mchiodi

7 Dicas Essenciais para Melhorar a Performance da sua API Node.js

veja mais!

O QUE É RESERRO COMUM: API LENTA E SOBRECARREGADAT?

Problema: APIs que processam muitas requisições simultâneas podem sofrer com lentidão, consumo excessivo de recursos e falhas.

🔹 Motivo: Código mal otimizado, falta de cache, consultas ineficientes ao banco de dados e falta de controle de concorrência.

📌 Como resolver? Confira as 7 melhores práticas nos próximos slides! 🚀

USE CACHE PARA REDUZIR REQUISIÇÕES

✅ O que é?

O cache armazena respostas frequentes para evitar consultas repetitivas ao banco de dados, reduzindo o tempo de resposta.

🔥 Como implementar?

✓ Use Redis para armazenar requisições temporárias.

✓ Utilize ETags para evitar o envio de dados desnecessários.

✓ Implemente cache no front-end para reduzir chamadas à API.

OTIMIZE AS CONSULTAS NO BANCO DE DADOS

✅ O que é?
Muitas APIs sofrem lentidão porque fazem consultas ineficientes e desnecessárias ao banco de dados.

🔥 Como otimizar?
✔️ Use índices no banco para buscas mais rápidas.
✔️ Evite SELECT *, retornando apenas os campos necessários.
✔️ Para bancos relacionais (MySQL, PostgreSQL), use ORMs eficientes como Sequelize ou TypeORM.
✔️ Para NoSQL (MongoDB), use Mongoose e agregações otimizadas.

»»

USE PAGINAÇÃO PARA GRANDES VOLUMES DE DADOS

Se sua API retorna milhares de registros em uma única requisição, isso sobrecarrega o servidor e torna o tempo de resposta lento.

🔥 Como resolver?

✓ Use limit e offset para retornar apenas parte dos dados:

{ GET /users?limit=10&offset=20 }

✓ Utilize cursor-based pagination para grandes bases de dados.

✓ Sempre forneça informações de paginação no retorno da API.

»»

IMPLEMENTE LAZY LOADING E STREAMING

✅ O que é?

🔹 Lazy Loading → Carrega apenas os dados necessários, melhorando a eficiência.

🔹 Streaming → Permite processar grandes arquivos sem sobrecarregar a memória.

🔥 Como otimizar?

✔️ Use res.json() para enviar apenas os dados essenciais.

✔️ Para arquivos grandes, implemente streams do Node.js em vez de carregar tudo na memória.

✔️ No banco de dados, carregue relações somente quando necessário.

»»

USE COMPRESSÃO DE DADOS

✅ O que é?
A compressão reduz o tamanho das respostas enviadas pela API, tornando as requisições mais rápidas.

🔥 Como implementar?
✓ Habilite Gzip ou Brotli no servidor:

```
const compression = require('compression');
app.use(compression());
```

✓ Minimize o uso de JSON aninhado, retornando apenas os dados essenciais.

UTILIZE WORKERS E PROCESSAMENTO ASSÍNCRONO

✅ O que é?

Se sua API processa tarefas pesadas, ela pode travar o event loop, deixando tudo lento.

🔥 Como resolver?

✔️ Use Node.js Worker Threads para rodar tarefas em paralelo.

✔️ Para tarefas demoradas (ex: envio de e-mails, geração de PDFs), use fila de processamento com BullMQ e Redis.

✔️ Evite bloquear o event loop com loops síncronos.

MONITORE E FAÇA LOGS DE PERFORMANCE

✅ O que é?

APIs bem monitoradas conseguem identificar gargalos antes que se tornem problemas.

🔥 Como otimizar?

✓ Use PM2 para gerenciar processos no Node.js.

✓ Implemente logs detalhados com winston ou morgan.

✓ Monitore erros e tempo de resposta com Sentry, New Relic ou Grafana.

Agora você já sabe como melhorar a performance da sua API com Node.js!

Comenta aqui se quiser mais tutoriais assim! 👇