**Trabalho de PROGRAMAÇÃO WEB e DESENVOLVIMENTO DE APLICATIVO**

Por: Alan Bianchi de Carvalho

CEFET RJ

Unidade Maria da Graça

Sumário

[1 Sistema ParkQ 2](#_Toc181997383)

[**1.1** **Descrição do MiniMundo** 2](#_Toc181997384)

[**1.2** **Requisitos Funcionais e Não-Funcionais:** 3](#_Toc181997385)

[**1.3** **Estrutura do Sistema** 5](#_Toc181997386)

[**1.4** **Funcionalidades Adicionais** 8](#_Toc181997387)

[2 Visão Geral 8](#_Toc181997388)

[**2.1** **Classes, Entidades e Métodos do Sistema ParkQ** 9](#_Toc181997389)

[**2.2** **Diagrama de Classe ParkQ** 11](#_Toc181997390)

[**2.3** **Banco de Dados** 12](#_Toc181997391)

[**2.4** **Tabelas** 12](#_Toc181997392)

[**2.5** **Diagrama do Banco de Dados** 16](#_Toc181997393)

# **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.

## **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.

## **Requisitos Funcionais e Não-Funcionais:**

### 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. |

### 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. |

## **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).

## **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.

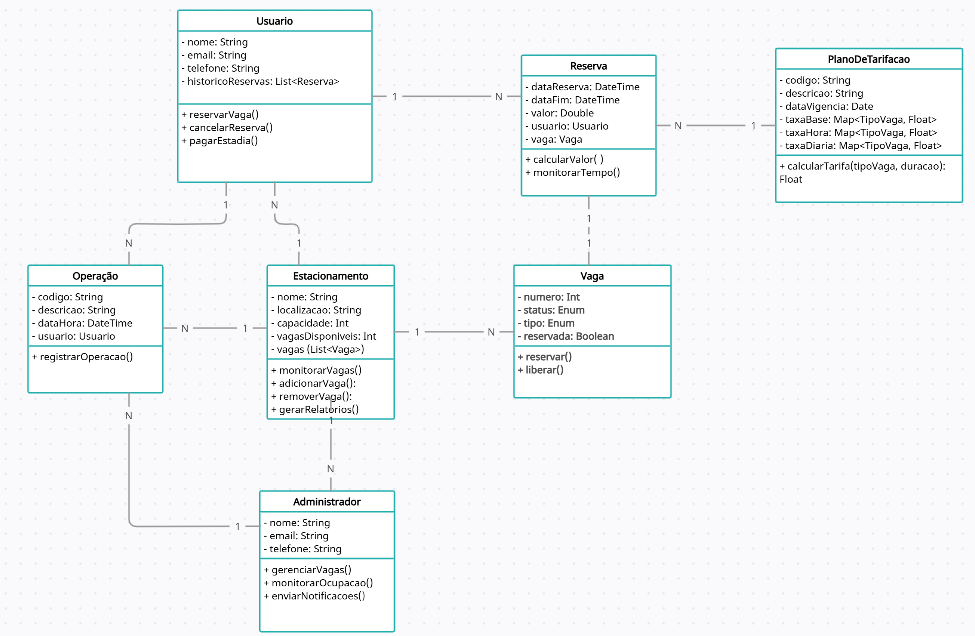
# **Visão Geral**

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.

## **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 - historicoReservas: List<Reserva> | - reservarVaga(): Reserva uma vaga específica. - cancelarReserva(): Cancela uma reserva ativa. - 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 |

## **Diagrama de Classe ParkQ**



1 - Diagrama de Classes do ParkQ

## **Banco de Dados**

## **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 ;

## **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();

## Mapa de cidade Descrição gerada automaticamente**Diagrama do Banco de Dados**

## **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. |

## **Diagrama de Casos de Uso**

Diagrama

Descrição gerada automaticamente

## **Diagrama de Sequência**

### Cadastro de Usuário

Diagrama

Descrição gerada automaticamente

### **Reserva de Vaga Pelo Usuário**

Diagrama

Descrição gerada automaticamente

### **Cálculo e Pagamento da Estadia**

Diagrama

Descrição gerada automaticamente

Cancelamento de Reserva

### **Cancelamento de Reserva Pelo Usuário**

Diagrama

Descrição gerada automaticamente

### **Consulta de Relatórios (Administrador)**

Diagrama

Descrição gerada automaticamente

### **Gerenciamento de Tarifas**

Diagrama

Descrição gerada automaticamente

### Diagrama Descrição gerada automaticamente**Monitoramento do Estacionamento (IoT)**

## **Diagrama de Comunicação**

Diagrama

Descrição gerada automaticamente

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