

Atividade Backend Node.js Express

Instrumentos Musicais



Objetivos

Construir uma API RESTful completa do zero

Implementar CRUD para entidades principais: Produtos
(instrumentos), Clientes, Pedidos

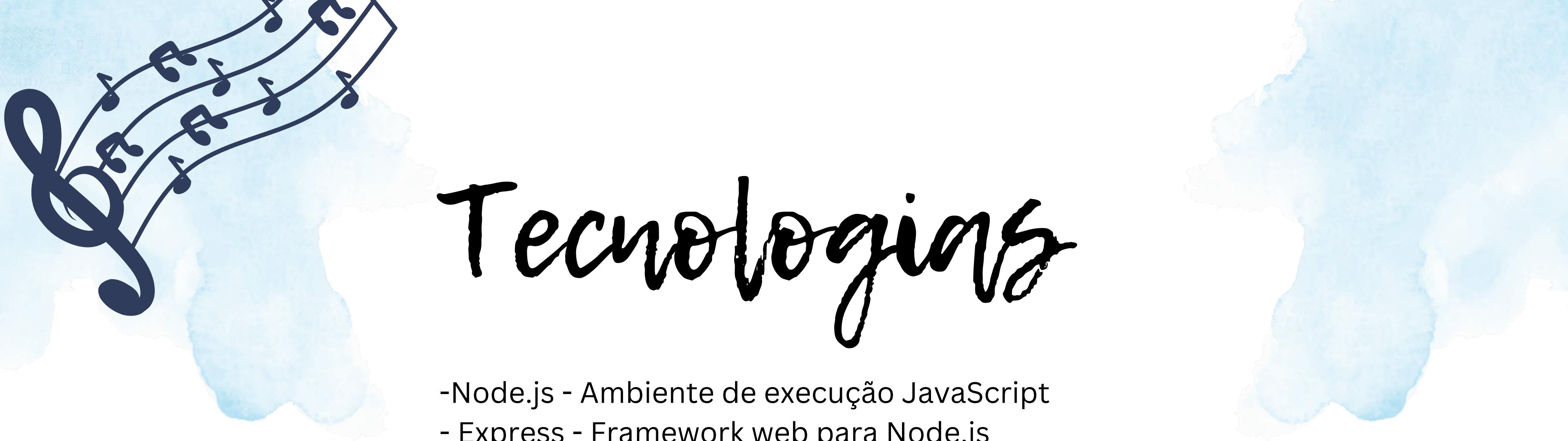
Organizar o projeto em pastas claras e reutilizáveis

Aplicar boas práticas: rotas, controllers, modelos,
middlewares, validação e tratamento de erros

Desenvolver uma API completa com:

- Node.js
- Express
- CRUD completo
- Organização de pastas
- Uso de rotas, controllers e modelos
- Cada aluno recebeu um comércio, o meu comércio:
Loja de Instrumentos Musicais





Tecnologias

- Node.js - Ambiente de execução JavaScript
- Express - Framework web para Node.js
- CORS - Middleware para habilitar CORS
- Nodemon - Ferramenta para desenvolvimento (reinicialização automática)



Entidades e relacionamentos

- Produto (Instrumento): id, nome, categoria (guitarra, teclado...), preço, estoque, descrição
- Cliente: id, nome, email, telefone, endereço
- Pedido: id, clientId, itens [{produtoid, quantidade, preco}], total, status

Relacionamentos:

- Cliente 1 – N Pedidos
- Pedido N – N Produtos (modelo simplificado: itens dentro do pedido)



1. Começo

- Criar pasta do projeto
- Executar: npminit -y
- Instalar dependências: express e nodemon



Inicialização

1

Criando o projeto

```
npm init -y
```

2

Instalando dependências

```
npm install express cors  
npm install nodemon -D
```

3

Scripts no package.json

```
"scripts": {  
  "dev": "nodemon src/server.js"  
}
```



Estrutura

```
projeto/
└── src/
    ├── controllers/
    │   ├── clienteController.js      # Lógica de clientes
    │   ├── pedidoController.js      # Lógica de pedidos
    │   └── produtoController.js      # Lógica de produtos
    ├── models/
    │   └── data.js                  # Dados em memória
    ├── routes/
    │   ├── clienteRoutes.js         # Rotas de clientes
    │   ├── pedidoRoutes.js          # Rotas de pedidos
    │   └── produtoRoutes.js          # Rotas de produtos
    └── server.js                  # Arquivo principal
    └── package.json
    └── package-lock.json
```



Configuração do Servidor Express

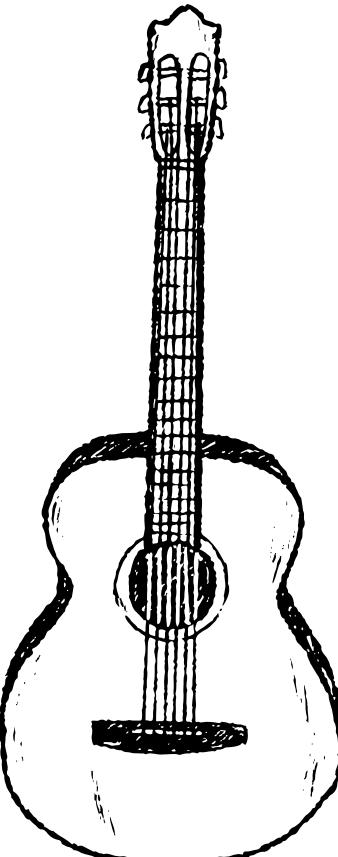
- Importar Express
- Criar app
- Middleware JSON
- Registrar rotas
- Subir servidor

Exemplo

```
const express = require('express');
const cors = require('cors');
const app = express();

app.use(cors());
app.use(express.json());

app.listen(3000, () => console.log("Servidor rodando!"));
```





Models

- Arquivo data.js
- Armazena dados em memória
- Produtos, clientes e pedidos
-



Controllers

Cada controller contém a lógica CRUD:

Exemplos de operações:

Criar

Listar

Buscar por ID

Atualizar

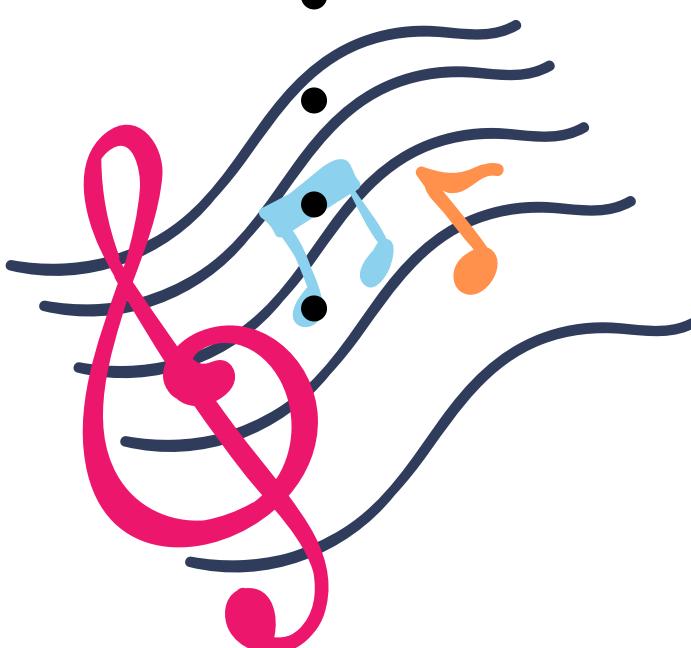
Deletar

Controladores usados:

produtoController

clienteController

pedidoController





Rotas

Cada entidade possui suas rotas:

Produtos

/api/produtos

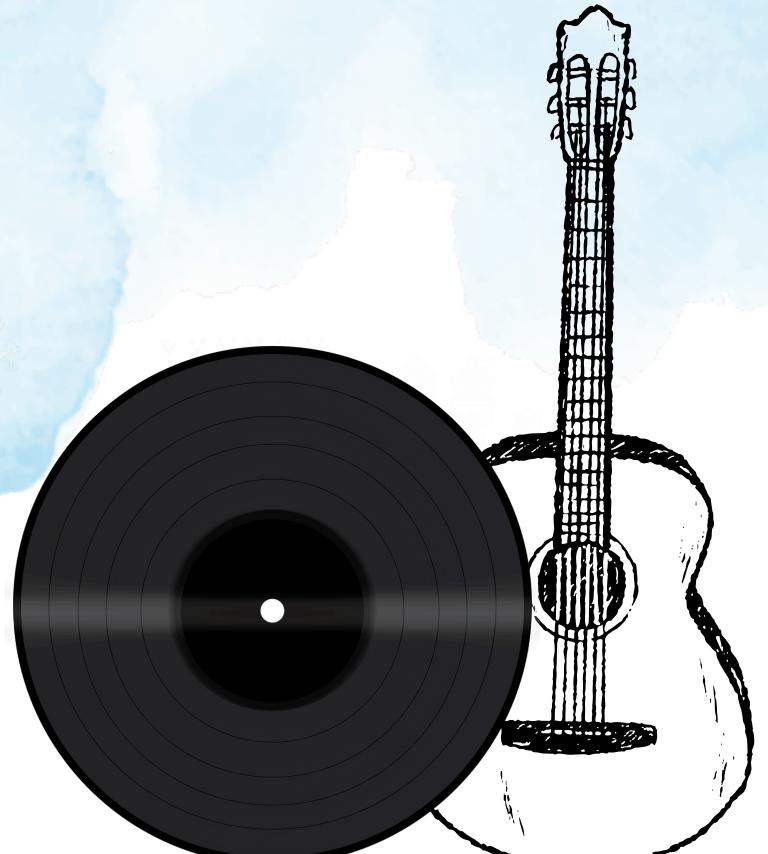
Clientes

/api/clientes

Pedidos

/api/pedidos

As rotas chamam os controllers responsáveis.



CRUD Implementado



Produtos

- ✓ GET
- ✓ POST
- ✓ PUT
- ✓ DELETE

Clientes

- ✓ GET
- ✓ POST
- ✓ PUT
- ✓ DELETE

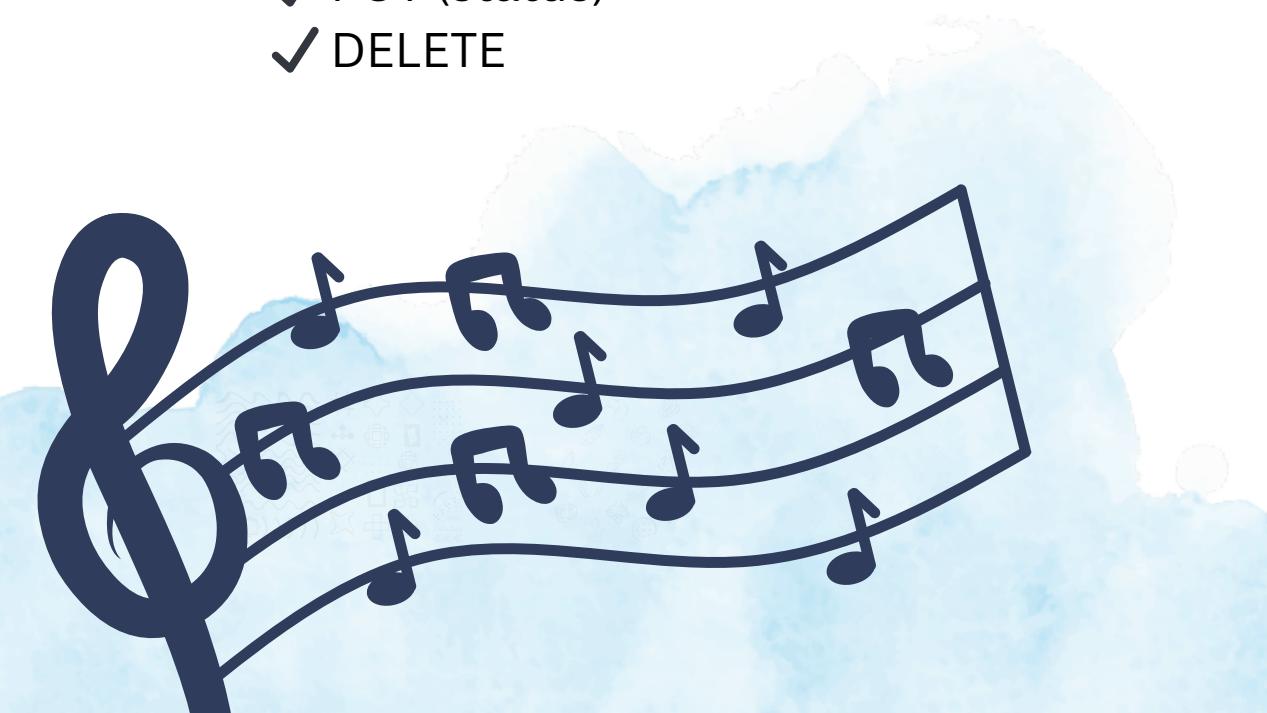
Pedidos

- ✓ GET
- ✓ POST
- ✓ PUT (status)
- ✓ DELETE

Método	Endpoint	Descrição
GET	`/produtos`	Lista todos os produtos
GET	`/produtos/:id`	Busca um produto por ID
POST	`/produtos`	Cria um novo produto
PUT	`/produtos/:id`	Atualiza um produto
DELETE	`/produtos/:id`	Remove um produto

Método	Endpoint	Descrição
GET	`/clientes`	Lista todos os clientes
GET	`/clientes/:id`	Busca um cliente por ID
POST	`/clientes`	Cria um novo cliente
PUT	`/clientes/:id`	Atualiza um cliente
DELETE	`/clientes/:id`	Remove um cliente

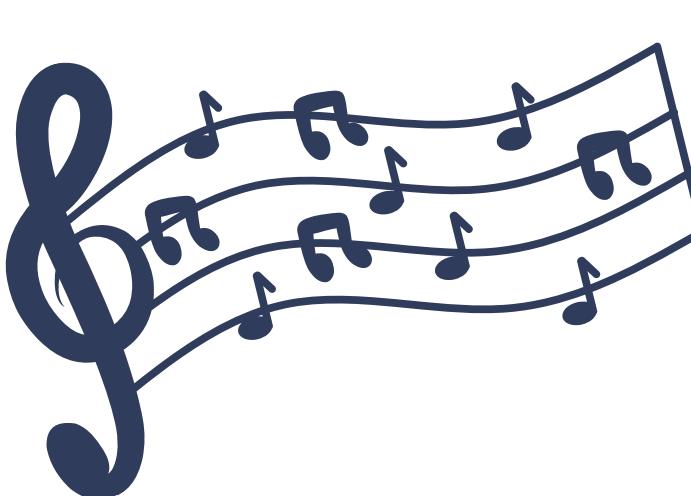
Método	Endpoint	Descrição
GET	`/pedidos`	Lista todos os pedidos
GET	`/pedidos/:id`	Busca um pedido por ID
POST	`/pedidos`	Cria um novo pedido
PUT	`/pedidos/:id`	Atualiza status do pedido
DELETE	`/pedidos/:id`	Cancela um pedido



Crud

- CREATE – adicionar instrumentos
- READ – listar e buscar
- UPDATE – editar
- DELETE – remover





Regras e Validações

- Produtos → nome, categoria e preço obrigatórios
- Clientes → nome e email obrigatórios
- Pedidos → clientelid, itens e quantidade obrigatórios
- Status do pedido:
- pendente, processando, enviado, entregue, cancelado





Conclusão

- API RESTful completa
 - Estrutura organizada
 - CRUD funcional
 - Ideal para loja de instrumentos musicais
- 

obrigada

