Sistema de Cafeteria

Documentação do Projeto de Programação Orientada a Objetos

Integrantes:
Fernando Ikeda – Prontuário: GU30557897
Gabriel Gomes – Prontuário: GU3055361

Professor: Cleber

Disciplina: Programação Orientada a Objetos

Sumário

1. Objetivo do Projeto

O sistema de cafeteria foi desenvolvido com o objetivo de aplicar os princípios de Programação Orientada a Objetos, acesso a banco de dados e construção de interfaces gráficas em Java. A proposta é criar um sistema onde diferentes tipos de usuários (clientes, funcionários e administradores) possam interagir com funcionalidades específicas, como login, menu e módulos de cadastro (CRUD).

2. Especificação dos Requisitos

```
Definições de Recursos Funcionais e não Funcionais

RF - Tudo aquilo que faz parte do programa, do que
ele vai fazer. Como cadastrar aluno, ativar aluno,
assim por diante.
RNF - Parte de fora do programa, que complementa o
sistema, que faz parte do sistema. Como leitor de
códigos de barras, impressora, APIs.


RF001  - Adicionar lote no estoque
    \Adicionar os dados do produto do estoque, tipo
de produto, nome, validade, marca, fornecedor e
lote. Geralmente feito pelo estoquista ou quem cuida
do estoque no geral.

RF002  - Definir como em uso
```

\Ele deve definir o lote como "em_uso" após aberto. Geralmente feito pelo estoquista atravez do próprio programa.

RF003 - Zerar lote do estoque

\Após o termino de um lote, ele deve ser declarado como "acabado", assim identificado seu estado. Pode ser feito pelo estoquista ou cozinheiro que tiver terminado de usar o lote.

RF004 - Adicionar Receitas

\Adicionar receitas ao sistema, contendo suas quantidades, produtos ligados a ele e modo de preparo. Feito pelo cozinheiro.

RF005 - Mostrar Receitas

\Exibir receitas que foram adicionadas anteriormente. Geralmente acessadas pelo cozinheiro.

RF006 - Atualizar Receitas

\As vezes o cozinheiro pode optar por mudar uma receita que antes já existia, colocando um novo ingrediente ou mudando o modo de preparo, para isso

essa nova função seria necessária. Pode ser acessada pelo cozinheiro.

RF007 - Adicionar produtos ao sistema

\Adicionar novos produtos ao sistema, geralmente quando produtos limitados ou novos são adicionados ao cardápio e não foi feito um cadastro a ele anteriormente. Pode ser acessado pelo estoquista ou cozinheiro.

RF008 - Remover produtos do sistema

\Produtos que não são mais utilizados ou que não serão mais servidos, devem ser removidos do sistema. Pode ser acessado pelo estoquista ou pelo cozinheiro.

RF009 - Adicionar fornecedores

\Fornecedores devem ser adicionados ao sistema para qe deste modo haja um melhor controle sobre os lotes que são comprados, relacionando o id dos forncedores aos lotes que eles mandam. Facilitando sua visualização do sistema. Geralmente feita por

estoquista ou gerente de estoque. (Quem controla toda gestão)

RF010 - Remover fornecedores

\Do mesmo modo que dá pra adicionar fornecedores, deve ser possível remover eles do sistema. Acessado por Gestor do Estoque.

RF011 - Cadastrar funcionários

\Adicionar novos logins de acesso a novos funcionários, vinculando a ele a sua função dentro da cafeteria. Acessado pelo RH, ou quem faz a gestão dos funcionáros no geral, como gerentes.

RF012 - Remover cadastro de funcionários

\Remover cadastro de funcionários demitidos para evitar que possam se logar ao sistema novamente. Acessado pelo RH ou Gerente.

RF013 - Alterar dados de funcionáros

\Em caso de erros ao inserir os dados seria interessante que houvesse essa opção, editando as informações de forma simples e adequada. Acessado

pelo RH, Gerente ou pelo próprio funcionário em casos de mudanças simples como nome ou cpf.

RF014 - Marcar ponto de horário de saída e de entrada de funcionários

    \Após a inserção da digital no relógio de ponto biométrico, guardar o horário que ele entrou na cafeteria, sendo a mesma coisa no horário de saída. Feito por relógio de ponto biométrico.

-- Recursos Não Funcionais --

RNF001 - Relógio de ponto biométrico

    \Integração com um relógio de ponto biométrico para marcação automática de horário de entrada e saída de funcionários. Deve ser capaz de enviar os dados ao sistema via API ou arquivo exportável.

RNF002 - Impressora de pedidos para cozinha

    \Integração com uma impressora térmica para impressão de comandas na cozinha. Cada novo pedido aprovado pelo sistema deve gerar automaticamente uma comanda impressa.

RNF003 - Leitor de códigos de barras

\Suporte a leitor de código de barras para facilitar a entrada de produtos no estoque e a consulta de produtos. O sistema deve reconhecer o código de barras e preencher automaticamente os campos do produto.

RNF004 - API de controle de estoque

\O sistema deve oferecer uma API RESTful que permita a integração com outros sistemas de gestão (ERP, controle de compras, etc.). A API deve permitir consulta e atualização de estoque.

RNF005 - Backup automático

\O sistema deve possuir um mecanismo de backup automático diário do banco de dados, garantindo a integridade dos dados em caso de falhas ou perda de conexão.

RNF006 - Sistema responsivo

\A interface do sistema deve ser responsiva, permitindo acesso tanto em computadores desktop

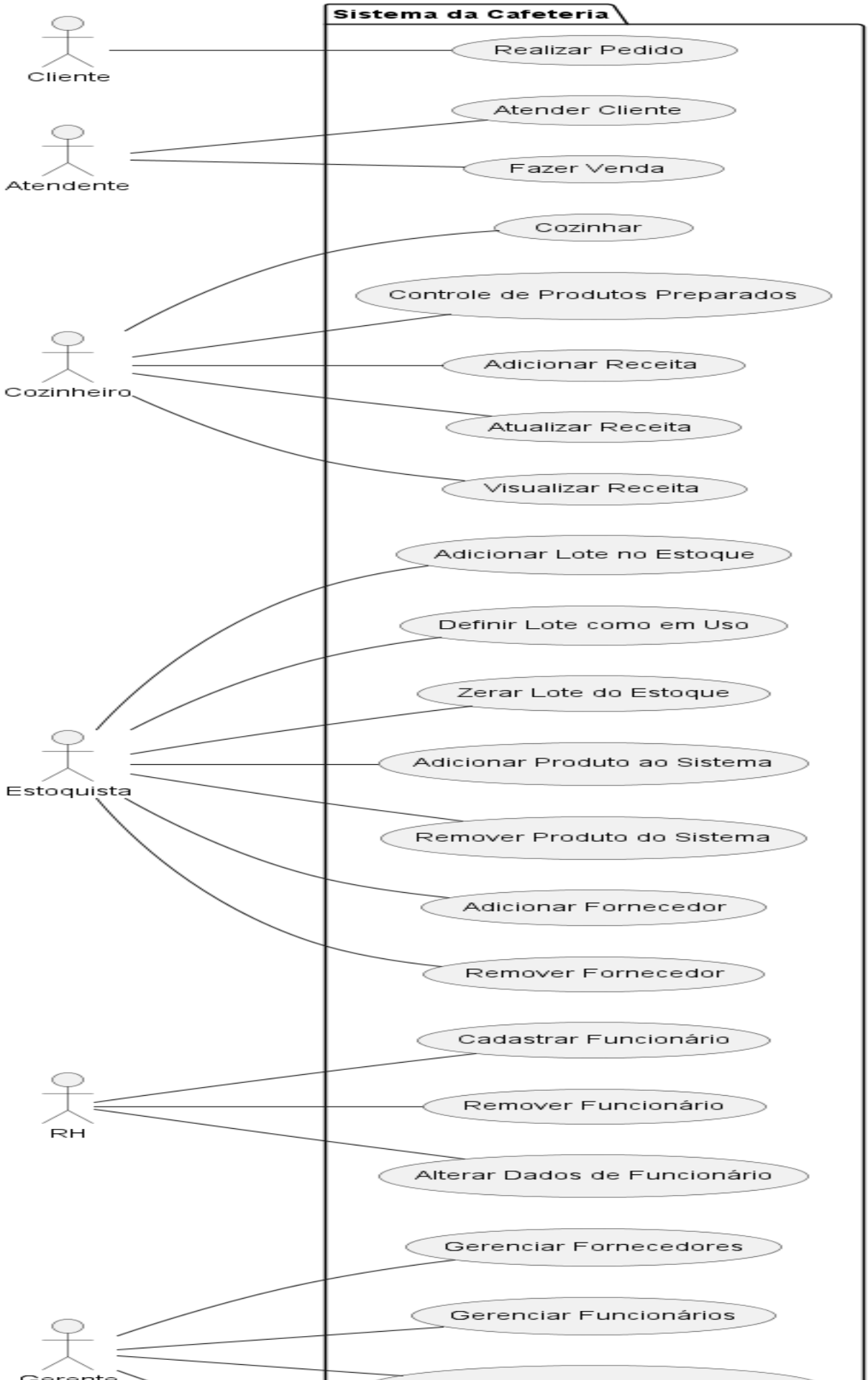
```
quanto em tablets e smartphones (ex.: para o
cozinheiro consultar receitas em um tablet na
cozinha).

RNF007 - Controle de permissões e segurança
    \O sistema deve implementar um mecanismo robusto
de controle de permissões baseado em perfis (RH,
gerente, estoquista, cozinheiro). Deve garantir que
cada função só consiga acessar as áreas autorizadas.

RNF008 - Desempenho
    \O sistema deve ser capaz de responder em até 2
segundos em operações comuns como: Consulta de
receitas, Consulta de estoque, Consulta de pedidos.
Garantir uma boa experiência mesmo com muitos
registros no banco.
```

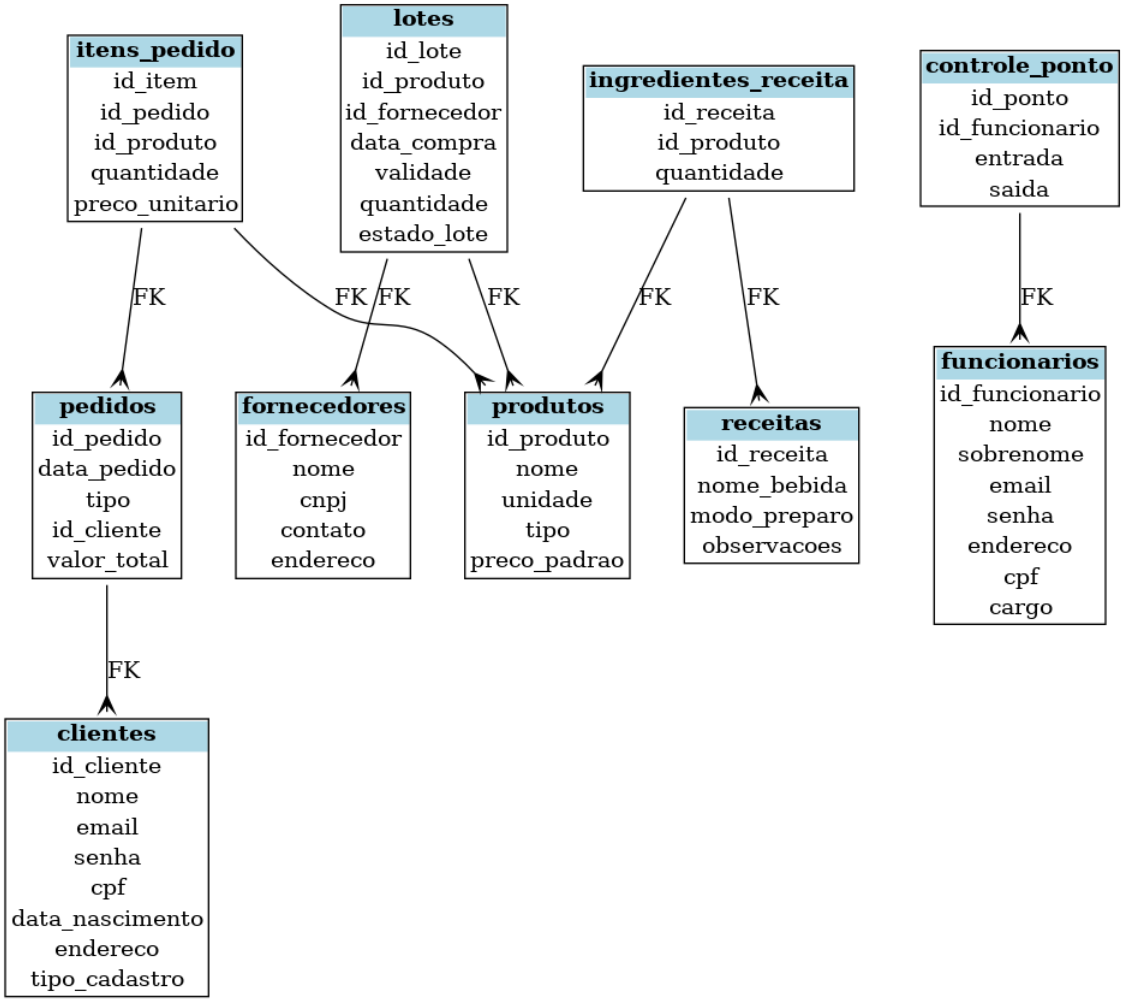
3. Diagrama de Caso de Uso

O diagrama abaixo representa os principais casos de uso relacionados aos diferentes tipos de usuários do sistema:


Sistema da Cafeteria
Cliente
Realizar Pedido
Atender Cliente
Atendente
Fazer Venda
Cozinhar
Controle de Produtos Preparados
Cozinheiro
Adicionar Receita
Atualizar Receita
Visualizar Receita
Adicionar Lote no Estoque
Definir Lote como em Uso
Zerar Lote do Estoque
Estoquista
Adicionar Produto ao Sistema
Remover Produto do Sistema
Adicionar Fornecedor
Remover Fornecedor
Cadastrar Funcionário
RH
Remover Funcionário
Alterar Dados de Funcionário
Gerenciar Fornecedores
Gerenciar Funcionários

4. Modelo Entidade-Relacionamento

O modelo abaixo descreve as entidades do banco de dados e seus relacionamentos:



5. O que será implementado

O sistema de cafeteria foi desenvolvido na linguagem Java, utilizando o padrão de arquitetura MVC (Model-View-Controller), com foco na separação de responsabilidades e facilidade de manutenção. A interface gráfica foi construída com Java Swing (JFrame), e a conexão com banco de dados é realizada via JDBC.

Organização dos Pacotes:

main	Contém a classe principal **Main.java**, responsável por iniciar a aplicação.
connection	Contém a classe **ConnectDB.java**, responsável pela conexão com o banco de dados.
classes	Contém as classes de regra de negócio, como **Funcionario**, **Login**, **Cozinheiro** e **RH**.
frames	Contém as interfaces gráficas (telas), como **LoginFrame**, **RHFrame** e **CozinheiroFrame.**

Resumo das Principais Classes:

Main.java:

Classe principal que inicializa o sistema e exibe a tela de login.

```
public class Main {
        public static void main(String[] args) {
                new LoginFrame();
        }
}
```

ConnectDB.java:

Classe utilitária para conexão com o banco de dados MySQL.

```
public class ConnectDB {
        public static Connection getConnection() throws SQLException {
                return DriverManager.getConnection(
```

```
                "jdbc:mysql://localhost:3306/cafeteria", "root", "");
        }
}
```

Login.java:

Responsável por autenticar o usuário a partir de e-mail e senha.

```
public class Login {
        public static Funcionario autenticar(String email, String senha) {
                // Verifica no banco se o email e senha correspondem a um funcionário
        }
}
```

Funcionario.java:

Classe modelo com atributos comuns a todos os funcionários.

```
public class Funcionario {
        private String nome;
        private String cpf;
        private String cargo;
// getters, setters, construtores
}
```

RH.java e Cozinheiro.java

Extensões da classe Funcionario, com comportamentos específicos por perfil.

LoginFrame.java

Interface gráfica inicial, com campos de login e botão de entrada.

RHFrame.java e CozinheiroFrame.java:

Telas apresentadas após o login, conforme o tipo de usuário.

Cada tela contém botões e ações específicas para o setor correspondente (ex: cadastro de funcionário, visualização de receitas).

Funcionalidades Já Implementadas

Tela de login com autenticação básica

Identificação de tipo de usuário (RH ou Cozinheiro)

Redirecionamento para a interface correspondente

Conexão com banco de dados via ConnectDB

Estrutura inicial de CRUD pronta para ser estendida

Organização clara por camadas (MVC)

Funcionalidades em Desenvolvimento

CRUD completo para produtos, receitas, funcionários

Integração com leitor de código de barras e relógio de ponto (futuro RNF)

Inserção de controle de permissões baseado em perfil

6. Conclusão

O projeto visa consolidar os conhecimentos adquiridos na disciplina de Programação Orientada a Objetos, integrando banco de dados, interface

gráfica e estrutura em camadas. Os próximos passos incluem a finalização do banco, implementação dos métodos CRUD e testes das funcionaonalidades.