# *SpotEasy –* Sistema de controle de estacionamento

# Descrição do sistema

O Sistema de Gerenciamento de Estacionamento foi desenvolvido para facilitar a administração de estacionamentos, permitindo o registro de veículos, controle de entradas e saídas, cálculo de tarifas e geração de relatórios. O sistema é implementado em JavaScript e segue uma arquitetura modular para fácil manutenção e escalabilidade.

# Público-alvo

Este sistema foi projetado para ser utilizado por operadores de estacionamento, gerentes e administradores que precisam monitorar e controlar o fluxo de veículos dentro de um estacionamento.

# Objetivo:

O sistema de gerenciamento de estacionamento tem como objetivo registrar a entrada e saída de veículos, controlando o tempo de permanência e calculando a tarifa de acordo com as regras definidas. Além disso, o sistema deve fornecer informações sobre a ocupação das vagas em tempo real e histórico.

# Funcionalidades:

## Entrada de veículos

O sistema deve permitir o registrar a entrada de Veículos:

O sistema deve permitir registrar a placa do veículo;

O sistema deve permitir identificar o tipo de veículo (carro, moto, caminhão, etc.);

O sistema deve permitir registrar a data e hora da entrada;

O sistema deve permitir anotar informações adicionais, se necessário (modelo, cor, etc.);

O sistema deve permitir emitir um tíquete de entrada com código único;

## Saída de Veículos:

O sistema deve permitir registrar a placa do veículo na saída;

O sistema deve permitir validar o tíquete de entrada;

O sistema deve permitir calcular a tarifa de acordo com o tempo de permanência e tipo de veículo;

O sistema deve permitir registrar a data e hora da saída;

O sistema deve permitir, permitir a saída do veículo mediante pagamento da tarifa;

O sistema deve permitir registrar o pagamento (dinheiro, cartão, etc.);

O sistema deve permitir emitir comprovante de pagamento.

## Controle de Vagas:

O sistema deve permitir monitorar a ocupação das vagas em tempo real;

O sistema deve permitir indicar vagas disponíveis e ocupadas;

O sistema deve permitir registrar histórico de ocupação das vagas.

## Relatórios e Estatísticas:

O sistema deve permitir gerar relatórios de entradas e saídas de veículos;

O sistema deve permitir apresentar dados sobre a arrecadação;

O sistema deve permitir indicar o índice de ocupação das vagas;

O sistema deve permitir identificar horários de pico de movimento;

## Regras de Negócio:

### Tarifas:

O sistema deve permitir definir diferentes tarifas por tipo de veículo (carro, moto, caminhão, etc.);

O sistema deve permitir estabelecer tarifas diferenciadas por período (horário normal, noturno, feriados, etc.);

### Formas de Pagamento:

O sistema deve permitir aceitar dinheiro, cartão de débito e, cartão de crédito e pix;

O sistema deve permitir integrar com sistemas de pagamento eletrônico.

## Segurança:

O sistema deve permitir controlar o acesso ao sistema de gerenciamento;

O sistema deve permitir rlegistrar todas as operações realizadas no sistema.

# Usabilidade

## Classes Principais

Veiculo

Descrição: Representa um veículo.

Propriedades:

placa: Placa do veículo.

modelo: Modelo do veículo.

cor: Cor do veículo.

tipo: Tipo de veículo (moto, carro, micro-ônibus).

Checkin

Descrição: Representa o check-in de um veículo no estacionamento.

Propriedades:

veiculo: Instância da classe Veiculo.

vaga: Identificação da vaga ocupada.

dataEntrada: Data de entrada do veículo.

horaEntrada: Hora de entrada do veículo.

Checkout

Descrição: Extende a classe Checkin e adiciona funcionalidades de check-out.

Propriedades:

dataSaida: Data de saída do veículo.

horaSaida: Hora de saída do veículo.

tarifa: Valor calculado pela permanência.

Métodos:

calcularPermanencia(): Calcula o tempo de permanência do veículo em horas.

calcularTarifa(valorPorHora): Calcula a tarifa com base no tempo de permanência e valor por hora.

Vaga

Descrição: Representa uma vaga no estacionamento.

Propriedades:

id: Identificação da vaga.

tipo: Tipo de veículo que a vaga suporta.

disp: Estado da vaga (livre ou ocupada).

Métodos:

ocupar(): Define a vaga como ocupada.

liberar(): Define a vaga como livre.

Funções e Interações

criarMapa()

Descrição: Cria o mapa de vagas baseado na quantidade de vagas para motos, carros

e micro-ônibus definidas pelo usuário. As vagas são armazenadas na variável global vagas.

Interações:

Atualiza o mapa visual e o dashboard com as informações das vagas.

Salva os dados das vagas no localStorage.

salvarDados() e carregarDados()

Descrição: salvarDados() salva o estado atual das vagas no localStorage e carregarDados() restaura esses dados quando a página é recarregada.

atualizarDashboard()

Descrição: Atualiza as informações de ocupação das vagas no dashboard, incluindo o total de vagas, vagas ocupadas, vagas livres e a taxa de ocupação.

encontrarVagaDisponivel(tipoVeiculo)

Descrição: Busca uma vaga disponível do tipo especificado.

Retorno: Retorna a primeira vaga livre encontrada para o tipo de veículo.

salvarTabela() e carregarTabela()

Descrição: salvarTabela() salva a tabela de veículos no localStorage e carregarTabela() restaura essa tabela ao recarregar a página.

limparDados()

Descrição: Limpa os dados de vagas e a tabela de veículos do localStorage, além de resetar o mapa e o dashboard.

realizarCheckout(botao)

Descrição: Realiza o processo de check-out de um veículo, liberando a vaga, removendo o veículo da tabela de veículos e atualizando o dashboard. Adiciona a informação do check-out à tabela de histórico.

adicionarAoHistorico(checkout)

Descrição: Adiciona uma linha à tabela de histórico com as informações de check-out, incluindo permanência e tarifa.

enviarDadosParaSheetDB()

Descrição: Envia os dados da tabela de histórico para uma API externa, como o SheetDB.

Funcionalidade de Persistência

As informações de vagas, tabela de veículos e histórico são persistidas usando localStorage, garantindo que os dados permaneçam disponíveis após o recarregamento da página.