TRYBE
Modulo III – Back-end

Bloco 28: Autenticação e Upload de Arquivos

1) JWT (Json Web Token)

Objetivo da aula: Entender melhor tokens, saber gerar e autenticar rotas express.

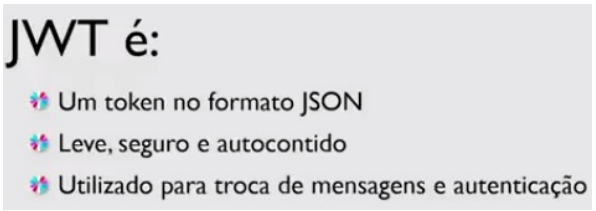
Importância do JWT

Padrão de mercado que define um token no formato JSON para a troca de informações.
Bom jeito de obter identidade de user, alternativa aos cookies. JWT disponibiliza um token/hash/código criptografado que pode enviar para uma API e validar como preferir.

Caracteristicas: leve (json), autocontido, seguro.

Vantagens: sem BD, reutilizável entre vários serviços.

Quando usar: para troca de informações entre aplicativos, em mecanismos de autenticação.



Documentação oficial.

O que é JWT

Token gerado a partir de dados "pessoais" que pode ser trafegado pela internet via URL. O tráfego acontece via requisições POST ou em um header HTTP com "segurança".

Sistemas de **criptografia** encapsulados nas bibliotecas do JWT:

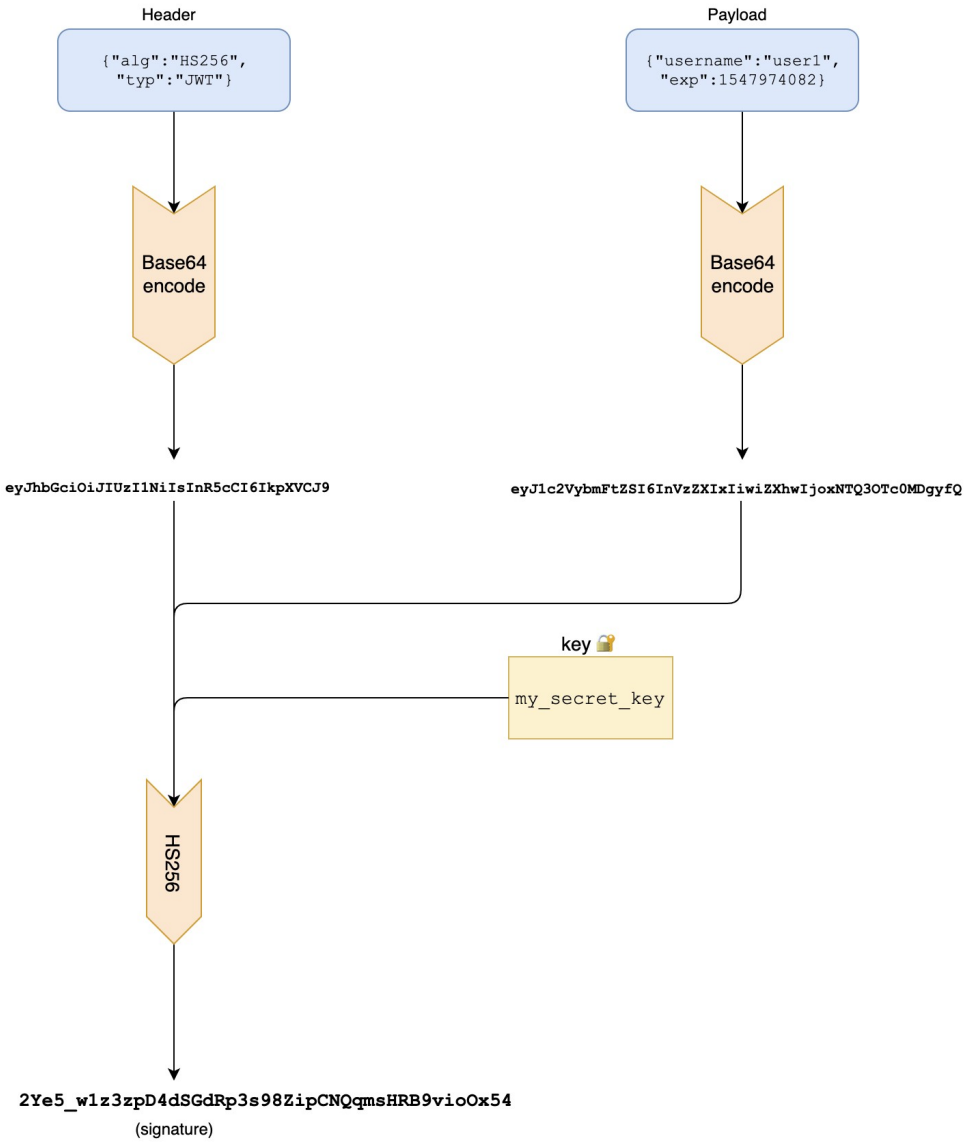
→ **HMAC**
Algoritmo para gerar um MAC (MAC (código de autenticação de mensagem).

HMAC(K, m) = hash(K1 + hash(K2 + m))
// Sendo K a chave secreta, m a mensagem, hash é a função de hash escolhida (md5, sha1 etc), K1 e K2 chaves secretas derivadas;
+ a operação de concatenação de strings.

→ **RSA**
Combinação de chave pública e chave privada.

Estrutura do JWT

Resultado final
(Header em base64).
(Payload em base64).
(Signature em base64)



Header
Duas propriedades:
- tipo do token (JWT)
- tipo do algoritmo de hash (HMAC-SHA256, RSA)

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

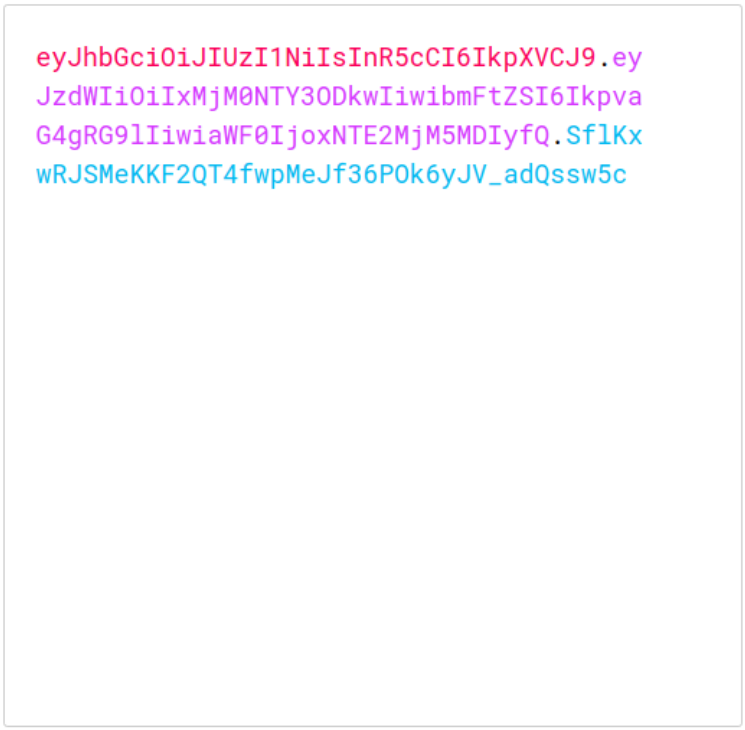
Payload
Tem dados, claims (reinvidicações) sobre entidade:

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Signature
Gera o hash JWT usando o header e payload em *base64*, usando o alg do header.

Um exemplo de envio de um JWT via header em uma requisição HTTP:

```
GET /foo/bar HTTP/1.1
Host: www.exemplo.com
Authorization: Bearer (Header em base64).(Payload em base64).(Signature em base64)
```



APIs com JWT

Precisamos alterar o code das APIs para adicionar autenticação via JWT e assim saber:
- se o usuário está de fato autenticado;
- quem essa pessoa é;
- setar um tempo de sessão para ela;
- permitir que ela utilize apenas uma autenticação para trafegar entre várias aplicações, etc.

Implementar JWT

→ **Instalar**
npm install jsonwebtoken

→ **controllers/login.js – code para gerir token:**

```
const jwt = require('jsonwebtoken'); //...
const secret = 'seusecretdetoken'; //...
const jwtConfig = {
    expiresIn: '7d',
    algorithm: 'HS256',
  }; //...
const token = jwt.sign({ data: user }, secret, jwtConfig);
res.status(200).json({ token });
```

// resumindo esse code é para criar o segredo, configurar o jwt com header, tirar info sensível do user para entrar afirmações no payload do token e assinar o "bilhete" que o user vai usar para dizer que foi bem autorizado e emitido por nós.

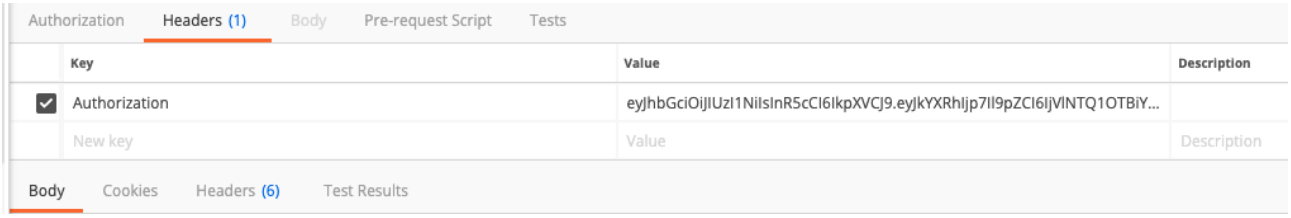
```
const payload = {
  iss: 'post-api', // Issuer -> Quem emitiu o token
  aud: 'frank-api', // Audience -> Quem deve aceitar este token
  sub: user._id, // Subject -> A quem se refere esse token
  userData: user,
};

const token = jwt.sign(payload, secret, jwtConfig); // Garantindo que quem lê o token saiba que é verdade esse bilete
```

→ **api-auth-validateToken.js: code para validar token** . Função que será usada como middleware para as nossas requisições, validando todas as rotas em que nós solicitarmos autenticação.

→ **Integrar como etapa no endpoint**
const validateJWT = require('./auth/validateJWT');
// middleware antes do controller
apiRoutes.get('/api/posts', validateJWT, routes.getPosts);

→ **Testar no Postman**



Para pegar informações e elementos de sessão que sejam apenas do user → Middleware de autenticação recupera o usuário do banco de dados e o coloca no req. Esse req é usado antes de chamar controller para assim pegar os posts apenas do user.

Dicas diversas (aula ao vivo):

Dois tipos de criptografia: simétrica e asimétrica.
- Simétrica: combinar senha (ou seja um segredo) para de um lado criptografar e mandar, do outro descriptografar e ler.
- No assimétrico, é com chave pública que criptografa, e com chave privada que descriptografa, ou vice versa (o que uma fecha, outra abre). Ambas chaves da mesma ponta, é como uma assinatura pessoal. Integridade da mensagem em ambos casos.

_

CUIDADO – Diversas medidas de segurança
→ **Token é público**, nunca gerir a partir de senhas, CPF, cartões de credito etc.

// tirar password antes de gerir token
// fazer function e chamar: IIFE (Immediately Invoked Function Expression)

→ Criar token conforme LGPD.
→ Dar poucas informações nas res para tokens errados.

_

Validar CPF:
123,456,789-ab
1*1+2*2+3*3+4*4+5*5+6*6+7*7+8*8+9*9 = a
1*9+2*8+3*7+4*6+5*5+6*4+7*3+8*2+9*1 = b

---

2) NodeJs – Upload de arquivos com Multer

Objetivo da aula: criar APIs que suportam upload de arquivos.

Code de preparo

Implementar uma API que recebe um arquivo e o armazena em uma pasta específica dentro do próprio servidor.

Criar pasta para armazenar code
npm init -y
npm install express multer
Criar endpoint com req e res simples (ping – pong)

Multer

Criar uma rota para receber nossos arquivos.