Professor Gustavo Typescript 17/10/2024

**Criando um Sistema de Gerenciamento de Estacionamento em TypeScript com Classes**

**Entendendo o Problema:**

Para criar um sistema de gerenciamento de estacionamento em TypeScript, precisamos modelar as entidades envolvidas (veículos, clientes, estacionamentos) e as ações que podem ser realizadas (registro de entrada e saída, cobrança, emissão de tickets, geração de relatórios).

**Estrutura de Classes:**

1. **Veículo:**
   * placa: string
   * modelo: string
   * cor: string
2. **Cliente:**
   * nome: string
   * cpf: string
   * telefone: string
3. **Estacionamento:**
   * vagas: number
   * valorHora: number
   * veículos: Veículo[]
4. **Ticket:**
   * id: number
   * veículo: Veículo
   * horaEntrada: Date
   * horaSaída: Date
   * valorTotal: number

5.classe vaga

. numero de vaga : number

.vaga ocupada: number

1. classe veiculo

.placa; number

.cor; string

Modelo; string

**Funcionalidades:**

* **Registrar Entrada:**
  + Verifica se há vagas disponíveis.
  + Cria um novo ticket com a hora de entrada.
  + Adiciona o veículo à lista de veículos estacionados.
* **Registrar Saída:**
  + Calcula o tempo de permanência.
  + Calcula o valor total a ser pago.
  + Atualiza o ticket com a hora de saída e o valor total.
  + Remove o veículo da lista de veículos estacionados.
* **Emitir Ticket:**
  + Imprime ou exibe as informações do ticket.
* **Gerar Relatórios:**
  + Relatório de veículos estacionados.
  + Relatório de veículos que entraram em determinado período.
  + Relatório de faturamento diário/mensal.

**Implementação em TypeScript:**

TypeScript

class Veiculo {

constructor(public placa: string, public modelo: string, public cor: string) {}

}

class Cliente {

constructor(public nome: string, public cpf: string, public telefone: string) {}

}

class Estacionamento {

vagasOcupadas: number = 0;

constructor(public vagas: number, public valorHora: number, public veículos: Veículo[] = []) {}

registrarEntrada(veiculo: Veiculo): void {

// ... implementação

}

registrarSaida(placa: string): void {

// ... implementação

}

// ... outras funções

}

class Ticket {

id: number;

horaEntrada: Date;

horaSaida: Date;

valorTotal: number;

constructor(public veículo: Veiculo) {

this.id = Date.now();

this.horaEntrada = new Date();

}

}

// Exemplo de uso

const estacionamento = new Estacionamento(10, 5);

const carro = new Veiculo('ABC1234', 'Gol', 'Prata');

estacionamento.registrarEntrada(carro);

// ... outras operações

Use o código [com cuidado](/faq#coding).

**Considerações Adicionais:**

* **Persistência de Dados:** Para armazenar os dados de forma persistente, você pode utilizar um banco de dados (SQL ou NoSQL) ou um arquivo JSON.
* **Interface do Usuário:** Para criar uma interface amigável, você pode utilizar bibliotecas como React, Vue ou Angular.
* **Validações:** Implemente validações para garantir a integridade dos dados, como verificar se a placa já está cadastrada ou se há vagas disponíveis.
* **Pagamentos:** Integre o sistema com alguma forma de pagamento (cartão de crédito, débito, etc.).
* **Relatórios Personalizados:** Permita que o usuário gere relatórios personalizados de acordo com suas necessidades.

**Próximos Passos:**

* **Detalhar as funções:** Implementar as funções de cada classe com mais detalhes, incluindo a lógica de cálculo do valor total, geração de relatórios, etc.
* **Criar a interface do usuário:** Desenvolver a interface para interação com o usuário.
* **Testar o sistema:** Realizar testes unitários e de integração para garantir que o sistema funcione corretamente.

**Recursos Adicionais:**

* **TypeScript Handbook:** [https://www.typescriptlang.org/docs/handbook/basic-types.html](https://www.google.com/url?sa=E&source=gmail&q=https://www.typescriptlang.org/docs/handbook/basic-types.html)
* **Frameworks:** React, Angular, Vue
* **Bancos de dados:** PostgreSQL, MongoDB

Com essa base, você pode criar um sistema de estacionamento completo e personalizável, atendendo às suas necessidades específicas.