ESPECIFICAÇÕES - PROJETO FINAL SISTEMA DE PEDIDOS DE PIZZARIA

Acesse o link do GitHub:

https://github.com/kaylannesantos/pizzariaDelivery

Visão Geral do Sistema

1. **Nome do projeto**: Pizzaria - Sistema de Controle de Pedidos
2. **Objetivo principal**: Automatizar o processo de pedidos de pizza, facilitando o registro de clientes e pedidos, garantindo persistência dos dados e permitindo consulta e edição de pedidos.
3. **Público-alvo**: Pizzarias e seus clientes.
4. **Problema que resolve**: Elimina a necessidade de registros manuais, reduzindo erros e otimizando o tempo de atendimento.
5. **Pré-requisitos**:
    a. Biblioteca `psycopg2` para conectar ao PostgreSQL. Instale comd:

```
pip install psycopg2
```

    b. PostgreSQL instalado e configurado.

Arquitetura do Sistema

1. **Arquitetura**: MVC (Model-View-Controller)
2. **Tecnologias Utilizadas**:
    a. **Frontend**: Interface Gráfica em Tkinter (Python)
    b. **Backend**: Python
    c. **Banco de Dados**: PostgreSQL
    d. **APIs e Integrações**: Nenhuma integração externa
    e. **Hospedagem**: Localhost, sem suporte a nuvem

Configuração do Banco de Dados

Criar Banco de Dados e Tabela

1. Crie o banco de dados "pizzaria":

```
CREATE DATABASE pizzaria;
```

2. Acesse o banco de dados criado:

```
\c pizzaria
```

3. Crie a tabela Cliente:

```
CREATE TABLE clientes (
```

```
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    telefone VARCHAR(20) NOT NULL,
    endereco TEXT NOT NULL,
    bairro VARCHAR(50) NOT NULL);
```

4. Crie a tabela Sabores e faça o insert:

```
CREATE TABLE sabores (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL);
INSERT INTO sabores (nome) VALUES
('Mussarela'),
('Calabresa'),
('Frango com Catupiry'),
('Portuguesa'),
('Quatro Queijos'),
('Napolitana'),
('Pepperoni'),
('Veggie'),
('Margherita'),
('Barbecue Chicken');
```

5. Crie a tabela Tamanho:

```
CREATE TABLE tamanho (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    quantidade INT NOT NULL,
    tamanho_cm INT NOT NULL,
    valor REAL NOT NULL
);
```

```sql
INSERT INTO tamanho (nome, quantidade, tamanho_cm, valor) VALUES

('Pequeno', 4, 25, 15.00),

('Médio', 6, 30, 30.00),

('Grande', 8, 40, 45.00),

('GG', 12, 50, 55.00);
```

6. Crie a tabela pedidos:

```sql
CREATE TABLE pedidos (
    id SERIAL PRIMARY KEY,
    cliente VARCHAR(100) NOT NULL,
    endereco VARCHAR(255) NOT NULL,
    tamanho VARCHAR(50) NOT NULL,
    sabor VARCHAR(100) NOT NULL,
    data_hora TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7. E por fim, a tabela Item pedido:

```sql
CREATE TABLE itens_pedido (

  id SERIAL PRIMARY KEY,

  id_pedido INT NOT NULL REFERENCES pedidos(id),

  id_sabor INT NOT NULL REFERENCES sabores(id),

  id_tamanho INT NOT NULL REFERENCES tamanho(id),

  quantidade_item INT NOT NULL,

  valor_total_item REAL NULL);
```

Configuração do Código

I. Clone este repositório ou copie o arquivo do código para o seu ambiente:

https://github.com/kaylannesantos/pizzariaDelivery.git

II. Abra o arquivo Python e configure os dados da conexão com o banco de dados na função conectar_bd:

```python
def conectar_bd():
    return psycopg2.connect(
        dbname="pizzaria",
        user="seu_usuario",  # Substitua pelo seu usuário do PostgreSQL
        password="sua_senha",  # Substitua pela sua senha do PostgreSQL
        host="localhost",
        port="5432"
    )
```

Como Executar

I. Execute o arquivo Python no terminal:

```
python pizzaria.py
```

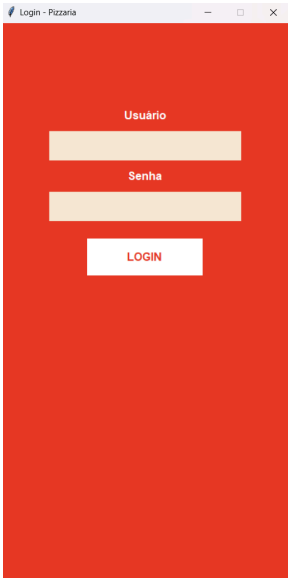
II. Faça login com as credenciais padrão:
   A. **Usuário:** admin
   B. **Senha:** 1234

III. Preencha os dados do cliente e do pedido na interface gráfica.

IV. Clique em "Registrar Pedido" para salvar o pedido no banco de dados PostgreSQL.

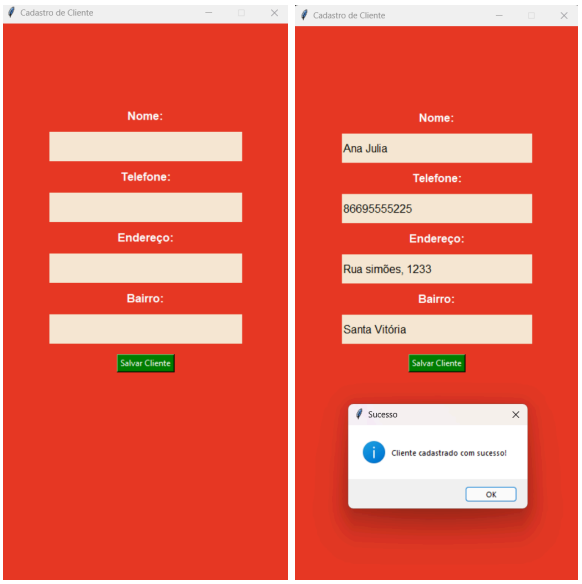
Telas da Interface

1. Tela de Login
   a. Usuário e Senha
   b. Botão login



2. Home
   a. Botões de
      i. Registro de pedidos
      ii. Cadastro de clientes
      iii. Histórico dos pedidos feitos
      iv. Sair → Volta para a tela login

3. Cadastrar Cliente
    a. Campos de preencher
        i. Nome
        ii. Telefone
        iii. Endereço
        iv. Bairro
    b. Botão de Salvar
        i. Após salvar o cliente exibe a mensagem



4. Registro dos pedidos
    a. Botão Verificar
        i. Busca o cliente no banco;
        ii. Se caso não tiver cadastrado ele pode cadastrar
    b. Campos de preencher
        i. Cliente

1. Botão verificar cliente no banco de dados e autopreencher endereço

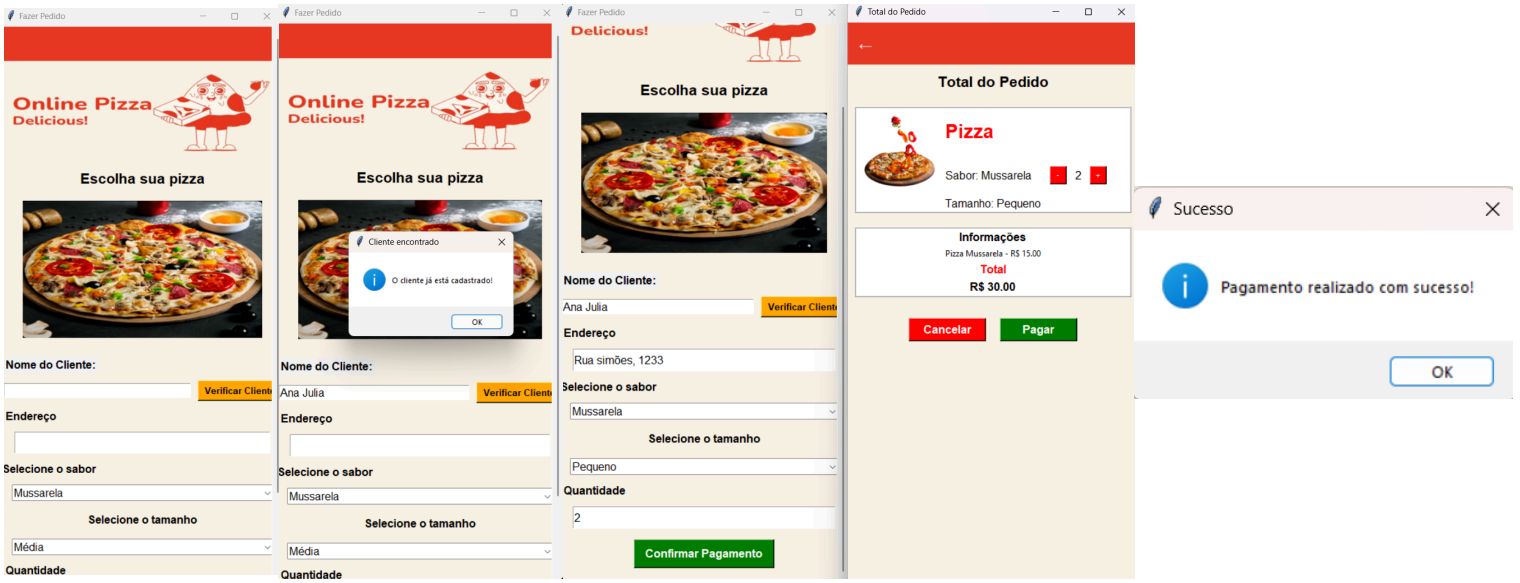
ii. Endereço
1. Se o cliente estiver cadastrado ele se auto preenche

iii. Escolher o sabor da pizza

iv. Escolher o tamanho da pizza

v. Escolher o quantidade de pizza

c. Botão confirmar pagamento → Leva para tela de pagamento



5. Histórico de pedidos feitos
   a. Frames com informações sobre os pedidos feitos

Regras de Negócio

1. O cliente pode pedir mais de uma pizza por pedido.
2. O atendente anota o pedido no sistema.
3. A pizza pode ter quatro tamanhos: Pequena, Média, Grande e Gigante.

4. Cada pizza tem apenas um sabor.
5. O atendente registra os seguintes dados:
    a. Cliente: Nome, telefone, endereço, bairro.
    b. Pedido: Sabor da pizza, tamanho, quantidade.

Funcionalidades Principais

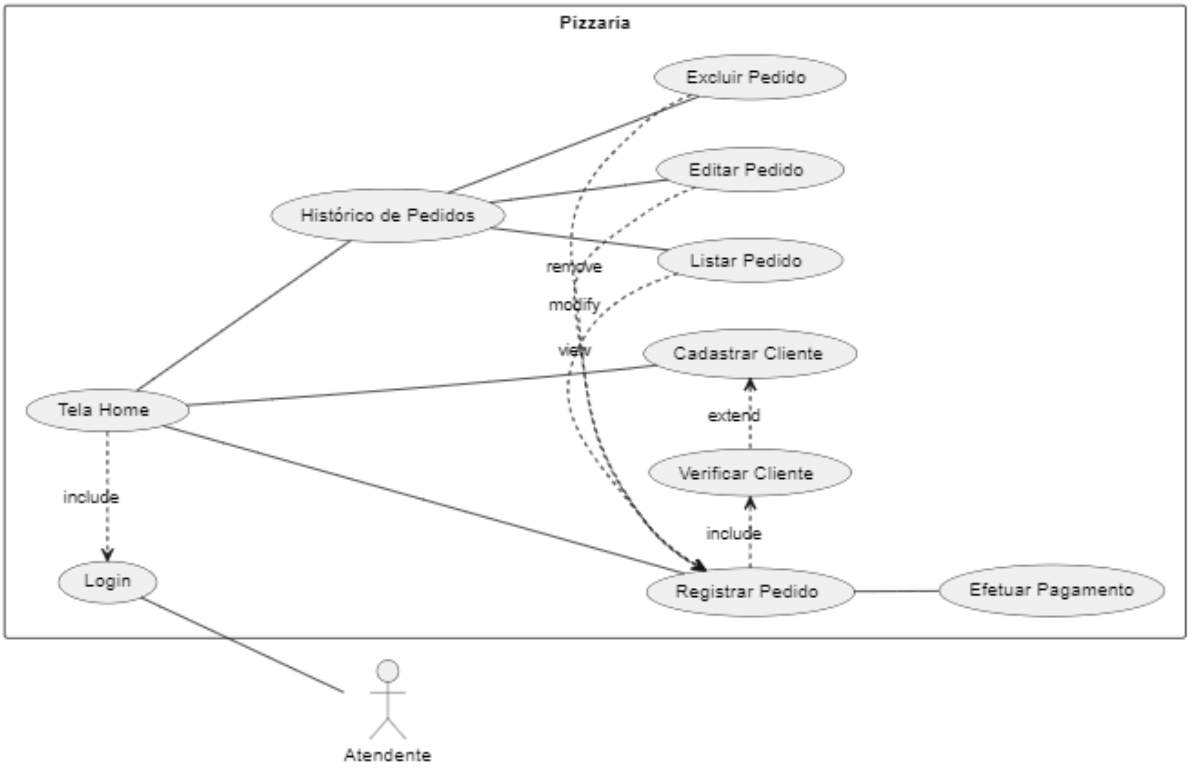
1. Login do atendente
2. Cadastro de clientes
3. Registro de pedidos
4. Listagem de pedidos
5. Controle de status do pedido(pagar pedido)

Método para implementação - Funcionalidade Pagar Pedido

```
def atualizar_total():
    nova_quantidade = int(quantidade_var.get())
    total = valor_tamanho * nova_quantidade
    total_label.config(text=f"R$ {total:.2f}")

def aumentar_quantidade():
    quantidade_var.set(quantidade_var.get() + 1)
    atualizar_total()

def diminuir_quantidade():
    if quantidade_var.get() > 1:
        quantidade_var.set(quantidade_var.get() - 1)
        atualizar_total()
```

Documentação

Diagrama de Casos de Uso



Descrição - Casos de Uso

Caso de Uso 1: Login

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente faça login no sistema para acessar as funcionalidades de gerenciamento.
3. **Pré-condições:**
    a. O atendente possui uma conta registrada no sistema.
4. **Pós-condições:**
    a. O atendente é autenticado e tem acesso ao painel de administração.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de login.
    b. **Passo 2:** O atendente insere o nome de usuário e senha.
    c. **Passo 3:** O sistema valida as credenciais.
    d. **Passo 4:** Se as credenciais forem válidas, o atendente é redirecionado para o painel de administração.
    e. **Passo 5:** Se as credenciais forem inválidas, o sistema exibe uma mensagem de erro.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Senha Incorreta):**
        i. Se a senha estiver incorreta, o sistema exibe uma mensagem de erro solicitando ao atendente tentar novamente.
    b. **Fluxo Alternativo 2 (Usuário Não Encontrado):**
        i. Se o usuário não for encontrado, o sistema exibe uma mensagem informando que o nome de usuário não está registrado.

Caso de Uso 2: Cadastrar Cliente

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente cadastre um novo cliente no sistema com seus dados pessoais.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema.
4. **Pós-condições:**
    a. O novo cliente é registrado no sistema e pode realizar pedidos.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de cadastro de cliente.
    b. **Passo 2:** O atendente insere as informações do cliente: nome, telefone, endereço e bairro.
    c. **Passo 3:** O sistema valida os dados inseridos.
    d. **Passo 4:** O sistema confirma o cadastro do cliente e o adiciona ao banco de dados.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Dados Inválidos):**
        i. Se algum dado estiver incorreto ou faltando, o sistema exibe uma mensagem de erro solicitando correção.

Caso de Uso 3: Verificar Cliente

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente verifique os dados de um cliente existente.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema.
4. **Pós-condições:**
    a. O atendente visualiza os dados do cliente selecionado.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de pesquisa de cliente.
    b. **Passo 2:** O atendente insere o nome ou telefone do cliente.
    c. **Passo 3:** O sistema exibe os resultados de clientes correspondentes.
    d. **Passo 4:** O atendente seleciona o cliente desejado para visualizar os dados completos.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Cliente Não Encontrado):**
        i. Se o cliente não for encontrado, o sistema exibe uma mensagem de erro.

Caso de Uso 4: Registrar Pedido

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Registrar um novo pedido para um cliente.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema e o cliente já está cadastrado.
4. **Pós-condições:**
    a. O pedido é registrado no sistema, associado ao cliente.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de registrar pedido.
    b. **Passo 2:** O atendente seleciona o cliente para o qual o pedido será feito.
    c. **Passo 3:** O atendente insere os itens do pedido (como pizzas, bebidas, etc.).
    d. **Passo 4:** O sistema calcula o valor total do pedido.
    e. **Passo 5:** O atendente confirma os itens e o valor.
    f. **Passo 6:** O sistema registra o pedido e exibe uma confirmação.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Itens Não Disponíveis):**
        i. Se algum item não estiver disponível, o sistema informa ao atendente e oferece alternativas.

Caso de Uso 5: Editar Pedido

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente edite os detalhes de um pedido existente.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema e o pedido já foi registrado.
4. **Pós-condições:**
    a. O pedido é atualizado com os novos dados fornecidos.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1: O** atendente acessa a tela de editar pedido.
    b. **Passo 2:** O atendente seleciona o pedido que deseja editar.
    c. **Passo 3:** O atendente altera os itens do pedido conforme necessário.
    d. **Passo 4:** O sistema recalcula o valor total do pedido.
    e. **Passo 5:** O atendente confirma as alterações.
    f. **Passo 6:** O sistema atualiza o pedido e exibe uma mensagem de confirmação.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Pedido Não Encontrado):**
        i. Se o pedido não for encontrado, o sistema exibe uma mensagem de erro.

Caso de Uso 6: Excluir Pedido

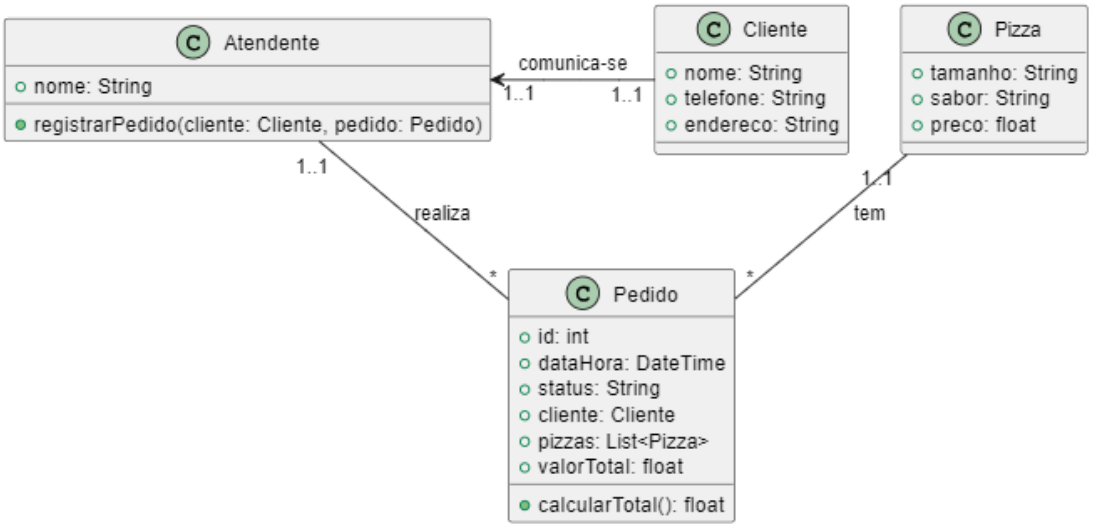
1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente exclua um pedido do sistema.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema e o pedido existe no banco de dados.
4. **Pós-condições:**
    a. O pedido é removido do sistema.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de excluir pedido.
    b. **Passo 2:** O atendente seleciona o pedido que deseja excluir.
    c. **Passo 3:** O sistema solicita confirmação para excluir o pedido.
    d. **Passo 4:** O atendente confirmar a exclusão.
    e. **Passo 5:** O sistema remove o pedido e exibe uma mensagem de sucesso.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Pedido Não Encontrado):**
        i. Se o pedido não for encontrado, o sistema exibe uma mensagem de erro.

Caso de Uso 7: Listar Pedido

1. **Ator:** Atendente
2. **Objetivo do Caso de Uso:**
    a. Permitir que o atendente visualize uma lista de pedidos registrados no sistema.
3. **Pré-condições:**
    a. O atendente está autenticado no sistema.
4. **Pós-condições:**
    a. O atendente visualiza a lista de pedidos registrados.
5. **Fluxo de Eventos (Cenário Principal):**
    a. **Passo 1:** O atendente acessa a tela de listar pedidos.
    b. **Passo 2:** O sistema exibe a lista de todos os pedidos registrados.
    c. **Passo 3:** O atendente pode filtrar ou buscar pedidos por data, status ou cliente.
6. **Fluxos Alternativos:**
    a. **Fluxo Alternativo 1 (Nenhum Pedido):**
        i. Se não houver pedidos registrados, o sistema exibe uma mensagem informando que não há pedidos no momento.

Diagrama de Classe

1. Classes principais:
    a. Cliente (id, nome, telefone, endereço, bairro)
    b. Pedido (id, cliente_id, sabor, tamanho, quantidade, status)
    c. Atendente (id, nome, login, senha)
    d. ItemPedido (id, quantidade, subtotal, pedido_id, sabor_id, tamanho_id)
    e. Sabor (id, nome)
    f. Tamanho (id, nome, quantidade, tamanho, preço)



Descrição do Diagrama de Sequência para "Registrar Pedido"

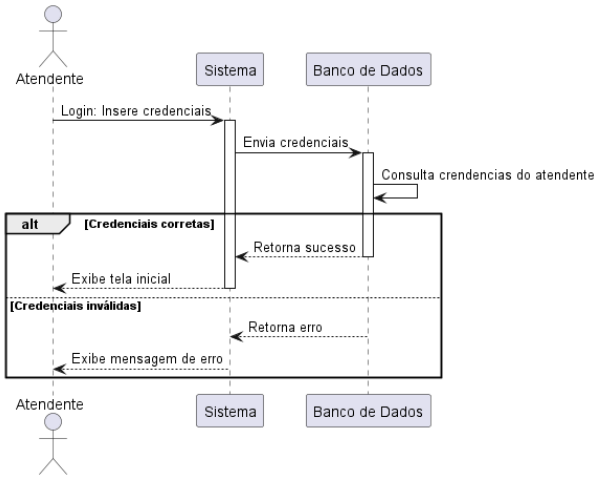
1. Atendente acessa a funcionalidade de registrar pedido.
2. O sistema solicita a seleção de um Cliente.
3. O atendente escolhe o cliente e adiciona "**ItensPedido**" ao "**Pedido**".
4. O sistema verifica a disponibilidade dos "**Produtos**" e calcula o "**valorTotal**".
5. O Pedido é registrado no banco de dados.
6. O sistema exibe uma confirmação ao Atendente

Mapeamento dos Códigos - Diagrama de Classe

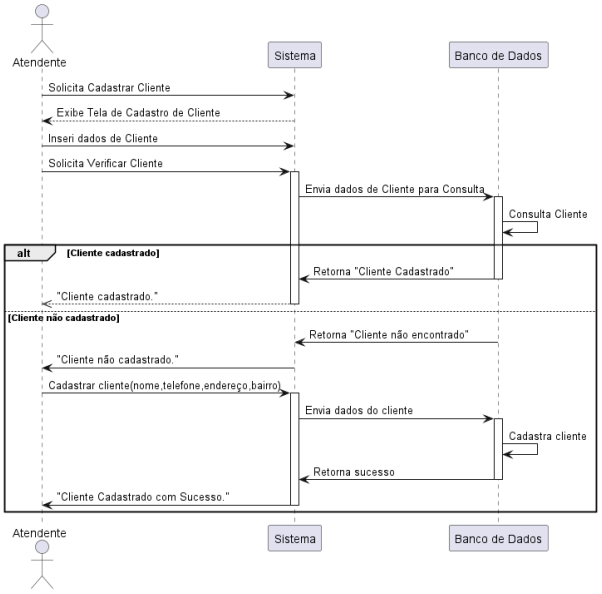
```
@startuml Digrama de Classe - Pizzaria

class Cliente {
    +nome: String
    +telefone: String
    +endereco: String
}

class Atendente {
    +nome: String
    +login(usuario: String, senha: String): boolean
    +registrarPedido(cliente: Cliente, pedido: Pedido)
    +cadastrarCliente(cliente: Cliente)
    +listarPedidos(): List<Pedido>
    +atualizarStatusPedido(pedido: Pedido, status: String)
}

class Pizza {
    +tamanho: String  // "Média", "Grande", "Gigante"
    +sabor: String
    +preco: float
}

class Pedido {
    +id: int
    +dataHora: DateTime
    +status: String
    +cliente: Cliente
    +pizzas: List<Pizza>
    +valorTotal: float
    +calcularTotal(): float
    +pagarPedido(): void
}

Cliente "1..1" -left→ "1..1" Atendente : comunica-se
Atendente "1..1" -- "*" Pedido : gerencia
Pedido "1..1" -- "*" Pizza : contém

@enduml
```

Diagrama de Sequência dos Casos de Uso

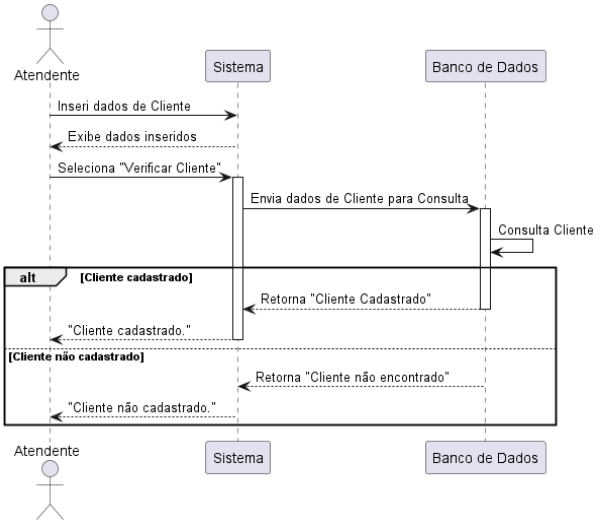
1. Login



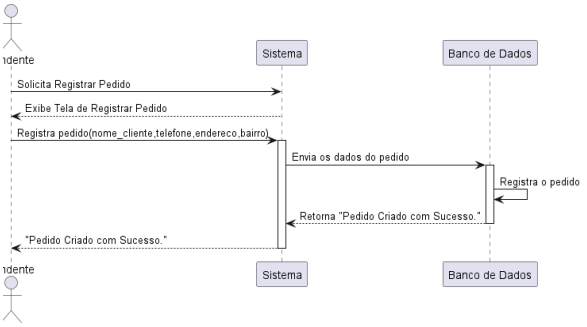
2. Cadastrar Cliente



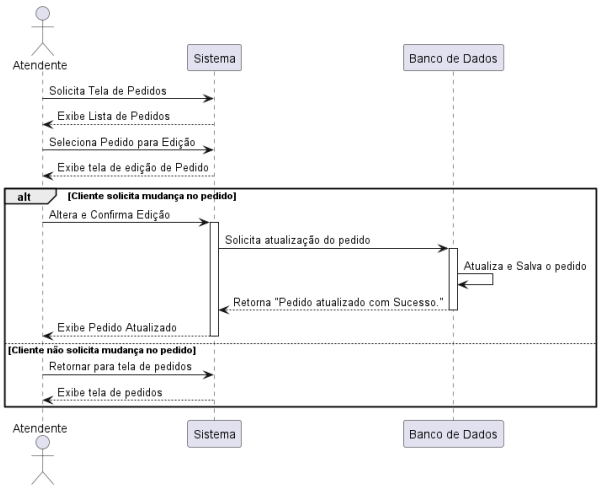
3. Verificar Cliente



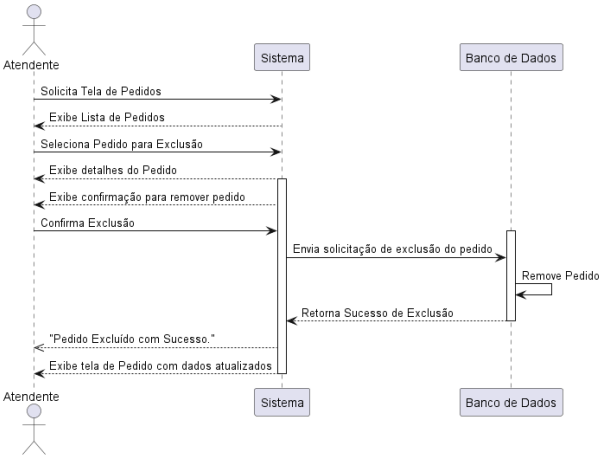
4. Registrar Pedido



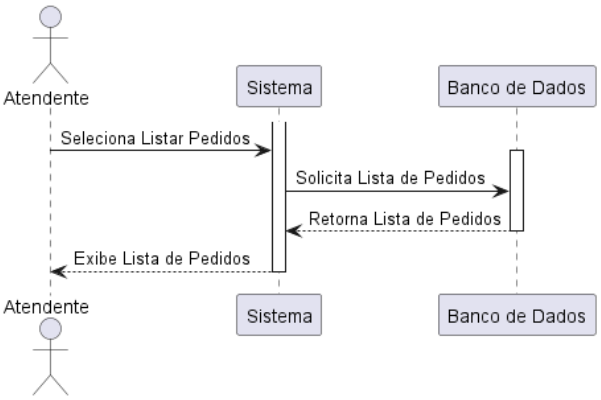
5. Editar Pedido



6. Excluir Pedido



7. Listar Pedido(histórico)



8. Pagar Pedido