

Desenvolvimento de uma API de autenticação utilizando Express, JWT e MongoDB

Prof. Fernando Belém
`fernandol Luiz@cotemig.com.br`

Uma rápida teoria...

► O que é API?

- “APIs são mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.”

Fonte: <<https://aws.amazon.com/pt/what-is/api/>>

Uma rápida teoria...

► O que é Express?

- “O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móveis.”

Fonte: <<https://expressjs.com/pt-br/>>

Uma rápida teoria...

► O que é JWT?

- “JSON Web Token (JWT) é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações com segurança entre as partes como um objeto JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente.”

Fonte: <<https://jwt.io/introduction>>

Uma rápida teoria...

► O que é MongoDB?

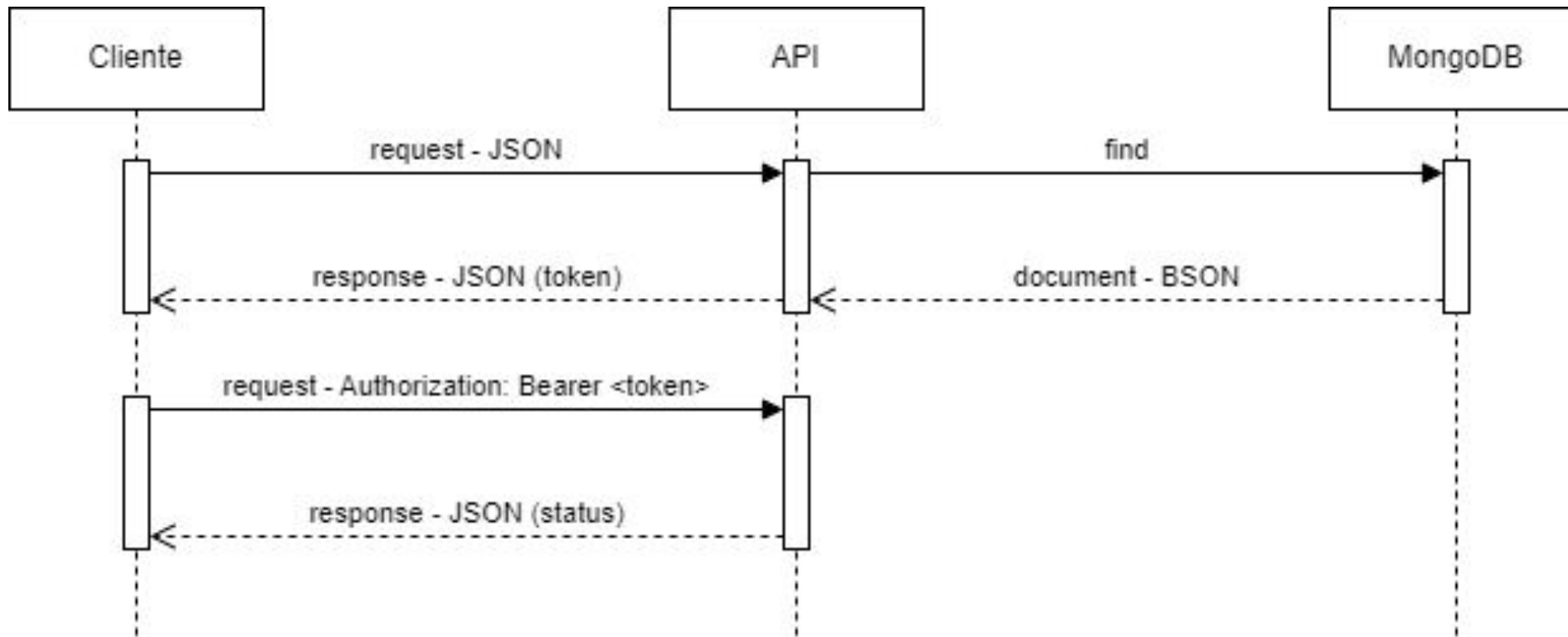
- “MongoDB é um banco de dados de documentos projetado para facilitar o desenvolvimento e dimensionamento de aplicativos.”

Fonte: <<https://www.mongodb.com/docs/manual/>>

Objetivos da Oficina

- ▶ Criar uma API compacta que possa ser integrada em um ambiente cuja arquitetura é baseada em microsserviços.
- ▶ Essa API terá como principal funcionalidade realizar a pesquisa de usuários em uma base de dados mongodb. Caso o usuário exista, ela irá gerar um token de autenticação com validade de 1 minuto.
- ▶ A API também irá prover um serviço de verificação se o token está válido ou não.

Fluxo de funcionamento da API



Mão na massa...



<https://github.com/nandobhx/auth-app>

Mão na massa...

- ▶ Criando uma instância do banco de dados MongoDB:
 - Criar um diretório “db”;
 - Iniciar a instância através do comando:

```
mongod --dbpath <caminho>
```

Mão na massa...

- ▶ Instalando as dependências do projeto:
 - Abrir a pasta do projeto no VSCode;
 - No terminal executar o seguinte comando:

npm i

Mão na massa...

- ▶ Ativando a compilação iterativa do TypeScript:
 - No terminal executar o seguinte comando:

```
npx tsc -w
```

Mão na massa...

- ▶ Criando o arquivo de variáveis de ambiente para aplicação:
 - Na raiz do projeto criar o arquivo “.env”;
 - Inserir as seguintes chaves:

DB_HOST="localhost"

DB_PORT="27017"

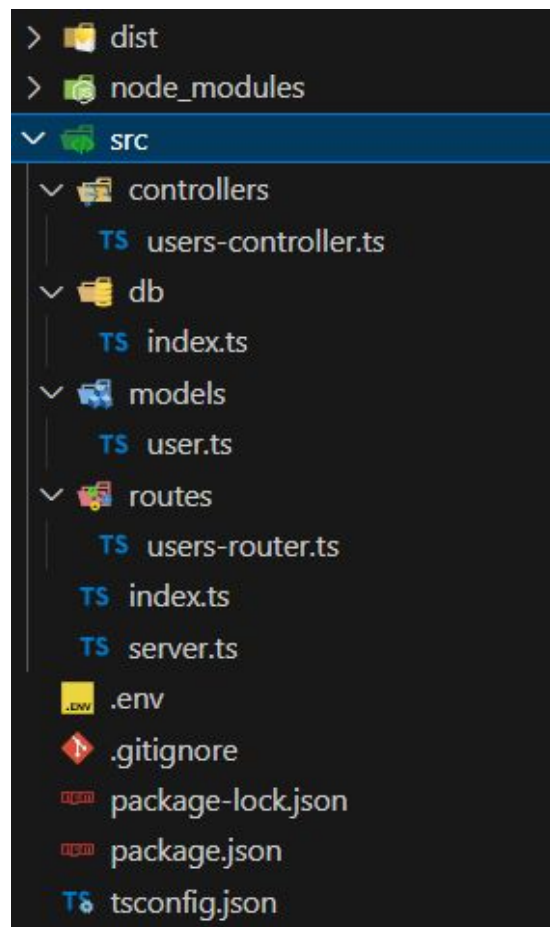
DB_NAME="auth"

HTTP_PORT="3000"

JWT_SECRET="COTEMIG@2024"

Mão na massa...

- Navegando pela estrutura de diretórios do projeto:



Mão na massa...

- Model:

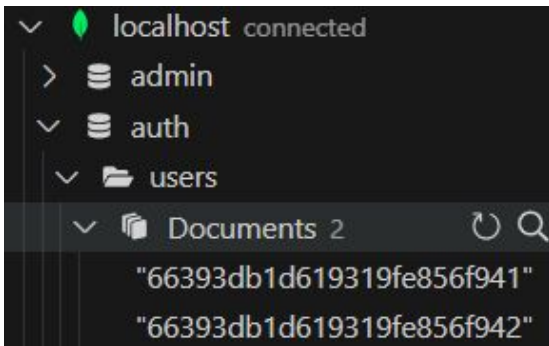
```
export default interface User {  
  fullname: string;  
  username: string;  
  password: string;  
  role: number;  
}
```

Mão na massa...

- ▶ Fazendo a carga de usuários no banco de dados MongoDB:
 - Abrir um novo terminal e executar o comando:

```
node .\dist\index.js
```

- Para visualizar os documentos criados iremos utilizar a extensão MongoDB do VSCode.



Mão na massa...

- ▶ Iniciando o servidor web Express:
 - Em um novo terminal executar o seguinte comando:

```
node .\dist\server.js
```


Mão na massa...

- ▶ Navegando pelos fontes...
 - `db/index.ts`
 - `server.ts`
 - `users-routers.ts`
 - `users-controller.ts`



Mão na massa...



<https://github.com/nandobhx/api-tester>

Mão na massa...

► Testando a rota POST:

API Tester

REQUEST

Tecnologia

Fetch

URL

POST

Header

Authorization

Bearer <token>

Body - JSON

```
{  
  "username": "nando",  
  "password": "123456"  
}
```

Enviar Limpar

RESPONSE

Status

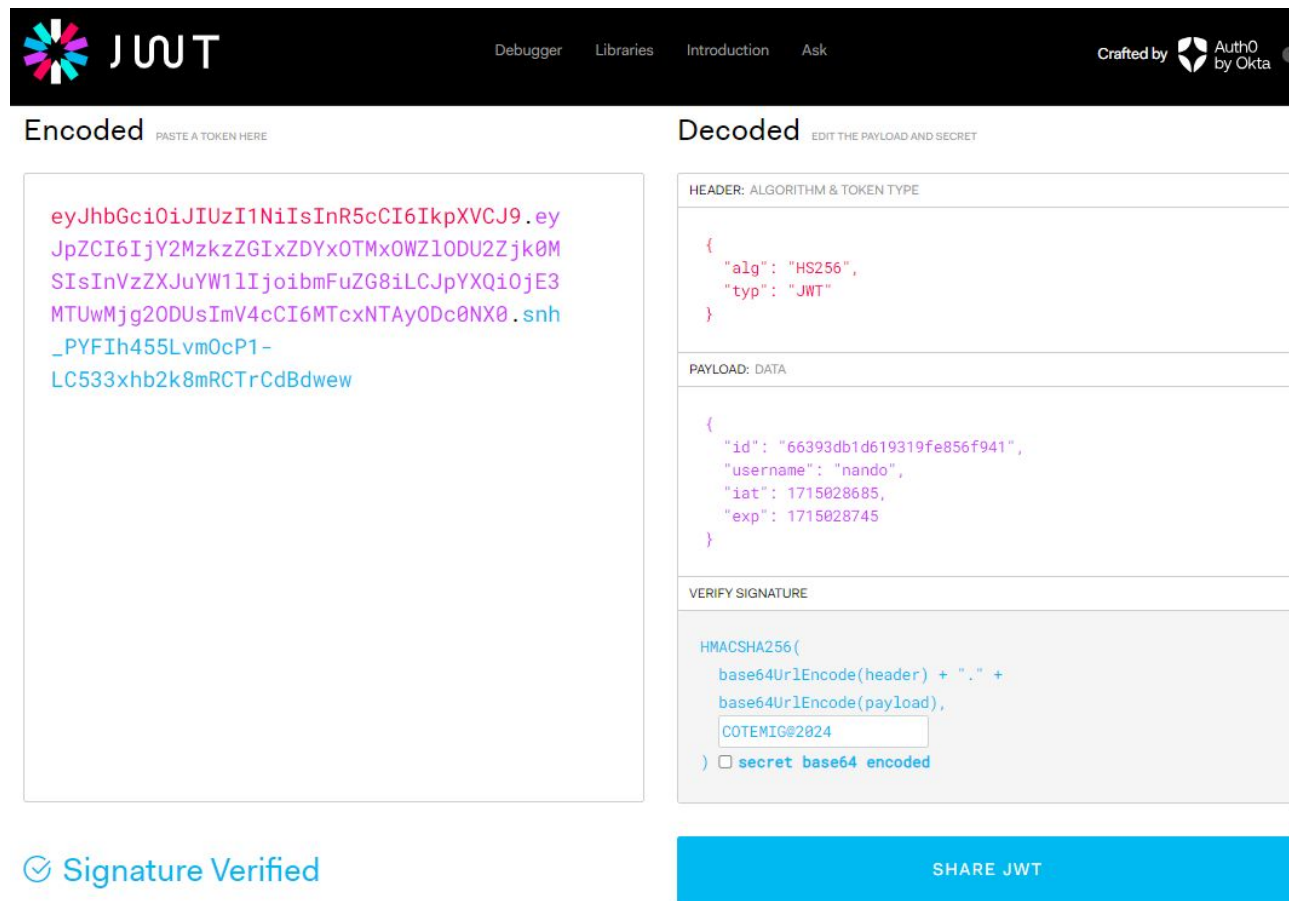
200 - OK

Body

```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MzkzZGIxZDYxOTMxOWZlODU2Zjk0MSIsInVzZXJuYW1lIjoibmFuZG8iLCJpYXQiOi0jE3MTUwMjg1ODIsImV4cCI6MTcxNTAyODY0Mn0. qgyfFKEPo23oH4ccRnWYooVV-5UCg_3wJTmZNsc0B5E"  
}
```

Mão na massa...

- Visualizando o payload do JWT:



The image shows the JWT.io web application interface. The top navigation bar includes links for 'Debugger', 'Libraries', 'Introduction', and 'Ask', along with a 'Crafted by Auth0 by Okta' logo. The main content area is split into two panels: 'Encoded' and 'Decoded'.

Encoded Panel: Labeled 'PASTE A TOKEN HERE', it contains the following JWT token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MzkzZGIxZDYxOTMxOWZlODU2Zjk0MSIsInVzZXJuYWI1IjoibmFuZG8iLCJpYXQiOiE3MTUwMjg2ODUzImV4cCI6MTcxNTAyODc0NX0.snh_PYFIh455Lvm0cP1-LC533xhb2k8mRCTrCdBdwew
```

Decoded Panel: Labeled 'EDIT THE PAYLOAD AND SECRET', it displays the token's structure:

- HEADER: ALGORITHM & TOKEN TYPE:**

```
{  "alg": "HS256",  "typ": "JWT"}
```
- PAYLOAD: DATA:**

```
{  "id": "66393db1d619319fe856f941",  "username": "nando",  "iat": 1715028685,  "exp": 1715028745}
```
- VERIFY SIGNATURE:** Shows the HMACSHA256 algorithm and a text input field containing 'COTEMIG@2024'. Below the input, there is a checkbox labeled 'secret' and the text 'base64 encoded'.

At the bottom left, a green checkmark icon is followed by the text 'Signature Verified'. At the bottom right, there is a blue button labeled 'SHARE JWT'.

Mão na massa...

► Testando a rota GET:

API Tester

REQUEST

Tecnologia
Fetch

URL
GET

Header
Authorization
Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY2MzkzZGlxZDYxOTMxO

Body - JSON

Enviar Limpar

RESPONSE

Status
200 - OK

Body

```
{"status":200,"message":"Autorizado!"}
```

The background features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.

Obrigado!

Prof. Fernando Belém
`fernandolui@cotemig.com.br`