

🔗 Atividade Prática — Criando uma API REST com Express

🎯 Objetivo

Desenvolver uma **API REST completa (CRUD)** utilizando **Express**, com dados armazenados em **arrays (vetores)** e todos os endpoints implementados no **arquivo `app.js`**.

Cada aluno ou grupo receberá **um dos 7 temas** por sorteio e deverá implementar **5 endpoints CRUD + 1 endpoint personalizado**.

⚙️ Modelo inicial do projeto (`app.js`)

```
import express from "express";

const app = express();
app.use(express.json());

const PORT = 3000;

// Sua lógica e endpoints aqui

app.listen(PORT, () => {
  console.log(`Servidor rodando na porta ${PORT}`);
});
```

📄 Temas

📖 1. Catálogo de Livros

Campos: `id`, `titulo`, `autor`, `anoPublicacao`, `genero`

Regras:

- O título não pode se repetir
- O ano deve ser maior que 1900

Endpoints:

Método	Rota	Descrição
GET	<code>/livros</code>	Listar todos os livros
GET	<code>/livros/:id</code>	Buscar livro por ID
POST	<code>/livros</code>	Adicionar novo livro

Método	Rota	Descrição
PUT	/livros/:id	Atualizar informações
DELETE	/livros/:id	Remover livro
GET	/livros/autor/:autor	🔍 Listar livros de um autor específico

🎓 2. Sistema de Alunos

Campos: id, nome, idade, curso, matricula

Regras:

- Matrícula deve ser única
- Idade mínima: 16 anos

Endpoints:

Método	Rota	Descrição
GET	/alunos	Listar alunos
GET	/alunos/:id	Buscar aluno
POST	/alunos	Criar aluno
PUT	/alunos/:id	Atualizar aluno
DELETE	/alunos/:id	Excluir aluno
GET	/alunos/corso/:curso	🔍 Listar alunos de um curso específico

📞 3. Agenda de Contatos

Campos: id, nome, telefone, email

Regras:

- E-mail deve ser único
- Telefone deve ter ao menos 9 dígitos

Endpoints:

Método	Rota	Descrição
GET	/contatos	Listar contatos
GET	/contatos/:id	Buscar contato
POST	/contatos	Criar contato
PUT	/contatos/:id	Atualizar contato
DELETE	/contatos/:id	Excluir contato

Método	Rota	Descrição
GET	/contatos/nome/:nome	🔍 Buscar contatos por nome (parcial)

🎬 4. Gerenciador de Filmes

Campos: id, titulo, diretor, anoLancamento, classificacao

Regras:

- Título não pode se repetir
- Ano não pode ser maior que o atual

Endpoints:

Método	Rota	Descrição
GET	/filmes	Listar filmes
GET	/filmes/:id	Buscar filme
POST	/filmes	Cadastrar filme
PUT	/filmes/:id	Atualizar filme
DELETE	/filmes/:id	Remover filme
GET	/filmes/classificacao/:classificacao	🔍 Listar filmes por classificação (ex: 12, 16, 18)

👨‍🍳 5. Cadastro de Receitas

Campos: id, nome, ingredientes, modoPreparo, tempoPreparo

Regras:

- Nome não pode repetir
- Tempo de preparo deve ser positivo

Endpoints:

Método	Rota	Descrição
GET	/receitas	Listar receitas
GET	/receitas/:id	Buscar receita
POST	/receitas	Adicionar receita
PUT	/receitas/:id	Atualizar receita
DELETE	/receitas/:id	Excluir receita

Método	Rota	Descrição
GET	/receitas/ingrediente/:ingrediente	 Buscar receitas que contenham determinado ingrediente


☑ 6. Controle de Tarefas

Campos: id, descricao, status, prioridade, responsavel

Regras:

- Status deve ser "pendente", "em andamento" ou "concluída"
- Prioridade deve ser "baixa", "média" ou "alta"

Endpoints:

Método	Rota	Descrição
GET	/tarefas	Listar tarefas
GET	/tarefas/:id	Buscar tarefa
POST	/tarefas	Criar tarefa
PUT	/tarefas/:id	Atualizar tarefa
DELETE	/tarefas/:id	Excluir tarefa
GET	/tarefas/status/:status	 Listar tarefas filtradas por status


7. Sistema de Veículos

Campos: id, modelo, marca, ano, placa

Regras:

- Placa deve ser única
- Ano não pode ser maior que o ano atual

Endpoints:

Método	Rota	Descrição
GET	/veiculos	Listar veículos
GET	/veiculos/:id	Buscar veículo
POST	/veiculos	Cadastrar veículo
PUT	/veiculos/:id	Atualizar veículo
DELETE	/veiculos/:id	Excluir veículo
GET	/veiculos/marca/:marca	 Listar veículos de uma marca específica

Dicas

- Use `try...catch` em cada endpoint para capturar erros.
 - Utilize métodos como `find()`, `filter()` e `findIndex()` para manipular o array.
 - Faça testes no **Postman** ou **Insomnia** para validar as rotas.
 - Lembre-se de converter parâmetros de rota (`req.params`) quando necessário (ex: `parseInt()`).
-

Entrega esperada

- CRUD completo dentro do `app.js`
 - Dados armazenados em **arrays**
 - Uso de `try/catch` para tratamento de erros
 - Implementação do endpoint personalizado
 - Testes realizados via Insomnia
-

Sorteio dos temas

Número	Tema
1	Catálogo de Livros
2	Sistema de Alunos
3	Agenda de Contatos
4	Gerenciador de Filmes
5	Cadastro de Receitas
6	Controle de Tarefas
7	Sistema de Veículos
