📦 API de Gerenciamento de Entregas e Veículos

Guia de Consulta e Estudo para a Prova de Desenvolvimento Web

Este projeto implementa uma API RESTful usando **Node.js**, **Express**, **TypeScript** e persistência em arquivos **JSON**. O foco é a modularização e regras de integridade de dados.

🚀 1. Comandos de Instalação (Terminal)

Se você estiver começando do zero no computador da prova, siga esta sequência no terminal (PowerShell/CMD):

```
# 1. Criar a pasta e entrar nela
mkdir prova-api
cd prova-api

# 2. Iniciar o projeto Node
npm init -y

# 3. Instalar dependências de Produção
npm install express uuid

# 4. Instalar dependências de Desenvolvimento (TypeScript)
npm install typescript @types/node @types/express @types/uuid tsx -D

# 5. Inicializar o TypeScript
npx tsc --init
```

📂 Criando Arquivos e Pastas Necessários

```
# Estrutura de pastas
mkdir src
mkdir src/controllers src/services src/models src/routes
mkdir data

# Arquivos de dados (IMPORTANTE: Devem ter [] dentro)
# No Windows PowerShell:
New-Item data/entregas.json -Value "[]"
New-Item data/veiculos.json -Value "[]"
```

⚙️ 2. Configurações Obrigatórias

Para o projeto rodar com `import/export` (ES Modules), você deve ajustar dois arquivos:

A. `package.json`

Adicione `"type": "module"` e o script de dev:

```
{
  "type": "module",
  "scripts": {
```

```
    "dev": "tsx watch src/index.ts"
  },
  ...restante do arquivo
}
```

B. `tsconfig.json`

Garanta que estas opções estejam configuradas:

```
{
  "compilerOptions": {
    "target": "es2020",
    "module": "NodeNext",
    "moduleResolution": "NodeNext",
    "outDir": "./dist",
    "rootDir": "./src",
    "strict": true,
    "esModuleInterop": true
  }
}
```

🧠 3. Resumo Teórico (Para explicar ao professor)

Estrutura MVC (Model - View - Controller)

O projeto está dividido em camadas para organização:

1. **Models (** `/models` **):** Apenas as interfaces (tipos) dos dados. Não tem lógica.
2. **Services (** `/services` **):** O "cérebro". Onde fica a regra de negócio (validações, leitura de arquivo, filtros).
3. **Controllers (** `/controllers` **):** O "porteiro". Recebe a requisição HTTP, chama o Service e devolve a resposta (200, 400, 500).
4. **Routes (** `/routes` **):** O "mapa". Liga a URL (`/entregas`) ao Controller.

A Regra dos Imports `.js`

- **Bibliotecas (npm):** Não usa extensão. Ex: `import express from 'express';`
- **Meus Arquivos:** Obrigatório usar `.js` no final. Ex: `import ... from './services.js';`
  - *Por que?* No modo `ESM` moderno, o Node não adivinha extensões. Ele precisa do caminho exato.

Integridade Referencial (Simulada)

É a regra que impede apagar um dado que está sendo usado por outro.

- *Exemplo:* Não posso apagar um Veículo se o ID dele estiver escrito dentro de uma Entrega com status 'Em Rota'.

📡 4. Documentação da API (Endpoints)

Use estes dados para testar no **Insomnia** ou **Postman**.

🚚 Veículos

Método	Rota	Descrição	Corpo (JSON)
POST	`/veiculos`	Cria veículo	`{ "placa": "ABC-1234", "modelo": "Fiorino", "status": "Disponível" }`
GET	`/veiculos`	Lista todos	-
DELETE	`/veiculos/:placa`	Deleta (se não estiver em uso)	-

📦 Entregas

Método	Rota	Descrição	Corpo (JSON)
POST	`/entregas`	Cria entrega	`{ "descricao": "PC Gamer", "status": "Pendente", "idVeiculo": null }`
GET	`/entregas`	Lista todas	-
GET	`/entregas/status/:status`	Filtra (ex: `/entregas/status/Pendente`)	-
PUT	`/entregas/associar`	**Associa Veículo**	`{ "idEntrega": "uuid-gerado", "placaVeiculo": "ABC-1234" }`
DELETE	`/entregas/:id`	Deleta entrega	-

💡 5. Principais Trechos de Código (Cola Rápida)

Como ler arquivo JSON assíncrono

```
// src/services/fileService.ts
const caminho = path.join(__dirname, '../../data', nomeArquivo);
const dados = await fs.readFile(caminho, 'utf-8');
return JSON.parse(dados);
```

Lógica de Integridade (Bloquear Delete)

```
// src/services/veiculoService.ts
const veiculoEmUso = entregas.some(e =>
    e.idVeiculo === placa && (e.status === 'Em Rota' || e.status === 'Pendente')
);

if (veiculoEmUso) {
    throw new Error('INTEGRITY_ERROR: Veículo em uso.');
```

```
}
```

Lógica de Associação (Regra de Negócio)

```
// src/services/entregaService.ts
if (veiculo.status !== 'Disponível') {
    throw new Error('Veículo indisponível');
}
entrega.idVeiculo = placaVeiculo;
entrega.status = 'Em Rota';
veiculo.status = 'Em Rota';
// Importante: Salvar os dois arquivos!
```

🆘 6. Solução de Problemas Comuns

1. **Erro: "Cannot find module..."**
   - *Causa:* Você esqueceu de colocar `.js` no final do import de um arquivo seu.
   - *Solução:* Mude `import x from './arquivo'` para `import x from './arquivo.js'`.
2. **Erro: "Unexpected end of JSON input"**
   - *Causa:* O arquivo `.json` na pasta `data` está totalmente vazio ou corrompido.
   - *Solução:* Abra o arquivo e coloque apenas um par de colchetes: `[]`.
3. **Erro: "TS2307: Cannot find module" (no VS Code)**
   - *Causa:* O VS Code se perdeu.
   - *Solução:* Pressione `F1 -> Developer: Reload Window` ou apenas confie no terminal se o código rodar.
4. **O Servidor não inicia**
   - *Causa:* Alguma porta já está em uso.
   - *Solução:* Mude a `PORT` no arquivo `index.ts` de 3000 para 3001.