TRABALHO DE PROGRAMAÇÃO WEB E DESENVOLVIMENTO DE APLICATIVO

Por: Alan Bianchi de Carvalho

CEFET RJ
Unidade Maria da Graça

Sumário

1 Sistema ParkQ 2

1.1 Descrição do MiniMundo 2

1.2 Requisitos Funcionais e Não-Funcionais: 3

1.3 Estrutura do Sistema 5

1.4 Funcionalidades Adicionais 8

2 Visão Geral 8

2.1 Classes, Entidades e Métodos do Sistema ParkQ 9

2.2 Diagrama de Classe ParkQ 11

2.3 Banco de Dados 12

2.4 Tabelas 12

2.5 Diagrama do Banco de Dados 22

1 Sistema ParkQ

O **ParkQ** é um sistema centralizado de gerenciamento de estacionamento rotativo, ideal para clientes como shoppings, hospitais e centros comerciais. Esse sistema permite o monitoramento eficiente das vagas, facilita reservas, controla o fluxo de entrada e saída e possibilita pagamentos diretos no aplicativo.

1.1 Descrição do MiniMundo

A proposta do **ParkQ** é desenvolver um sistema centralizado de gerenciamento de estacionamento rotativo que será disponibilizado em um aplicativo para diversos clientes, como shoppings, hospitais e centros comerciais. O objetivo principal deste produto é automatizar a gestão das filas de entrada e saída dos estacionamentos, monitorar as vagas disponíveis e facilitar as reservas para os usuários finais.

Cada estacionamento cliente possui um **código identificador único**, um **nome**, uma **localização** e uma **capacidade máxima de vagas**. O estacionamento é gerenciado por usuários que têm permissão para acessar e operar o sistema, sendo alguns desses usuários designados como **administradores**.

Cada cliente pode ter vários **usuários**, cada um com um **código**, um **login** e uma **senha**. Um usuário é identificado por seu código e pelo código do estacionamento ao qual está associado. Além disso, cada estacionamento pode ter pelo menos um usuário com perfil de **administrador**, que possui privilégios especiais para gerenciar as vagas e monitorar a ocupação.

Os **usuários** podem realizar várias **reservas** de vagas, e cada reserva é associada a um **usuário** específico e a uma **vaga** disponível. Cada reserva possui informações como **data de início**, **data de fim** e **valor calculado** para a estadia. Uma vaga é identificada por um **número único** e pode ter um **status** (disponível ou ocupada) e um **tipo** (carro ou moto).

As **reservas** devem estar associadas a um **plano de tarifação**, que define o valor da estadia com base na duração e no tipo de vaga ocupada. Cada plano de tarifação possui um **código único**, uma **data de vigência** e um **valor**, e pode ser compartilhado por várias reservas.

O sistema também registra **operações** realizadas pelos usuários para auditoria. Cada operação tem um **código único**, uma **descrição** e é acompanhada pela **data e hora** em que foi realizada. As operações podem incluir ações como reservas, cancelamentos e pagamentos.

Para a gestão das filas de entrada e saída, o sistema integra **sensores IoT** que monitoram o estado das vagas em tempo real, permitindo que os usuários vejam a disponibilidade e reservem vagas de forma eficiente. O sistema também permite que usuários compartilhem suas vagas com outros, em situações em que diferentes motoristas se revezam utilizando a mesma vaga.

Em resumo, o sistema **ParkQ** visa fornecer uma solução completa e automatizada para o gerenciamento de estacionamento, garantindo eficiência e controle detalhado para administradores e uma experiência fluida para os usuários.

1.2 Requisitos Funcionais e Não-Funcionais:

1.2.1 Requisitos Funcionais

Identificação	Descrição
Gestão de Estacionamento	Cadastro de estacionamento com atributos como código, nome, localização e capacidade máxima de vagas. Atualização do número de vagas disponíveis em tempo real (monitorarVagas()). Geração de relatórios de ocupação, faturamento e tempos médios de uso (gerarRelatorios()). Adição e remoção de vagas (adicionarVaga() e removerVaga()).
Gerenciamento de Vagas	Identificação das vagas com número único, status (disponível ou ocupada), tipo de veículo (carro ou moto) e indicação de reserva. Alteração do status da vaga para reservada ou disponível (reservar() e liberar()).
Cadastro e Autenticação de Usuários	Cadastro de usuários com informações de contato e credenciais para acesso (login e senha). Controle de histórico de reservas por usuário.
Reservas de Vagas	Realização de reservas (reservarVaga()) e cancelamento de reservas

	(cancelarReserva()). Associar reserva a uma vaga e a um usuário, com detalhes de data de início e fim, e valor calculado da estadia.
Pagamento de Estadia	Realização de pagamento da estadia no aplicativo através de métodos como cartão de crédito, PIX, entre outros (pagarEstadia()).
Gestão de Tarifação	Cálculo do valor da estadia com base na tarifa base, por hora ou diária, e tipo de vaga (calcularTarifa()). Cadastro de planos de tarifação com código único, descrição, data de vigência e taxas.
Monitoramento de Operações	Registro de operações como reservas, cancelamentos e pagamentos para auditoria, com código único, descrição e data/hora.
Monitoramento em Tempo Real com Sensores IoT	Integração de sensores para monitorar o status das vagas em tempo real e otimização do fluxo de veículos.
Compartilhamento de Vagas e Clubes de Vantagens	Permitir o compartilhamento de vagas entre usuários em turnos e criação de eventos ou clubes com descontos.
Notificações Automáticas	Envio de notificações sobre eventos, mudanças e fim de estadias para usuários.

1.2.2 Requisitos Não Funcionais

Identificação	**Descrição**
Segurança	Registro das operações para auditoria e segurança do sistema, com a possibilidade de revisão de ações como reservas e pagamentos.
Desempenho	O sistema deve processar reservas em até 5 segundos.
Usabilidade	O sistema deve ser intuitivo e fácil de usar, acessível para usuários com pouca experiência técnica.

1.3 Estrutura do Sistema

Estacionamento (Cliente)

- **Atributos**:
  - codigo: Identificador único do estacionamento.
  - nome: Nome do estacionamento.
  - localizacao: Localização física.
  - capacidade: Capacidade máxima de vagas.
- **Métodos**:
  - monitorarVagas(): Atualiza o número de vagas disponíveis em tempo real.
  - gerarRelatorios(): Gera relatórios de ocupação, faturamento e tempos médios de uso.
  - adicionarVaga(): Adiciona uma vaga ao estacionamento.
  - removerVaga(): Remove uma vaga do estacionamento.

Usuário (Visitante)

- **Atributos**:
  - codigo: Identificador único do usuário.
  - nome, email, telefone: Informações de contato.
  - login, senha: Credenciais para acesso ao sistema.
  - historicoReservas: Lista de reservas feitas pelo usuário.
- **Métodos**:
  - reservarVaga(): Reserva uma vaga.
  - cancelarReserva(): Cancela uma reserva ativa.

- pagarEstadia(): Realiza o pagamento da estadia.

Administrador (Usuário com Permissões)

- **Atributos**:
  - nome, email, telefone: Informações de contato.
- **Métodos**:
  - gerenciarVagas(): Administra status e disponibilidade das vagas.
  - monitorarOcupacao(): Monitora ocupação e disponibilidade em tempo real.
  - enviarNotificacoes(): Notifica eventos ou alterações no estacionamento.

Vaga

- **Atributos**:
  - numero: Identificador único da vaga.
  - status: Estado atual (Disponível/Ocupada).
  - tipo: Tipo de veículo compatível (Carro/Moto).
  - reservada: Indica se a vaga está reservada.
- **Métodos**:
  - reservar(): Altera o status da vaga para "Reservada".
  - liberar(): Altera o status para "Disponível".

Reserva

- **Atributos**:
  - dataReserva, dataFim: Datas de início e fim da reserva.
  - valor: Custo total da estadia.
  - usuario: Usuário que fez a reserva.
  - vaga: Vaga associada à reserva.

- **Métodos**:
  - calcularValor(): Calcula o valor da estadia com base no tempo reservado.
  - monitorarTempo(): Monitora o tempo restante para a reserva expirar.

Operação

- **Atributos**:
  - codigo: Identificador único da operação.
  - descricao: Descrição da operação.
  - dataHora: Data e hora de realização.
- **Utilização**:
  - Registrado para auditoria e segurança, possibilitando a revisão de ações como reservas e pagamentos.

PlanoDeTarifacao

- **Atributos**:
  - codigo: Identificador único do plano de tarifação.
  - descricao: Uma breve descrição do plano.
  - dataVigencia: Data de início de vigência do plano.
  - taxaBase: Valor base a ser aplicado por tipo de vaga (pode ser uma tabela ou mapa).
  - taxaHora: Taxa por hora de uso da vaga, ajustável por tipo de vaga (carro, moto).
  - taxaDiaria: Taxa diária para reservas de mais de 24 horas.

- **Métodos**:
  - calcularTarifa(tipoVaga, duracao): Calcula o valor total da estadia com base no tipo de vaga e na duração da reserva (em horas ou dias, dependendo do caso).

1.4 Funcionalidades Adicionais

- **Pagamentos e Tarifação**:
    - Associa cada reserva a um plano de tarifação, determinando valores conforme o tipo de vaga e duração. As transações podem ser realizadas via cartão de crédito, PIX, entre outros métodos.
- **Filas e Sensores IoT**:
    - Sensores monitoram o status das vagas em tempo real, integrados a um sistema de filas inteligentes, otimizando o fluxo de veículos conforme a demanda. Notificações são enviadas ao usuário próximo ao fim de sua estadia.
- **Clubes de Compartilhamento de Vagas**:
    - O sistema permite que usuários compartilhem vagas em turnos, e ainda possibilita a criação de eventos e clubes, onde os membros podem obter descontos e promoções exclusivas.

2 Visão Geral

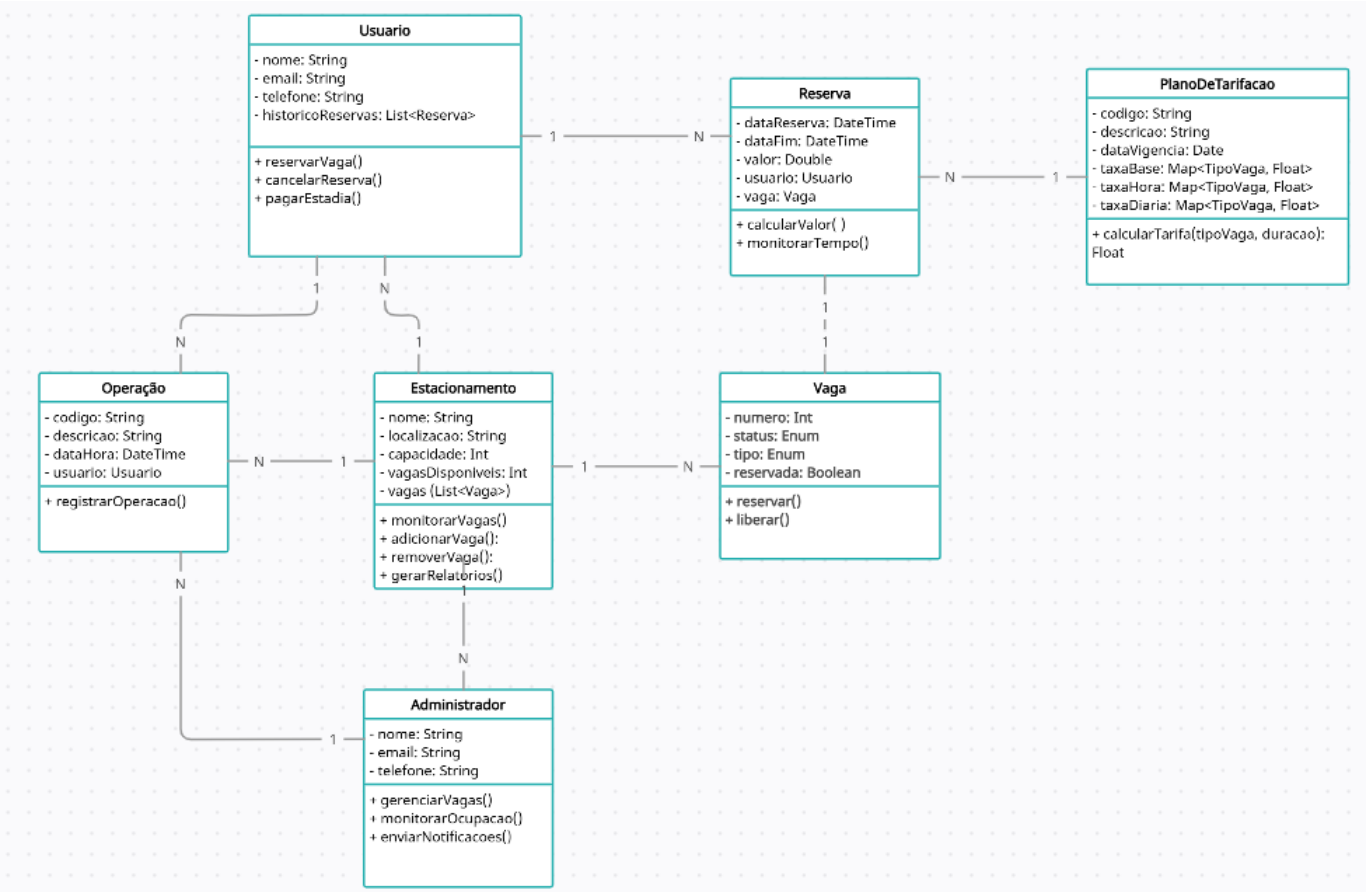
O ParkQ oferece uma solução completa e automatizada para o gerenciamento de estacionamentos, integrando controle detalhado e uma experiência otimizada para administradores e usuários. As funcionalidades visam à eficiência e controle para o administrador e à conveniência para o usuário final.

2.1 Classes, Entidades e Métodos do Sistema ParkQ

Classe	Entidades (Atributos)	Métodos
Estacionamento	- nome: String - localizacao: String - capacidade: Int - vagasDisponiveis: Int	- monitorarVagas(): Atualiza o número de vagas disponíveis em tempo real. - gerarRelatorios(): Gera relatórios sobre ocupação, faturamento e tempos médios de uso.
Vaga	- numero: Int - status: Enum (Disponível/Ocupada) - tipo: Enum (Carro/Moto) - reservada: Boolean	- reservar(): Altera o status da vaga para "Reservada". - liberar(): Altera o status da vaga para "Disponível".
Administrador	- CPF: String - nome: String - email: String - telefone: String	- gerenciarVagas(): Gerencia o status e disponibilidade das vagas. - monitorarOcupacao(): Monitora a ocupação e disponibilidade em tempo real. - enviarNotificacoes(): Envia notificações sobre eventos ou mudanças no estacionamento. - adicionarVaga(): Adiciona uma vaga ao estacionamento. - removerVaga(): Remove uma vaga do estacionamento.
Usuario	- CPF: String - nome: String - email: String - telefone: String	- reservarVaga(): Reserva uma vaga específica. - cancelarReserva(): Cancela uma reserva ativa.

	- historicoReservas: List<Reserva>	- pagarEstadia(): Realiza o pagamento da estadia (cartão de crédito, PIX, etc.).
PlanoDeTarifacao	codigo: Identificador único do plano de tarifação. descricao: Uma breve descrição do plano. dataVigencia: Data de início de vigência do plano. taxaBase: Valor base a ser aplicado por tipo de vaga (pode ser uma tabela ou mapa). taxaHora: Taxa por hora de uso da vaga, ajustável por tipo de vaga (carro, moto). taxaDiaria: Taxa diária para reservas de mais de 24 horas.	calcularTarifa(tipoVaga, duracao): Calcula o valor total da estadia com base no tipo de vaga e na duração da reserva (em horas ou dias, dependendo do caso).
Reserva	- dataReserva: DateTime - dataFim: DateTime - valor: Double - usuario: Usuario - vaga: Vaga	- calcularValor(): Calcula o valor total da estadia com base no tempo reservado. - monitorarTempo(): Monitora o tempo restante para a reserva expirar.
Operação	- codigo: Int - descricao: String - dataHora: DateTime - usuario: Usuario	- N/A

2.2 Diagrama de Classe ParkQ



1 - Diagrama de Classes do ParkQ

2.3 Banco de Dados

2.4 Tabelas

```
-- Tabela Estacionamento
CREATE TABLE Estacionamento (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    localizacao VARCHAR(255) NOT NULL,
    capacidade INT NOT NULL,
    vagas_disponiveis INT NOT NULL
);

-- Tabela Usuario
CREATE TABLE Usuario (
    CPF INT PRIMARY KEY UNIQUE NOT NULL,
    tipo_usuario ENUM('cliente', 'visitante', 'administrador') NOT NULL,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    login VARCHAR(50) UNIQUE NOT NULL,
    senha VARCHAR(100) NOT NULL,
    id_estacionamento INT,
    FOREIGN KEY (id_estacionamento) REFERENCES Estacionamento(id)
);
```

```
-- Tabela Vaga
CREATE TABLE Vaga (
    id INT PRIMARY KEY AUTO_INCREMENT,
    numero INT NOT NULL,
    id_estacionamento INT NOT NULL,
    status ENUM('disponivel', 'ocupada') NOT NULL,
    tipo ENUM('carro', 'moto') NOT NULL,
    reservada BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (id_estacionamento) REFERENCES Estacionamento(id)
);

-- Tabela Reserva
CREATE TABLE Reserva (
    id INT PRIMARY KEY AUTO_INCREMENT,
    data_reserva DATETIME NOT NULL,
    data_fim DATETIME,
    valor DECIMAL(10, 2) NOT NULL,
    id_usuario INT NOT NULL,
    id_vaga INT NOT NULL,
    id_plano INT, -- Plano de tarifação vinculado diretamente
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id),
    FOREIGN KEY (id_vaga) REFERENCES Vaga(id),
    FOREIGN KEY (id_plano) REFERENCES PlanoTarifacao(id)
```

```
);

-- Tabela PlanoTarifacao
CREATE TABLE PlanoTarifacao (
    id INT PRIMARY KEY AUTO_INCREMENT,
    descricao VARCHAR(255),
    data_vigencia DATE NOT NULL,
    taxa_base DECIMAL(10, 2) NOT NULL,
    taxa_hora DECIMAL(10, 2),
    taxa_diaria DECIMAL(10, 2)
);

-- Tabela Operacao
CREATE TABLE Operacao (
    id INT PRIMARY KEY AUTO_INCREMENT,
    descricao VARCHAR(255) NOT NULL,
    data_hora DATETIME NOT NULL,
    id_usuario INT NOT NULL,
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id)
);
-- Tabela Pagamento
CREATE TABLE Pagamento (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_reserva INT NOT NULL,
```

```
    metodo_pagamento ENUM('cartao_credito', 'PIX', 'boleto') NOT NULL,
    valor_pago DECIMAL(10, 2) NOT NULL,
    data_hora DATETIME NOT NULL,
    FOREIGN KEY (id_reserva) REFERENCES Reserva(id)
);
```

Triggers

```
-- Trigger para atualizar vagas disponíveis automaticamente
DELIMITER //

CREATE TRIGGER trg_atualizar_vagas_disponiveis
AFTER UPDATE ON Vaga
FOR EACH ROW
BEGIN
    IF NEW.status = 'disponivel' AND OLD.status = 'ocupada' THEN
        UPDATE Estacionamento SET vagas_disponiveis = vagas_disponiveis + 1
        WHERE id = NEW.id_estacionamento;
    ELSEIF NEW.status = 'ocupada' AND OLD.status = 'disponivel' THEN
        UPDATE Estacionamento SET vagas_disponiveis = vagas_disponiveis - 1
        WHERE id = NEW.id_estacionamento;
    END IF;
END;
//
DELIMITER ;
```

2.5 Banco 2:

```
-- Tabela Estacionamento
CREATE TABLE Estacionamento (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    localizacao VARCHAR(255) NOT NULL,
    capacidade INT NOT NULL,
    vagas_disponiveis INT NOT NULL
);

-- Tabela PlanoTarifacao (precisa ser criada antes de Reserva)
CREATE TABLE PlanoTarifacao (
    id SERIAL PRIMARY KEY,
    descricao VARCHAR(255),
    data_vigencia DATE NOT NULL,
    taxa_base DECIMAL(10, 2) NOT NULL,
    taxa_hora DECIMAL(10, 2),
    taxa_diaria DECIMAL(10, 2)
);

-- Tabela Usuario (dados gerais)
CREATE TABLE Usuario (
    id SERIAL PRIMARY KEY, -- Identificador único do usuário
```

```
    CPF BIGINT UNIQUE NOT NULL, -- ALTERADO para BIGINT para comportar números grandes
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    telefone VARCHAR(20),
    login VARCHAR(50) UNIQUE NOT NULL,
    senha VARCHAR(100) NOT NULL,
    tipo_usuario VARCHAR(20) CHECK (tipo_usuario IN ('cliente', 'administrador', 'visitante')) NOT NULL, -- Alterado o ENUM para um campo VARCHAR com CHECK
    id_estacionamento INT,
    FOREIGN KEY (id_estacionamento) REFERENCES Estacionamento(id)
);

-- Tabela Vaga
CREATE TABLE Vaga (
    id SERIAL PRIMARY KEY,
    numero INT NOT NULL,
    id_estacionamento INT NOT NULL,
    status VARCHAR(20) CHECK (status IN ('disponivel', 'ocupada')) NOT NULL, -- Usando VARCHAR e CHECK no lugar de ENUM
    tipo VARCHAR(20) CHECK (tipo IN ('carro', 'moto')) NOT NULL,
    reservada BOOLEAN DEFAULT FALSE,
    FOREIGN KEY (id_estacionamento) REFERENCES Estacionamento(id)
);
```

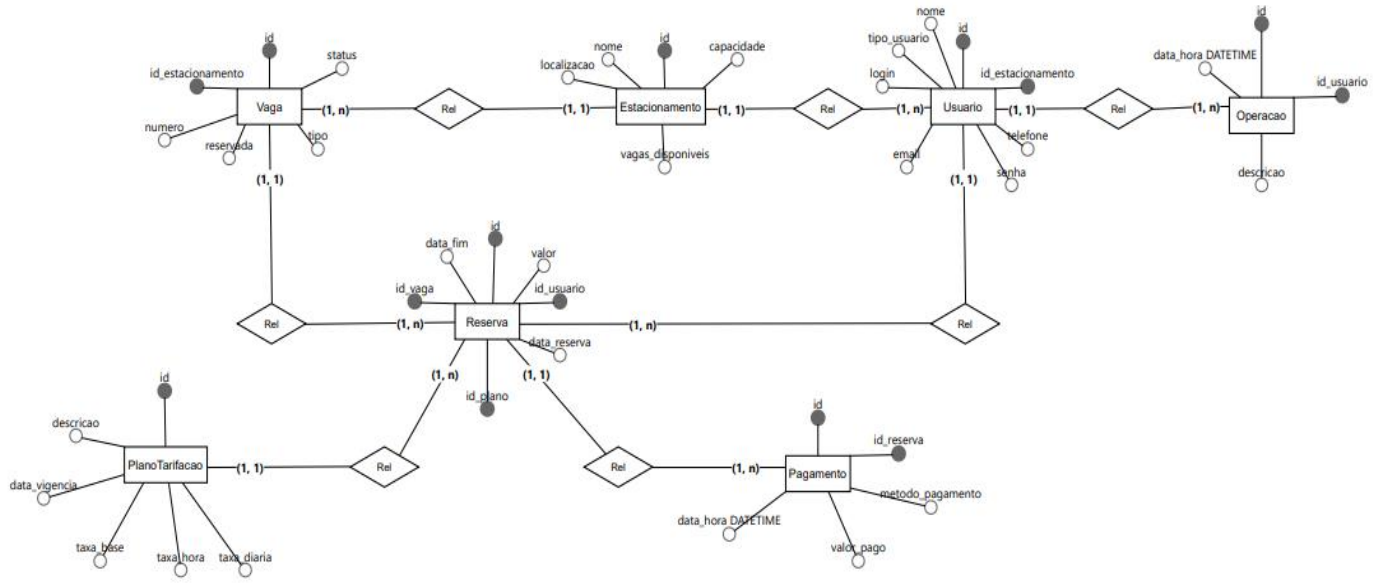
```
-- Tabela Reserva (relacionando com PlanoTarifacao)
CREATE TABLE Reserva (
    id SERIAL PRIMARY KEY,
    data_reserva TIMESTAMP NOT NULL,
    data_fim TIMESTAMP,
    valor DECIMAL(10, 2) NOT NULL,
    id_usuario INT NOT NULL, -- Relacionamento com a tabela Usuario (não com Cliente ou Administrador diretamente)
    id_vaga INT NOT NULL,
    id_plano INT, -- Plano de tarifação vinculado diretamente
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id),
    FOREIGN KEY (id_vaga) REFERENCES Vaga(id),
    FOREIGN KEY (id_plano) REFERENCES PlanoTarifacao(id) -- Relacionamento com a Tabela PlanoTarifacao
);

-- Tabela Operacao
CREATE TABLE Operacao (
    id SERIAL PRIMARY KEY,
    descricao VARCHAR(255) NOT NULL,
    data_hora TIMESTAMP NOT NULL,
    id_usuario INT NOT NULL, -- Relacionamento com a tabela Usuario
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id)
);
```

```
-- Tabela Pagamento
CREATE TABLE Pagamento (
    id SERIAL PRIMARY KEY,
    id_reserva INT NOT NULL,
    metodo_pagamento VARCHAR(20) CHECK (metodo_pagamento IN
('cartao_credito', 'PIX', 'boleto')) NOT NULL,
    valor_pago DECIMAL(10, 2) NOT NULL,
    data_hora TIMESTAMP NOT NULL,
    FOREIGN KEY (id_reserva) REFERENCES Reserva(id)
);

-- Tabela Cliente (relacionada com Usuario)
CREATE TABLE Cliente (
    id SERIAL PRIMARY KEY, -- Identificador único do cliente
    id_usuario INT NOT NULL, -- Chave estrangeira para Usuario
    data_registro TIMESTAMP NOT NULL, -- Data de registro do cliente
    preferencias TEXT, -- Preferências do cliente
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id)
);

-- Tabela Administrador (relacionada com Usuario)
CREATE TABLE Administrador (
    id SERIAL PRIMARY KEY, -- Identificador único do administrador
    id_usuario INT NOT NULL, -- Chave estrangeira para Usuario
    cargo VARCHAR(100), -- Cargo do administrador
```

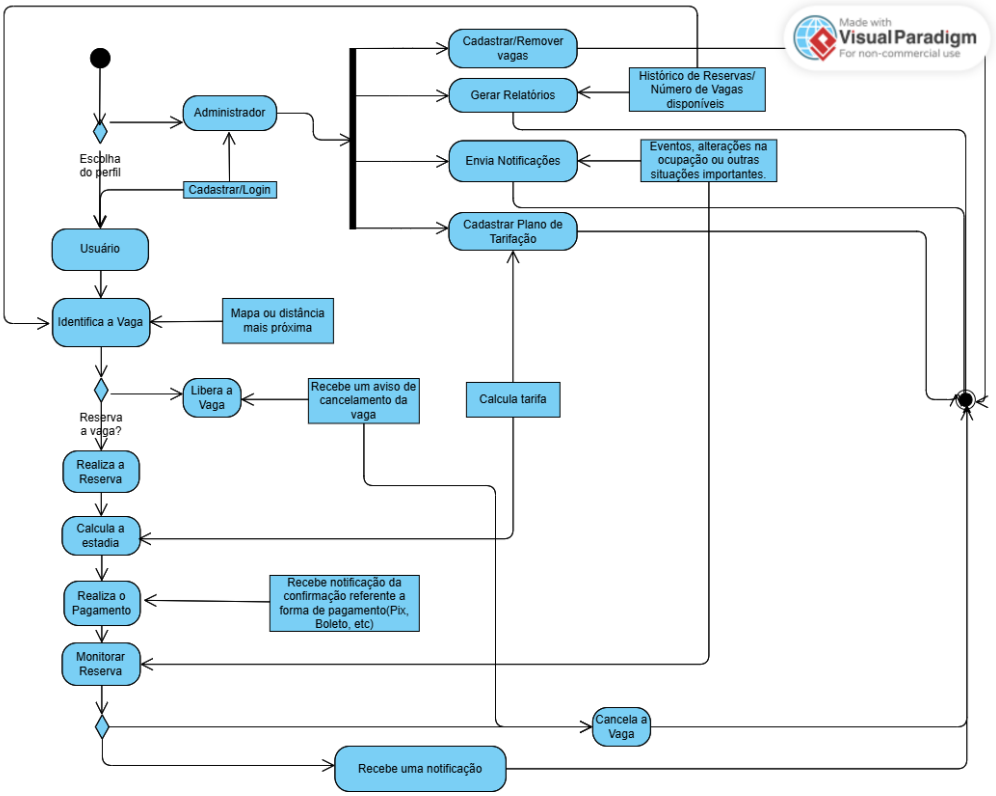
```
    privilegios TEXT, -- Privilegios especiais
    FOREIGN KEY (id_usuario) REFERENCES Usuario(id)
);

-- Função para atualizar vagas disponíveis automaticamente
CREATE OR REPLACE FUNCTION atualizar_vagas_disponiveis()
RETURNS TRIGGER AS
$$
BEGIN
    IF NEW.status = 'disponivel' AND OLD.status = 'ocupada' THEN
        UPDATE Estacionamento SET vagas_disponiveis = vagas_disponiveis + 1
        WHERE id = NEW.id_estacionamento;
    ELSIF NEW.status = 'ocupada' AND OLD.status = 'disponivel' THEN
        UPDATE Estacionamento SET vagas_disponiveis = vagas_disponiveis - 1
        WHERE id = NEW.id_estacionamento;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

-- Criar o Trigger em si
CREATE TRIGGER trg_atualizar_vagas_disponiveis
AFTER UPDATE ON Vaga
```

```
FOR EACH ROW

EXECUTE FUNCTION atualizar_vagas_disponiveis();
```

2.6 Diagrama do Banco de Dados

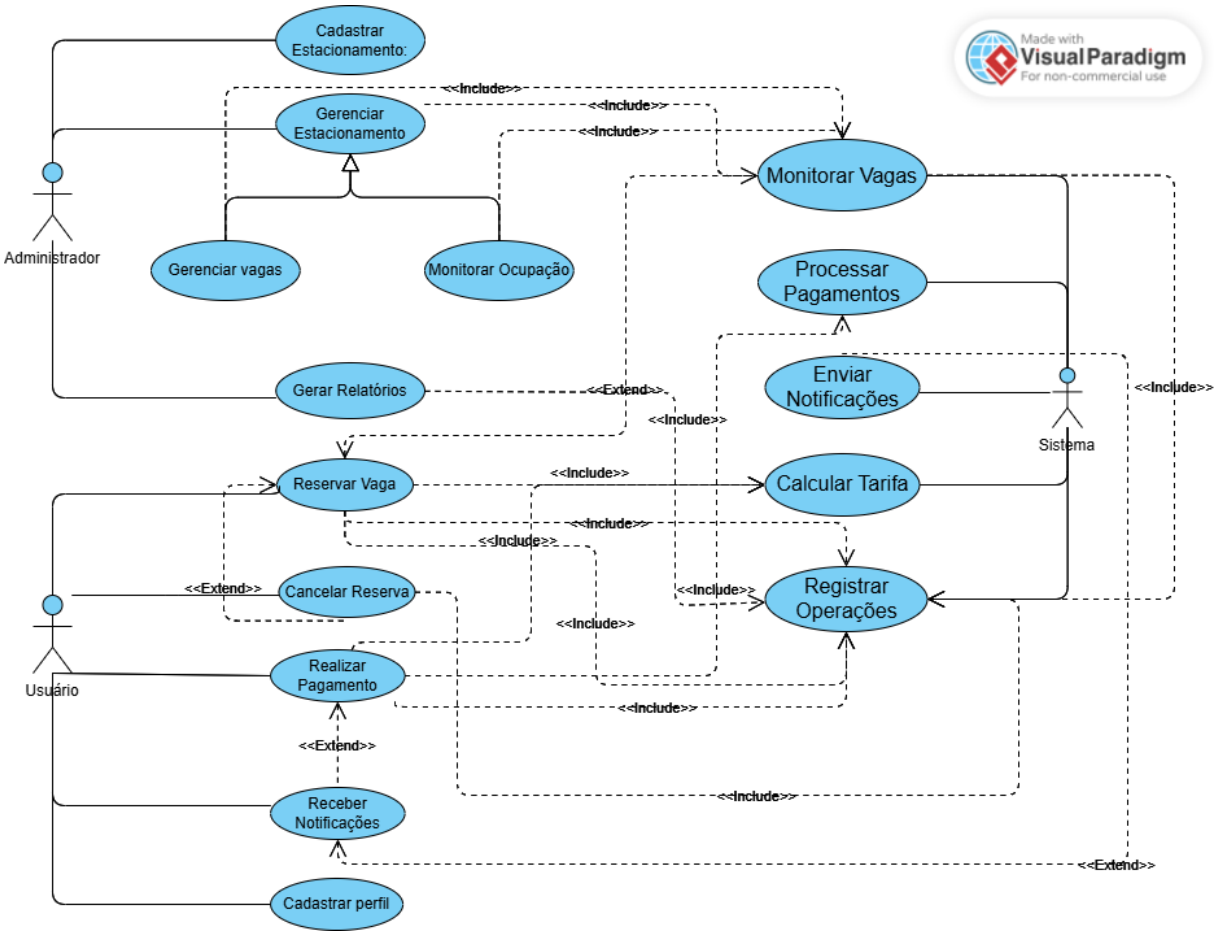


2.7 Diagrama de Atividade



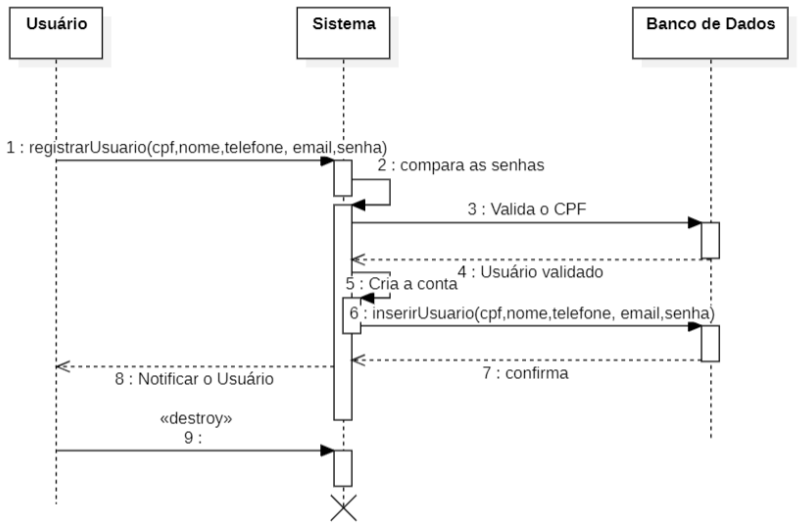
Caso de Uso	Relação	Outro Caso de Uso	Descrição
Reservar Vaga	<<include>>	Calcular Tarifa	Calcula o valor antes de reservar.
Reservar Vaga	<<extend>>	Cancelar Reserva	Cancelar depende de uma reserva ativa.
Reservar Vaga	<<include>>	Registrar Operações	Reserva é registrada para auditoria.
Cancelar Reserva	<<extend>>	Reservar Vaga	Ocorre apenas se houver reserva.
Cancelar Reserva	<<include>>	Registrar Operações	Cancelamento é registrado.
Realizar Pagamento	<<include>>	Calcular Tarifa	Calcula valor devido.
Realizar Pagamento	<<extend>>	Receber Notificações	Notificação é opcional após pagamento.
Realizar Pagamento	<<include>>	Registrar Operações	Pagamento é registrado.
Monitorar Vagas	<<include>>	Sistema de Sensores	Depende de dados dos sensores.
Monitorar Vagas	<<include>>	Registrar Operações	Monitoramento é registrado.
Gerenciar Vagas	<<include>>	Monitorar Vagas	Gestão depende de dados dos sensores.
Gerenciar Vagas	<<include>>	Registrar Operações	Gestão é registrada.
Gerenciar Estacionamento	<<include>>	Registrar Operações	Gestão é registrada.
Gerenciar Estacionamento	<<include>>	Gerar Relatórios	Gera relatórios para análise.
Gerenciar Estacionamento	Generalização	Gerenciar Vagas, Monitorar Ocupação	Casos especializados da gestão geral.
Gerar Relatórios	<<include>>	Registrar Operações	Usa dados registrados para gerar relatórios.
Gerar Relatórios	<<include>>	Calcular Tarifa	Inclui cálculo financeiro em relatórios.
Calcular Tarifa	<<include>>	Diversos (mencionados acima)	Base para calcular valores em diversos casos de uso.
Registrar Operações	<<include>>	Diversos (mencionados acima)	Base para registrar todas as ações.

2.8 Diagrama de Casos de Uso

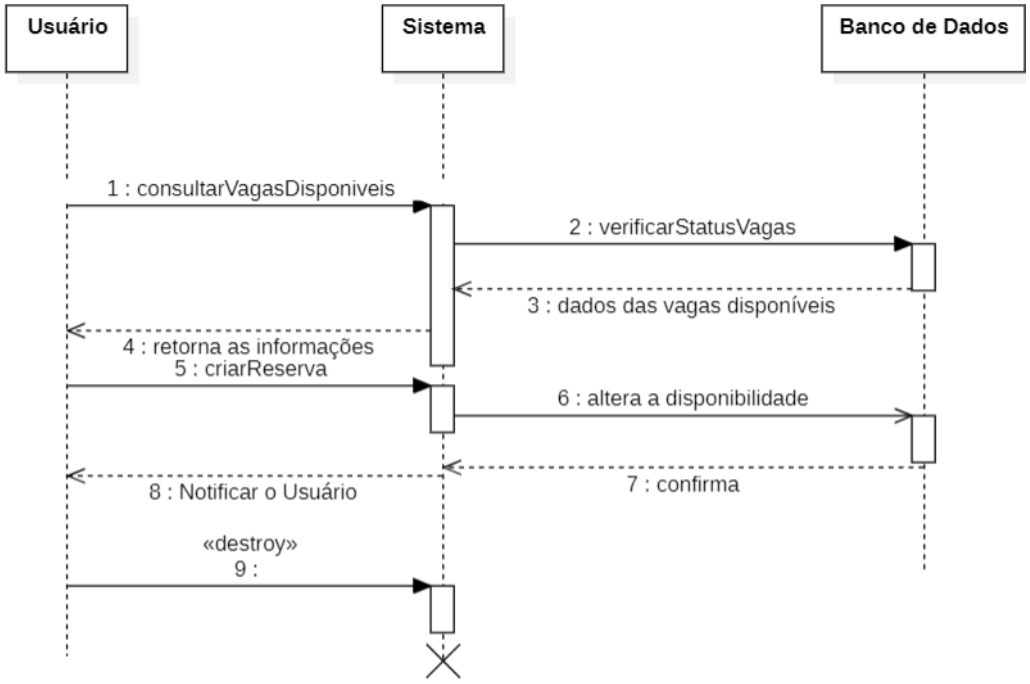


2.9 Diagrama de Sequência

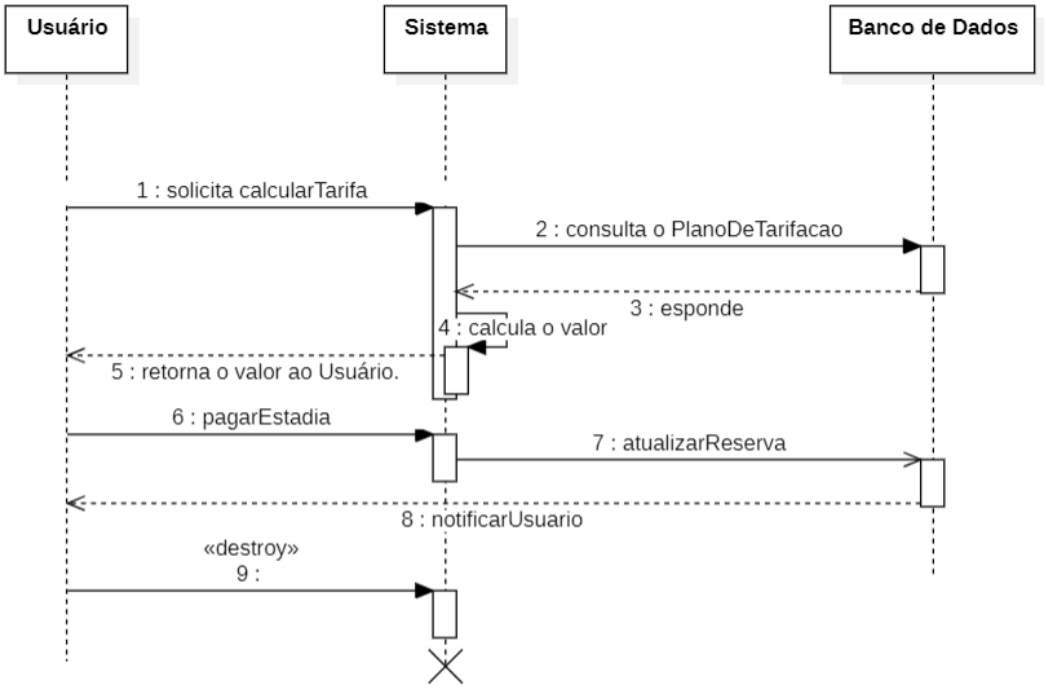
2.9.1 Cadastro de Usuário



2.9.2 Reserva de Vaga Pelo Usuário

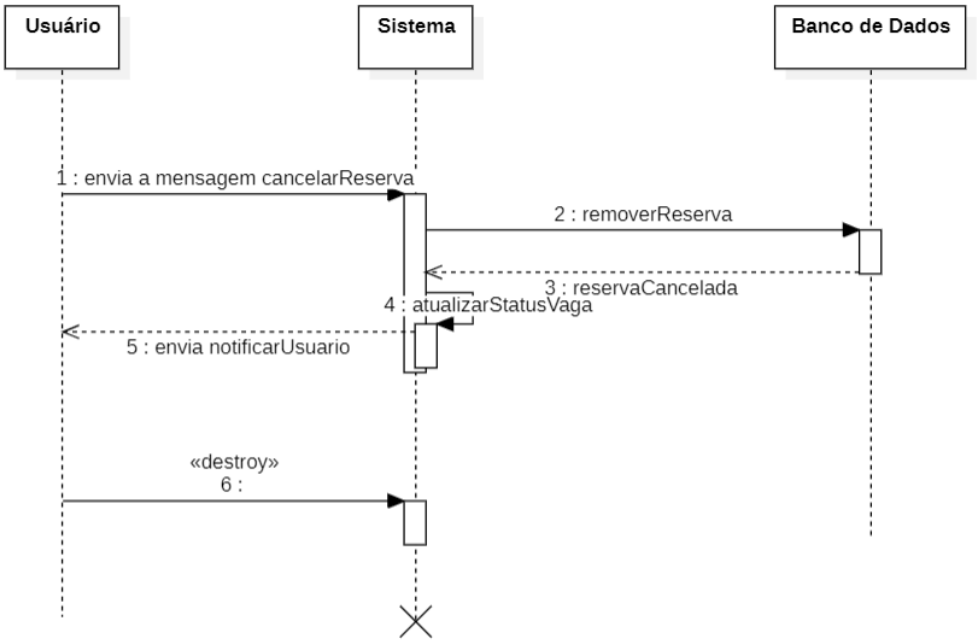


2.9.3 Cálculo e Pagamento da Estadia

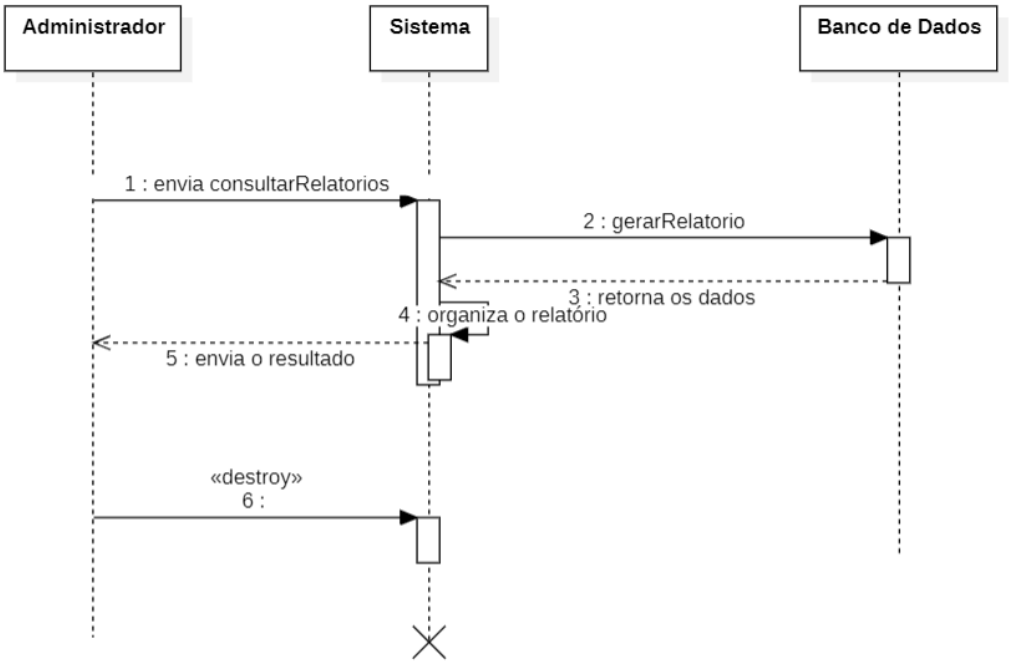


Cancelament

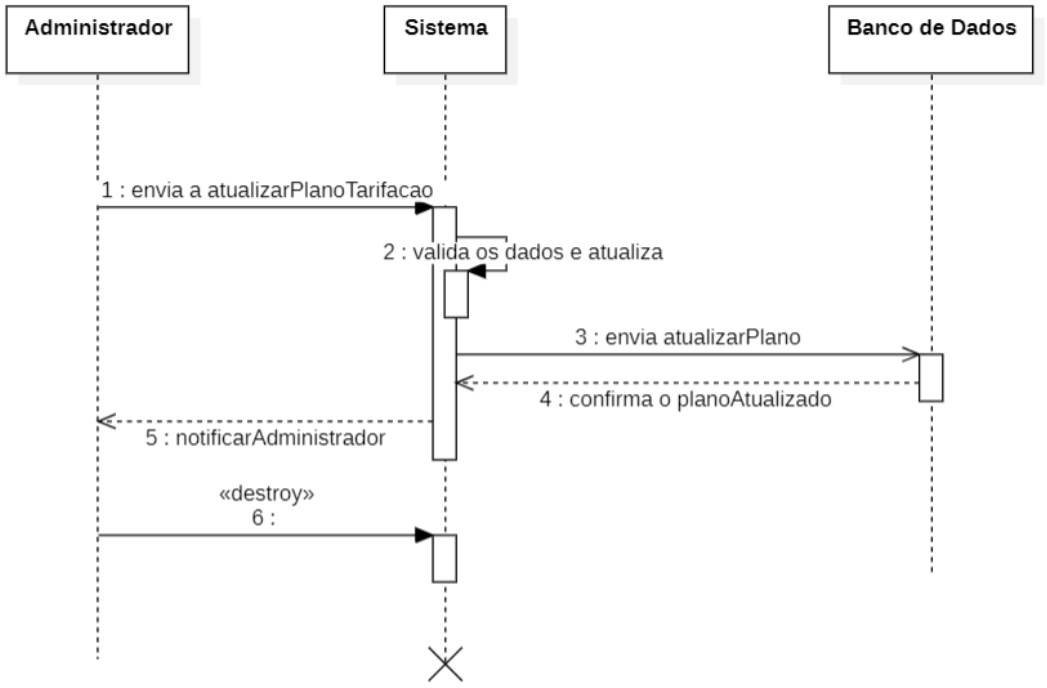
2.9.4 Cancelamento de Reserva Pelo Usuário



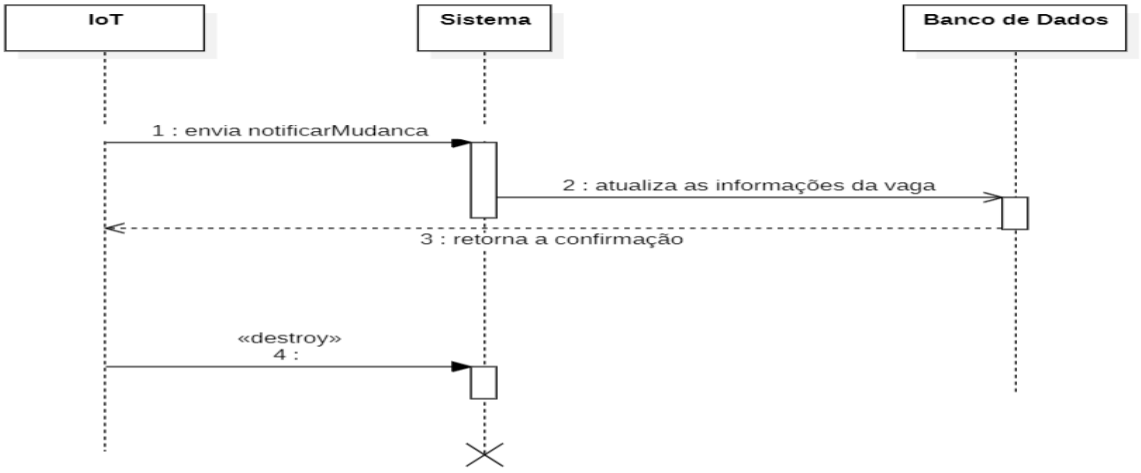
2.9.5 Consulta de Relatórios (Administrador)



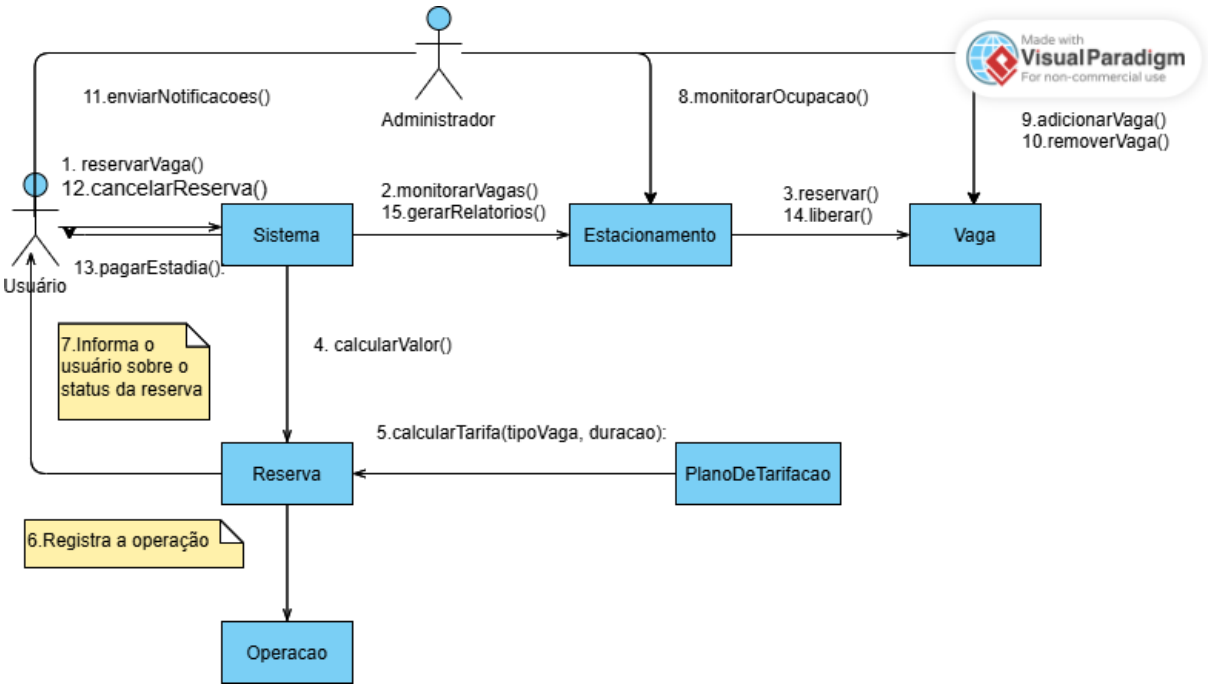
2.9.6 Gerenciamento de Tarifas



2.9.7 Monitoramento do Estacionamento (IoT)



2.10 Diagrama de Comunicação



parkq/

|

├── src/

| ├── common/

| | └── models/

| | ├── estacionamento.model.ts

| | ├── usuario.model.ts

| | ├── vaga.model.ts

```
|   |       ├── reserva.model.ts
|   |       ├── plano-tarifacao.model.ts
|   |       ├── operacao.model.ts
|   |       └── pagamento.model.ts
|   |
|   ├── estacionamentos/
|   |   ├── controllers/
|   |   |   └── estacionamento.controller.ts
|   |   ├── services/
|   |   |   └── estacionamento.service.ts
|   |   ├── repositories/
|   |   |   └── estacionamento.repository.ts
|   |   └── estacionamento.module.ts
|   |
|   ├── usuarios/
|   |   ├── controllers/
|   |   |   └── usuario.controller.ts
|   |   ├── services/
|   |   |   └── usuario.service.ts
|   |   ├── repositories/
|   |   |   └── usuario.repository.ts
|   |   └── usuario.module.ts
|   |
|   ├── vagas/
|   |   ├── controllers/
|   |   |   └── vaga.controller.ts
|   |   ├── services/
|   |   |   └── vaga.service.ts
|   |   ├── repositories/
|   |   |   └── vaga.repository.ts
```

```
|   |   └── vaga.module.ts
|   |
|   ├── reservas/
|   |   ├── controllers/
|   |   |   └── reserva.controller.ts
|   |   ├── services/
|   |   |   └── reserva.service.ts
|   |   ├── repositories/
|   |   |   └── reserva.repository.ts
|   |   └── reserva.module.ts
|   |
|   ├── planos-tarifacao/
|   |   ├── controllers/
|   |   |   └── plano-tarifacao.controller.ts
|   |   ├── services/
|   |   |   └── plano-tarifacao.service.ts
|   |   ├── repositories/
|   |   |   └── plano-tarifacao.repository.ts
|   |   └── plano-tarifacao.module.ts
|   |
|   ├── operacoes/
|   |   ├── controllers/
|   |   |   └── operacao.controller.ts
|   |   ├── services/
|   |   |   └── operacao.service.ts
|   |   ├── repositories/
|   |   |   └── operacao.repository.ts
|   |   └── operacao.module.ts
|   |
|   ├── pagamentos/
```

```
|   |   ├── controllers/
|   |   |   └── pagamento.controller.ts
|   |   ├── services/
|   |   |   └── pagamento.service.ts
|   |   ├── repositories/
|   |   |   └── pagamento.repository.ts
|   |   └── pagamento.module.ts
|   |
|   ├── app.module.ts
|   └── main.ts
|
├── .env
├── package.json
└── tsconfig.json
```