mchiodi

Aprenda a criar um CRUD completo com Node.js!

veja mais!

01

O Que é CRUD?

CRUD é um conceito fundamental no desenvolvimento web e representa as quatro operações básicas para manipulação de dados em um sistema:

✓ Create (Criar) → Adicionar novos registros ao banco de dados

✓ Read (Ler) → Buscar informações armazenadas

✓ Update (Atualizar) → Modificar registros existentes

✓ Delete (Deletar) → Remover dados do banco

02

Instalando as Dependências

⚙️ Antes de começar, precisamos instalar algumas ferramentas:

1️⃣ Node.js → Para rodar o código no servidor

2️⃣ Express.js → Framework que simplifica a criação de APIs

3️⃣ Banco de Dados (MongoDB ou MySQL) → Para armazenar as informações

03

Estruturando o Projeto

Organização é essencial para manter o código limpo e escalável!

💡 Aqui está um exemplo de estrutura de pastas para o CRUD:

📂 meu-projeto/
- 📂 controllers/ → Lógica das operações do CRUD
- 📂 models/ → Modelos de dados do banco
- 📂 routes/ → Definição das rotas da API
- 📂 config/ → Configuração do banco de dados
- 📄 server.js → Arquivo principal do servidor

04

Criando a Rota de Criação (Create)

A rota POST permite adicionar novos usuários ao banco de dados.

🛠️ Passo a passo:

- Criamos um endpoint /users
- Recebemos os dados via JSON
- Armazenamos no banco de dados
- Retornamos uma mensagem de sucesso

Rota de Criação

```
const User = require('./models/User');

app.post('/users', async (req, res) => {
  try {
    const { name, email, age } = req.body;
    if (!name || !email || !age) {
      return res.status(400).json({ error: '
Todos os campos são obrigatórios' });
    }

    const newUser = new User({ name, email, age });
    await newUser.save();

    res.status(201).json({ message: '
Usuário criado com sucesso!', user: newUser });
  } catch (error) {
    res.status(500).json({ error: '
Erro ao criar usuário' });
  }
});
```

05

Criando a Rota de Leitura (Read)

A rota GET permite visualizar os usuários cadastrados.

🔍 Como funciona?

- Criamos um endpoint /users
- Buscamos todos os registros no banco
- Retornamos uma lista em JSON

Rota de Leitura

```
app.get('/users', async (req, res) => {
  try {
    const users = await User.find();
    res.status(200).json(users);
  } catch (error) {
    res.status(500).json({ error: '
Erro ao buscar usuários' });
  }
});
```

06

Criando a Rota de Atualização (Update)

A rota PUT permite modificar um usuário existente.

🔄 Passos da atualização:

1 O usuário envia o ID e os novos dados

2 O servidor localiza o registro no banco

3 Atualizamos as informações

4 Retornamos uma mensagem de confirmação

Rota de Atualização

```
app.put('/users/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const { name, email, age } = req.body;

    const updatedUser = await User.findByIdAndUpdate(
      id,
      { name, email, age },
      { new: true }
    );

    if (!updatedUser) {
      return res.status(404).json({ error: '
Usuário não encontrado' });
    }

    res.status(200).json({ message: '
Usuário atualizado com sucesso!', user: updatedUser });
  } catch (error) {
    res.status(500).json({ error: 'Erro ao atualizar usuário'
 });
  }
});
```

07

Criando a Rota de Exclusão (Delete)

A rota DELETE remove um usuário do banco de dados.

🚨 O que acontece?

- O cliente envia um ID válido
- O servidor procura esse usuário
- Se encontrado, ele é removido do banco
- Retornamos uma mensagem de sucesso

Rota de Exclusão

```
app.delete('/users/:id', async (req, res) => {
  try {
    const { id } = req.params;

    const deletedUser = await User.
findByIdAndDelete(id);

    if (!deletedUser) {
      return res.status(404).json({ error: '
Usuário não encontrado' });
    }

    res.status(200).json({ message: '
Usuário removido com sucesso!' });
  } catch (error) {
    res.status(500).json({ error: '
Erro ao remover usuário' });
  }
});
```

08

Testando a API no Postman

gora que todas as rotas estão prontas, vamos testar a API!

📌 Dicas:

✅ Use Postman ou Insomnia para testar requisições

✅ Verifique os códigos de status HTTP:

- 200 → Sucesso
- 201 → Criado
- 400 → Erro de requisição
- 404 → Não encontrado

✅ Simule cadastros, buscas, atualizações e remoções

Gostou desse tutorial?

Comenta aqui e compartilha com quem está aprendendo!