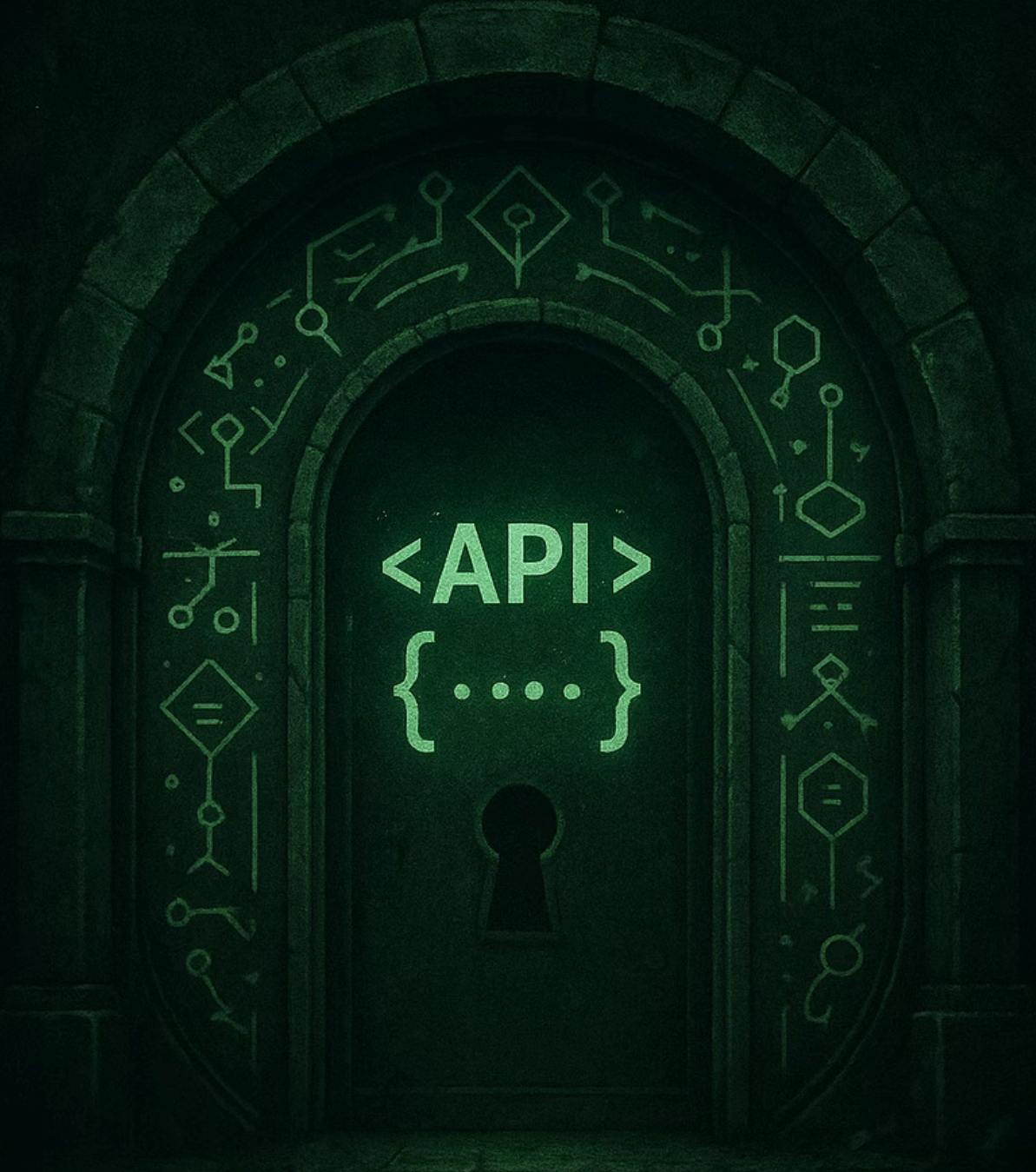


A CÂMARA DAS APIs OCULTAS

DOMINE FASTIFY NO NODE.JS



<API>
{...}

Explore este guia completo sobre Fastify, o framework de Node.js que promete revolucionar a construção de APIs. Descubra por que ele é a escolha dos desenvolvedores que buscam performance, eficiência e uma experiência mágica no desenvolvimento de aplicações.

"Aqueles que buscam a velocidade e a clareza nas APIs encontrarão seu caminho através dos portais do Fastify."



Capítulo 1 Despertando a Magia: O que é o Fastify e por que usá-lo?

No vasto mundo das APIs, cada desenvolvedor precisa de uma varinha poderosa e o **Fastify** surge como uma das mais afiadas do arsenal Node.js. Enquanto muitos ainda empunham o velho e confiável **Express**, os magos modernos da performance estão migrando para um framework mais leve, rápido e encantadoramente extensível: o **Fastify**.

Fastify vs Express <Duelo de Magia/>

Performance	 Extremamente rápido	 Razoável, mas mais pesado
Tipagem nativa	 Suporte robusto com TypeScript	 Precisa de configuração extra
Plugins	 Modular e escalável	 Limitado
Validação	 Embutida com JSON Schema	 Manual ou com libs externas
Comunidade	 Crescendo rápido	 Estabelecida

Por que usar o Fastify?

- Desempenho absurdo
- Validação embutida
- Escalabilidade modular
- Experiência fluida com TypeScript

Casos de Uso Ideais

- APIs REST leves e performáticas
- Microserviços com Node.js
- Gateways de API
- Apps que priorizam performance

Capítulo 2 Preparando a Poção: Configurando seu Projeto com Fastify

Ingredientes:

- Node.js 18+
- NPM/Yarn
- VSCode (e um pouco de coragem)

Iniciando o Projeto

```
mkdir camara-das-apis
cd camara-das-apis
npm init -y
npm install fastify
```

Com TypeScript

```
npm install -D typescript ts-node @types/node
npx tsc --init
```

Scripts:

```
"scripts": { "dev": "ts-node src/server.ts" }
```

Estrutura de Projeto

```
camara-das-apis/
├── src/
│   └── server.ts
├── tsconfig.json
└── package.json
```

Server Base

```
import Fastify from 'fastify';
const app = Fastify();
app.get('/', async () => {
```


Capítulo 3 Criando Encantamentos: Rotas, Handlers e Validações

Rotas e Handlers

Definir rotas no Fastify é como criar feitiços que respondem a diferentes chamados.

```
app.get('/users', async () => [ { id: 1, nome: 'Hermione' } ]);
app.post('/users', async (req, reply) => {
  const { nome } = req.body;
  return { id: 2, nome };
});
```

O Fastify cuida automaticamente da serialização de respostas e do parsing de corpo (JSON, formulários, etc.), tornando o desenvolvimento mais fluido e menos propenso a erros.

Validação com Zod + Fastify-sensible

A validação é crucial para APIs robustas. O Fastify possui um sistema de validação poderoso, e com **Zod** e **fastify-sensible**, fica ainda mais mágico.

```
npm install zod fastify-sensible
```

```
import { z } from 'zod';
const schema = z.object({ nome: z.string().min(3) });
```

Você pode aplicar este schema diretamente às rotas para garantir que os dados de entrada estejam sempre em conformidade. Isso evita que dados mágicos indesejados corrompam sua base de dados.

Boas Práticas

Para manter a ordem na sua câmara de APIs, é essencial organizar seus arquivos:

```
src/
├── routes/user.ts
├── services/userService.ts
├── schemas/userSchema.ts
```

Separar as preocupações garante que seus feitiços (código) sejam mais manuteníveis e escaláveis, mesmo em projetos complexos.

Capítulo 4 Artefatos Mágicos: Plugins e Middlewares Essenciais

O poder do Fastify reside em sua arquitetura baseada em plugins. Eles são como artefatos mágicos que você pode adicionar para estender as funcionalidades do seu servidor sem poluir o código principal.

CORS

Permita que outras origens interajam com sua API.

```
npm install @fastify/cors
```

```
app.register(require('@fastify/cors'), { origin: '*' });
```

JWT (JSON Web Tokens)

Para autenticação segura de usuários.

```
app.register(require('@fastify/jwt'), { secret: 'chave' });
```

Swagger (OpenAPI)

Gere documentação interativa para sua API automaticamente.

```
npm install @fastify/swagger @fastify/swagger-ui
```

```
app.register(require('@fastify/swagger'), {  
  // Configurações do Swagger...  
});  
app.register(require('@fastify/swagger-ui'), {  
  routePrefix: '/docs',  
});
```

Decorators

Adicione novas funcionalidades ao objeto Fastify ou aos objetos `request/reply`.

Made with **GAMMA**

Capítulo 5

Abrindo os Portais: Deploy e Manutenção da API

Depois de conjurar sua API com Fastify, é hora de abri-la para o mundo. O deploy e a manutenção são passos cruciais para garantir que sua câmara de APIs permaneça acessível e eficiente.

Testando com Postman ou Insomnia

Antes de mais nada, teste exaustivamente seus endpoints. Use ferramentas como Postman ou Insomnia para simular requisições e garantir que tudo está funcionando conforme o esperado.

- GET <http://localhost:3000>
- POST com JSON no corpo da requisição

Docker

Empacote sua aplicação em um contêiner para implantação consistente.

```
FROM node:18
WORKDIR /app
COPY . .
RUN npm install
CMD ["npm", "run", "dev"]
EXPOSE 3000
```

PM2 (Process Manager)

Mantenha sua aplicação rodando continuamente em produção.

```
npm install -g pm2
pm2 start src/server.ts --name camara-api --interpreter ts-node
```

Monitoramento

Fique de olho na saúde da sua API para garantir que ela opere com o máximo desempenho.

- Logs com `app.log`
- Dashboard com Prometheus/Grafana para métricas em tempo real

Agradecimentos

A todos os devs bruxos, bruxas e mestres das trevas que continuam acreditando na magia da performance, na alquimia do código limpo e na coragem de adotar novas ferramentas.

Este grimório é para você.

Feito com feitiços, Fastify e café.



✨ **Obrigado por conjurar conhecimento junto comigo!**



Mantenha a magia viva!