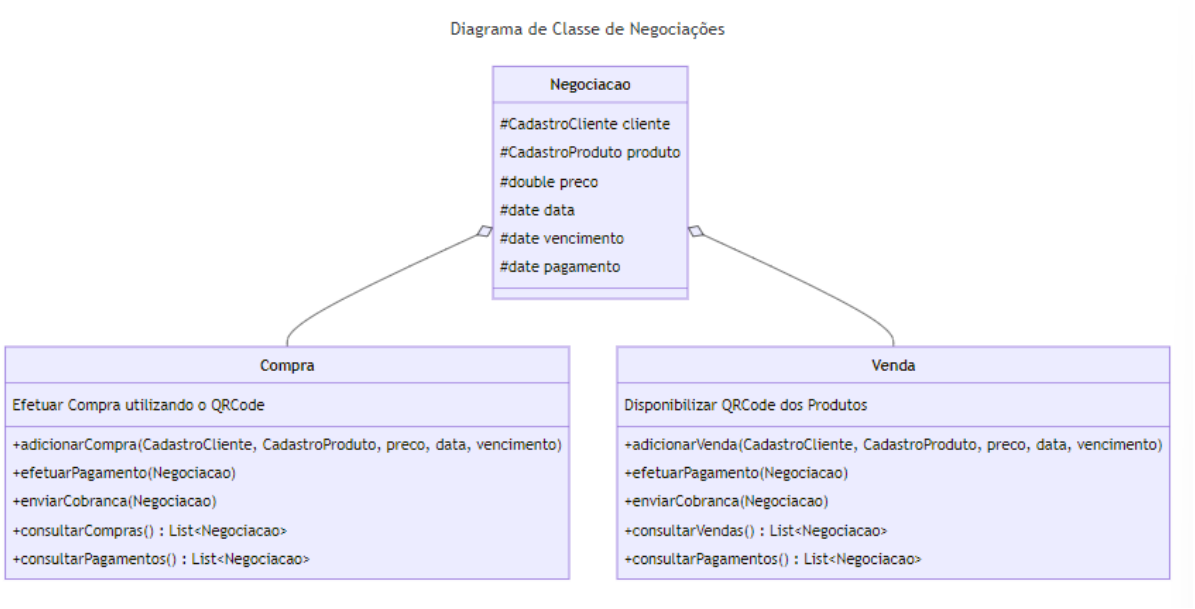

Figura 6 - Diagrama de classe Negociação

A classe "Negociacao" descreve uma transação comercial entre um cliente e um produto. Ela contém informações como o cliente envolvido, o produto em questão, o preço, as datas de realização, vencimento e pagamento da negociação. A classe "Compra" representa o processo de compra, em que é possível adicionar uma nova compra informando os dados do cliente, do produto, o preço, a data, e o vencimento. Além disso, a classe oferece métodos para efetuar o pagamento da negociação, enviar cobranças, consultar as compras realizadas e verificar os pagamentos efetuados.

Por outro lado, a classe "Venda" representa o processo de venda, em que é possível adicionar uma nova venda informando os dados do cliente, do produto, o preço, a data e o vencimento. Assim como na classe "Compra", a classe "Venda" também possui métodos para efetuar o pagamento da negociação, enviar cobranças, consultar as vendas realizadas e verificar os pagamentos efetuados. Essas duas classes estão relacionadas com a classe "Negociacao" por meio de uma associação, indicando que uma negociação pode ser tanto uma compra como uma venda. Essa estrutura de classes permite o gerenciamento de transações comerciais, registrando as informações pertinentes a cada tipo de negociação e oferecendo funcionalidades para controle e consulta das compras, vendas e pagamentos efetuados.

4.2.4. Fiado

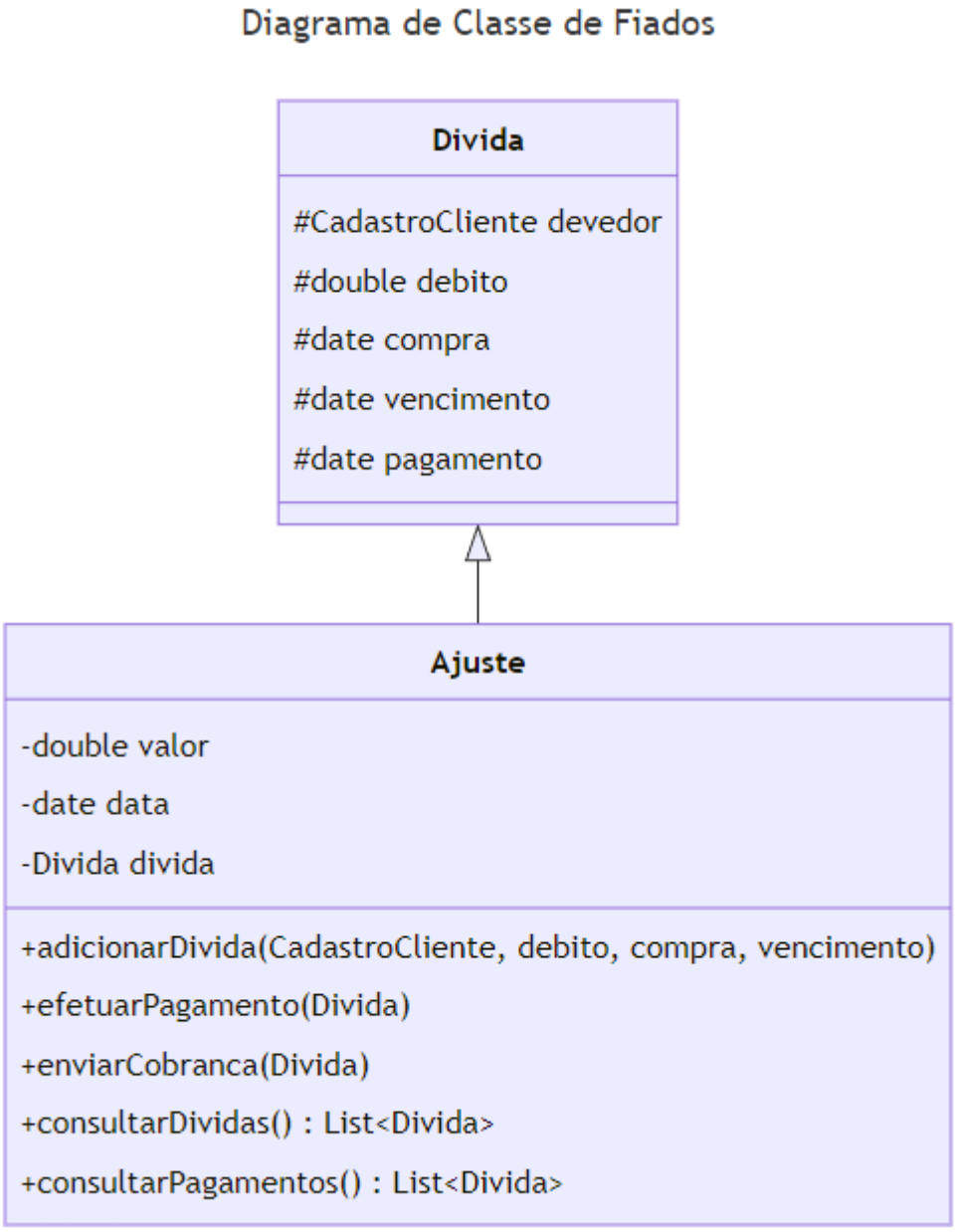

Figura 7 - Diagrama de classe Fiado

A estrutura de classe para o sistema de Fiado é composta pelas classes "Divida" e "Ajuste". A classe "Divida" representa uma dívida registrada para um cliente devedor. Ela contém informações como o cliente envolvido, o valor do débito, as datas de realização, vencimento e pagamento da dívida. A classe "Ajuste" está relacionada com a classe "Divida" por meio de uma herança, indicando que um ajuste é um tipo específico de dívida. A classe "Ajuste" possui atributos privados como valor e data, que representam o valor e a data do ajuste realizado. Além disso, a classe oferece métodos para adicionar uma nova dívida, informando os dados do cliente, do débito,

da compra e do vencimento, efetuar o pagamento de uma dívida, enviar cobranças, consultar as dívidas registradas e verificar os pagamentos efetuados.

Essa estrutura de classes permite o gerenciamento das dívidas de clientes, registrando informações relevantes sobre cada dívida e oferecendo funcionalidades para adicionar novas dívidas, realizar pagamentos, enviar cobranças e consultar tanto as dívidas quanto os pagamentos efetuados. Dessa forma, o sistema de Fiado proporciona um controle eficiente das transações realizadas a crédito e facilita a gestão financeira dos clientes devedores.

4.3. Entidades de Relacionamento

Modelo Entidade-relacionamento é um modelo conceitual amplamente utilizado na área de banco de dados para representar as entidades (objetos) envolvidas em um sistema e os relacionamentos entre elas.

No modelo ER, as entidades são representadas como retângulos e os relacionamentos são representados como linhas que conectam as entidades. Cada entidade possui atributos que descrevem suas características ou propriedades.

Os relacionamentos podem ser de diferentes tipos, como "um para um", "um para muitos" e "muitos para muitos", indicando a cardinalidade das relações entre as entidades. O modelo ER permite visualizar e analisar a estrutura e as relações dos dados em um sistema, facilitando o projeto e a implementação de um banco de dados. Ele fornece uma representação abstrata e fácil de entender dos requisitos e da lógica de negócios de um sistema, servindo como uma base para a criação do esquema de banco de dados.

4.3.1. Cadastro

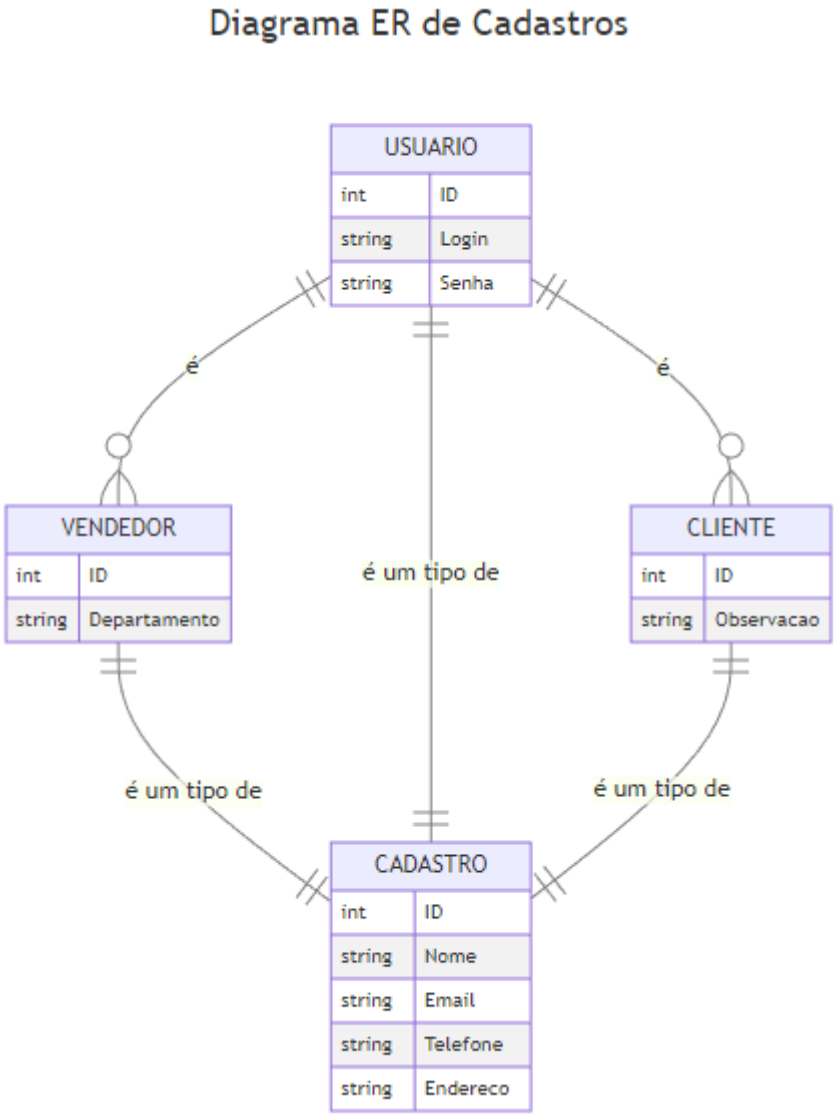

Figura 8 - Diagrama ER de Cadastro

Neste diagrama de modelo entidade-relacionamento (ER), temos quatro entidades principais: "VENDEDOR", "USUARIO", "CLIENTE" e "CADASTRO". A entidade "CADASTRO" representa um registro genérico com atributos como ID, Nome, Email, Telefone e Endereço, que são comuns a vendedores, usuários e clientes.

As entidades "VENDEDOR", "USUARIO" e "CLIENTE" estão relacionadas à entidade "CADASTRO" por meio de uma associação de "é um tipo de". Isso significa que um vendedor, usuário ou cliente é um tipo específico de registro de cadastro.

Além disso, existem relacionamentos adicionais indicados pelas linhas entre as entidades. Por exemplo, a linha entre "USUARIO" e "VENDEDOR" representa que um usuário pode ser associado a um vendedor, indicando uma relação de pertencimento ou vínculo.

Em resumo, esse diagrama de ER descreve a estrutura básica das entidades "VENDEDOR", "USUARIO", "CLIENTE" e "CADASTRO", mostrando suas associações e relacionamentos. Ele serve como um modelo para representar a estrutura de dados e as relações entre essas entidades em um sistema ou banco de dados.

4.3.2. Autenticação

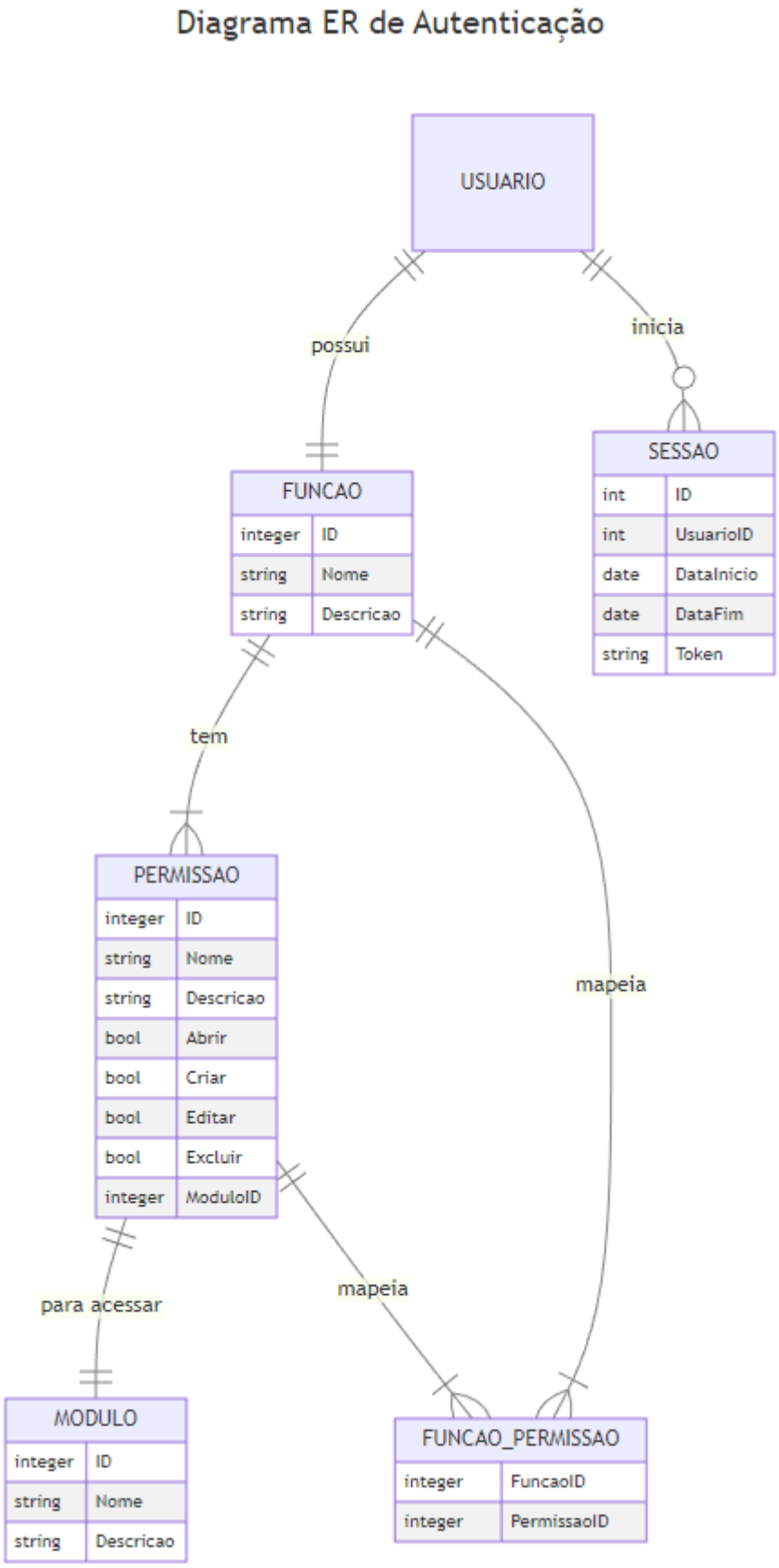

Figura 9 - Diagrama ER de Autenticação

Este diagrama representa um modelo Entidade-Relacionamento (ER) relacionado a funções, permissões, módulos, usuários e sessões. No diagrama, temos a entidade "Função", que possui um ID, nome e descrição. As funções estão associadas às permissões por meio da entidade "Função_Permissão", que mapeia a relação entre elas. As permissões têm um ID, nome, descrição e atributos booleanos que indicam se a permissão permite abrir, criar, editar ou excluir. As permissões também estão relacionadas aos módulos, indicando quais módulos podem ser acessados por meio dessas permissões.

Além disso, temos a entidade "Usuário", que possui uma relação "possui" com a entidade "Função", indicando que um usuário pode ter várias funções associadas a ele. Também temos a entidade "Sessão", que representa o início de uma sessão de usuário e está relacionada ao usuário por meio da relação "inicia". A entidade "Sessão" possui um ID, ID do usuário, data de início, data de fim e um token para autenticação.

4.3.3. Estoque

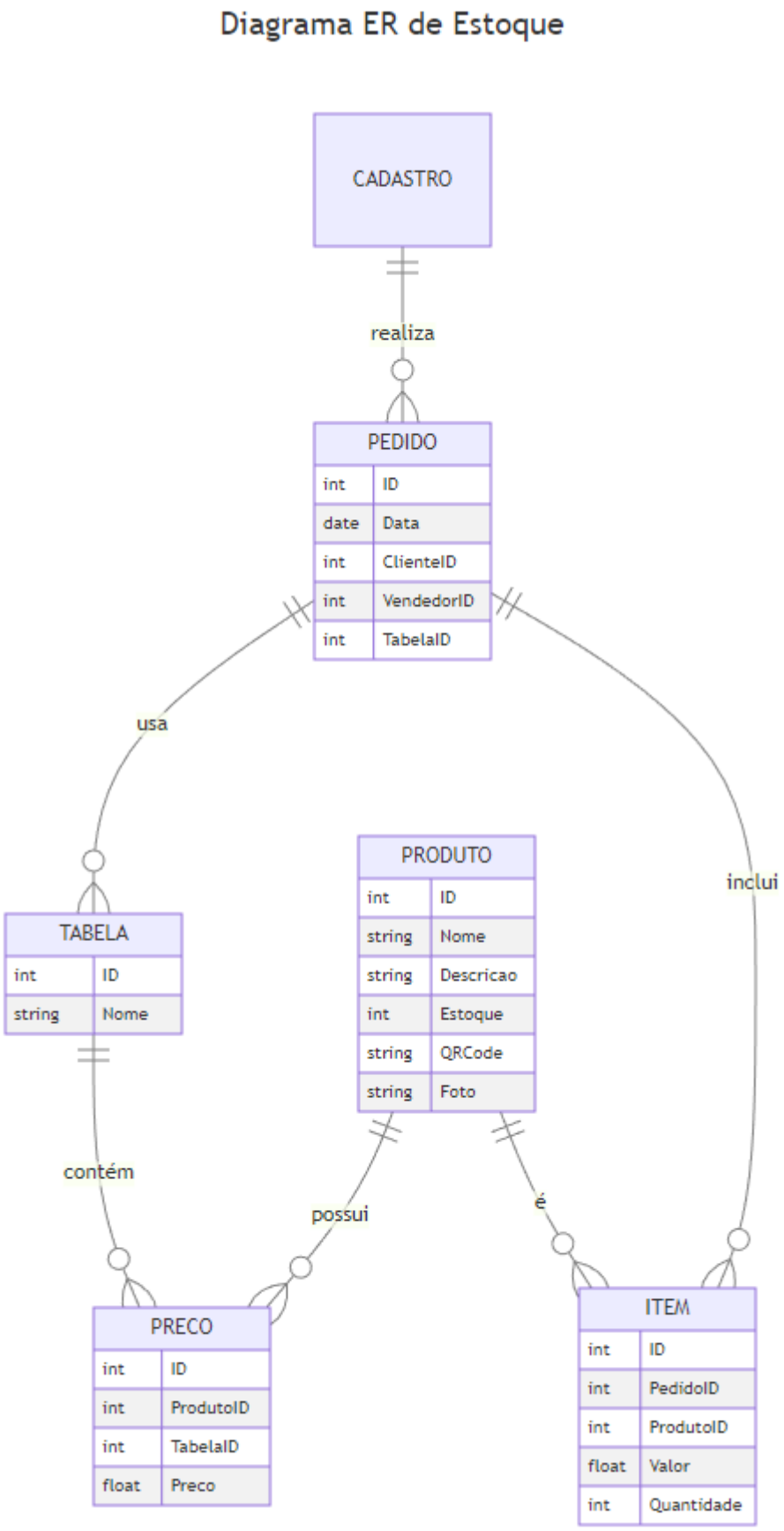
Diagrama ER de Estoque
CADASTRO
realiza
PEDIDO
int ID
date Data
int ClienteID
int VendedorID
int TabelaID
usa
inclui
TABELA
int ID
string Nome
PRODUTO
int ID
string Nome
string Descricao
int Estoque
string QRCode
string Foto
contém
possui
é
PRECO
int ID
int ProdutoID
int TabelaID
float Preco
ITEM
int ID
int PedidoID
int ProdutoID
float Valor
int Quantidade

Figura 10 - Diagrama ER de Estoque

Este diagrama representa um modelo Entidade-Relacionamento (ER) relacionado a produtos, tabelas de preços, pedidos e itens de pedido. No diagrama, temos a entidade "Produto", que possui um ID, nome, descrição, quantidade em estoque, QRCode e foto. Cada produto pode ter um ou mais preços associados a ele, representados pela entidade "Preco". A entidade "Tabela" representa as diferentes tabelas de preços disponíveis e está relacionada à entidade "Preco" por meio da relação "contém". Cada tabela pode ter um ou mais preços associados a diferentes produtos.

Além disso, temos a entidade "Pedido", que possui um ID, data, ID do cliente, ID do vendedor e ID da tabela de preços utilizada para o pedido. A entidade "Item" representa os itens incluídos em um pedido e está relacionada ao pedido e ao produto por meio das relações "inclui" e "é", respectivamente. Cada item possui um ID, ID do pedido, ID do produto, valor e quantidade. Por fim, temos a relação "realiza" entre a entidade "Cadastro" (que não está definida no diagrama) e a entidade "Pedido", indicando que um cadastro (provavelmente associado a um cliente) realiza um pedido.

Esse modelo ER descreve a estrutura básica de produtos, tabelas de preços, pedidos e itens de pedido, permitindo o registro e a associação de informações relacionadas a essas entidades. Ele pode ser utilizado como base para o desenvolvimento de um sistema de gerenciamento de estoque, vendas e pedidos.

4.4. Estados do Aplicativo

Diagrama de estados é uma ferramenta da UML (Unified Modeling Language) utilizada para modelar o comportamento de um sistema. Ele mostra os possíveis estados de um objeto e como ele pode mudar de um estado para outro ao longo do tempo. O diagrama de estados é útil para entender o comportamento de um sistema e para projetar sistemas que se comportam de maneira diferente em função do estado atual.

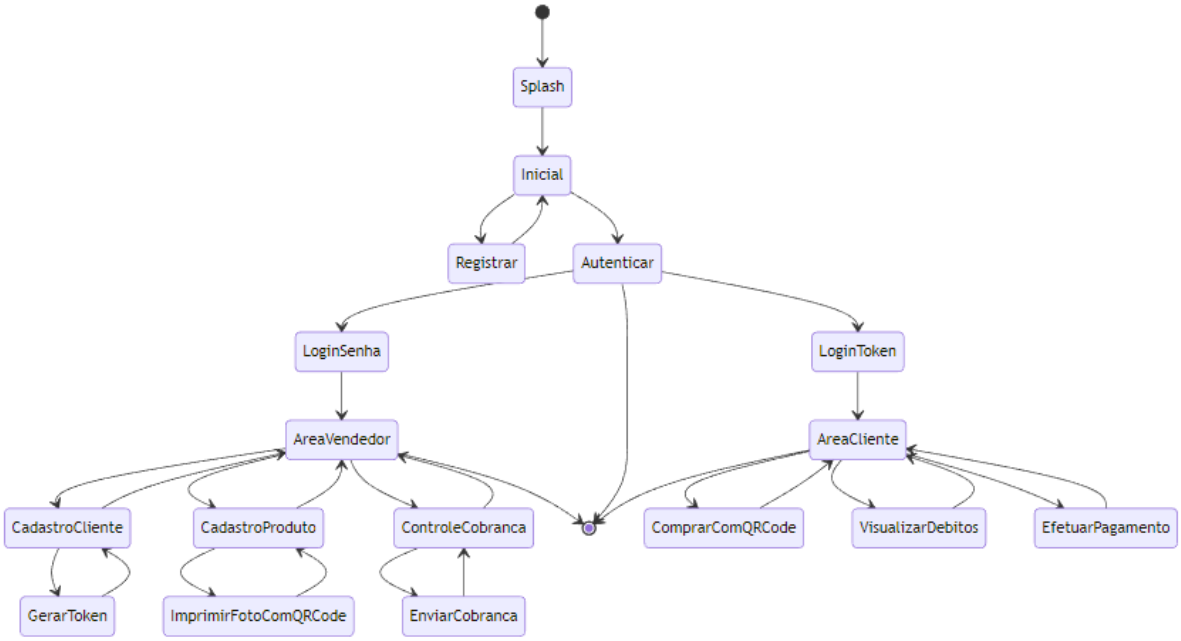

Figura 11 - Diagrama de Estados do Aplicativo

O processo começa no estado "Splash", onde ocorre uma apresentação inicial. Em seguida, o sistema transita para o estado "Inicial", onde o usuário pode escolher entre registrar-se ou autenticar-se.

Se o usuário optar por registrar-se, ele entra no estado "Registrar" e depois retorna ao estado "Inicial". Se o usuário escolher autenticar-se, ele avança para o estado "Autenticar".

No estado "Autenticar", o usuário tem duas opções: login com senha ou login com token. Se o usuário escolher login com senha, ele entra no estado "LoginSenha" e, em seguida, transita para o estado "AreaVendedor" após autenticação bem-sucedida.

No estado "AreaVendedor", o vendedor pode acessar diferentes funcionalidades, como cadastrar clientes, cadastrar produtos, controlar cobranças, etc. Cada transição de estado dentro do estado "AreaVendedor" indica a interação com uma determinada funcionalidade e o retorno ao estado "AreaVendedor" após a conclusão.

Se o usuário optar por login com token, ele entra no estado "LoginToken" e, em seguida, transita para o estado "AreaCliente" após autenticação bem-sucedida.

No estado "AreaCliente", o cliente tem acesso a funcionalidades como comprar utilizando QRCode, visualizar débitos e efetuar pagamentos. Novamente, cada transição de estado dentro do estado "AreaCliente" indica a interação com uma funcionalidade específica.

No final de cada fluxo de interação, seja do vendedor ou do cliente, o sistema retorna ao estado inicial "[*]".

4.5. Protótipos da Interface

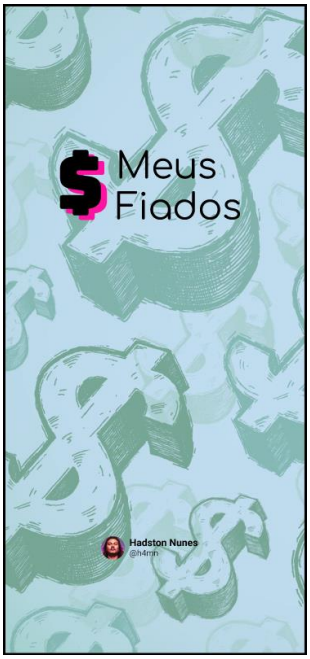

Figura 12 - Protótipo da tela Splash

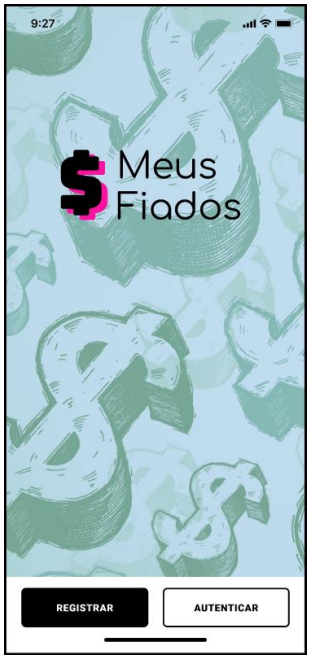

Figura 13 - Protótipo da tela Inicial

Figura 14 - Protótipo da tela Registrar

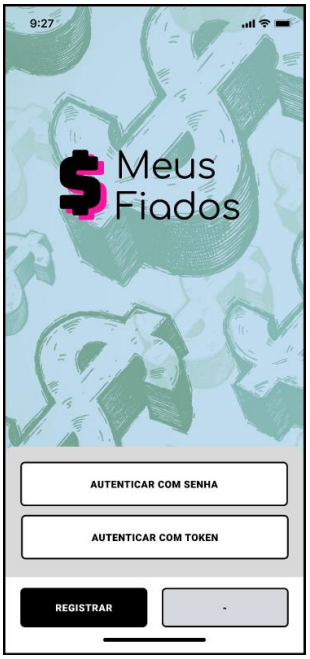

Figura 15 - Protótipo da tela Tipos de Autenticação

Figura 16 - Protótipo da tela Autenticar com Senha

Figura 17 - Protótipo da tela Autenticar com Token

Figura 18 - Protótipo da tela de Vendedor

Figura 19 - Protótipo da tela de Cliente

5. METODOLOGIA

A metodologia adotada para este projeto será baseada em Scrum. Scrum é um conjunto de recomendações para agilidade amplamente utilizado em desenvolvimento de software para demonstrar o ciclo de desenvolvimento. Esta metodologia é caracterizada pelos ciclos iterativos e incrementais, chamados de “sprints”, que geralmente têm duração de quatro semanas. Cada sprint é composto por algumas etapas, incluindo o desenvolvimento do projeto, o planejamento, a definição das tarefas a serem realizadas, o desenvolvimento, os testes e a revisão do progresso. Ao final de cada sprint é realizada uma retrospectiva para avaliar e revisar os resultados alcançados, fazer reajustes e planejar o próximo sprint.

Os resultados do projeto são avaliados com base na satisfação do cliente, eficiência do processo do gerenciamento do fiado e a facilidade de uso do aplicativo; através da coleta de feedback dos usuários e da realização de testes, será possível identificar possíveis melhorias e ajustes a serem feitos.

No contexto do aplicativo de gerenciamento de fiados, a seguinte organização foi elaborada:

Sprint 1 – 22 de fevereiro de 2023:

- Etapa 1: Pesquisa sobre necessidades e similares;
- Etapa 2: Início da documentação e levantamento de requisitos;
- Etapa 3: Pesquisa sobre possíveis ferramentas para o projeto;

Sprint 2 – 26 de abril de 2023

- Etapa 1: Pesquisa de revisão bibliográfica e embasamento teórico;
- Etapa 2: Início da modelagem dos diagramas de caso de uso;
- Etapa 3: Modelagem dos diagramas de classes, entidades de relacionamento;

Sprint 3 – 22 de junho de 2023

- Etapa 1: Modelagem dos estados do aplicativo;
- Etapa 2: Modelagem do protótipo da interface;
- Etapa 3: Finalização da documentação do projeto;

Sprint 4 – 25 de agosto de 2023

- Etapa 1: Pesquisar sobre MVP;
- Etapa 2: Iniciar implementação do MVP;
- Etapa 3: Iniciar aulas de Engenharia 3;

Ao longo desta seção foi apresentada diferentes etapas realizadas com o intuito de estrutura o projeto de forma eficiente. Desde o primeiro sprint que foi dedicado a pesquisa e a busca pela estruturação do projeto através da correta documentação, a seguir muita concentração da leitura e pesquisa para embasar os conceitos apresentados. Esta estruturação permitiu avançar progressivamente nas diferentes etapas do projeto garantindo um desenvolvimento consistente e alinhado aos objetivos iniciais.

6. CONCLUSÃO

Durante o desenvolvimento deste projeto, foram realizadas várias etapas de pesquisa, análise e implementação, com o objetivo de criar um sistema de gerenciamento de estoque eficiente. Inicialmente, foi realizada uma extensa revisão da literatura para entender as melhores práticas e abordagens utilizadas no campo. Em seguida, procedeu-se à coelta de dados e à análise de requisitos, o que permitiu identificar os principais desafios enfrentados pelas empresas de comércio informal. Com base nestas informações o projeto foi sendo desenvolvido alcançando bons resultados e resultados inesperados. Em resumo, acredito que este projeto alcançou uma solução promissora, eficiente e confiável, que tem um grande potencial para auxiliar empresas e até mesmo entidades físicas a otimizar suas operações no gerenciamento do comércio informal.

7. REFERÊNCIAS

Silva, L. A. (2018). Estudo da Influência da Mineiridade na Relação entre Clientes e Micro e Pequenas Empresas. *Universidade Federal de Ouro Preto - Instituto de Ciências Sociais e Aplicadas*, p. 22.

SOMMERVILLE, I. (2011). *Engenharia de software.* São Paulo: Pearson Education do Brasil.