# FICHA TRABALHO 4

| | | | |
|---|---|---|---|
| MODALIDADE: | Aprendizagem + | Não aplicável | |
| CURSO: | Técnico/a especialista em Gestão de Informação e Ciência dos Dados | | |
| UFCD: | Segurança no desenvolvimento de software | CÓDIGO UFCD: | 10795 |
| FORMADOR/A: | Bruno Silva | DATA: | |

## *Parte 1 – JWT*

Na era digital, a autenticação e a segurança de dados são fundamentais para o desenvolvimento de aplicações. Uma das tecnologias mais populares para garantir a transmissão segura de informações entre as várias partes é o JSON Web Token (JWT).

JWT é a sigla para JSON Web Token e trata-se de um formato compacto e autossuficiente para representar informações entre duas partes de maneira segura como um objeto ***JSON***. É frequentemente usado para autenticação e troca de informações entre um servidor e um cliente, como uma alternativa às sessões tradicionais.

***Exemplo do que é um objeto JSON:***

```
{
  "name": "Alice",
  "age": 20,
  "email": "alice@email.com"
}
```

***Uma chave JWT é constituída por 3 partes separadas por pontos (.):***

1. Header (Cabeçalho)
2. Payload (Carga)
3. Signature (Assinatura)

***Header (Cabeçalho)***

O cabeçalho tipicamente consiste em duas partes: ***o tipo do token***, que é JWT, e o ***algoritmo de assinatura***, como HMAC SHA256, RSA, entre outros tipos.

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

Este JSON é então codificado em Base64Url para formar a ***primeira parte*** do JWT.

***Payload (Carga)***

É o corpo do JWT, que contém as reivindicações, que são informações sobre uma entidade (geralmente o utilizador) e metadados adicionais.

As informações adicionais podem ser compostos por 3 tipos: registered, public, e private.

- ***Registered claims:*** conjunto de dados predefinidas recomendadas, para fornecer um conjunto de informações úteis e interoperáveis. Exemplos:

  - **Issuer (iss):** Identifica o emissor do token;
  - **Subject (sub):** Identifica o assunto do token (geralmente o ID do utilizador);
  - **Audience (aud):** Define a audiência para a qual o token é destinado;
  - **Expiration Time (exp):** Define o tempo de expiração do token;
  - **Not Before (nbf):** Especifica a data a partir da qual o token é válido;
  - **Issued At (iat):** Indica o momento em que o token foi emitido;
  - **JWT ID (jti):** Fornece um identificador exclusivo para o token;

***Exemplo:***

```
{
  "iss": "http://www.mysite.com/",
  "sub": "1234567890",
  "exp": "1239965891",
  "name": "John Doe",
  "admin": true
}
```

Este JSON é também codificado em Base64Url para formar a ***segunda parte*** do JWT.

***Signature (Assinatura)***

É a assinatura do JWT, que adiciona segurança a esse sistema de autenticação. A assinatura garante a possibilidade de verificar se o token foi adulterado durante a transmissão. Esta é criada a partir do header e do payload usando uma chave secreta.

Para criar a assinatura, precisamos de pegar o **header codificado** em Base64Url, o **payload codificado** em Base64Url, **qual a palavra-chave secreta e o algoritmo especificado no header**. Por exemplo, usando HMAC SHA256:

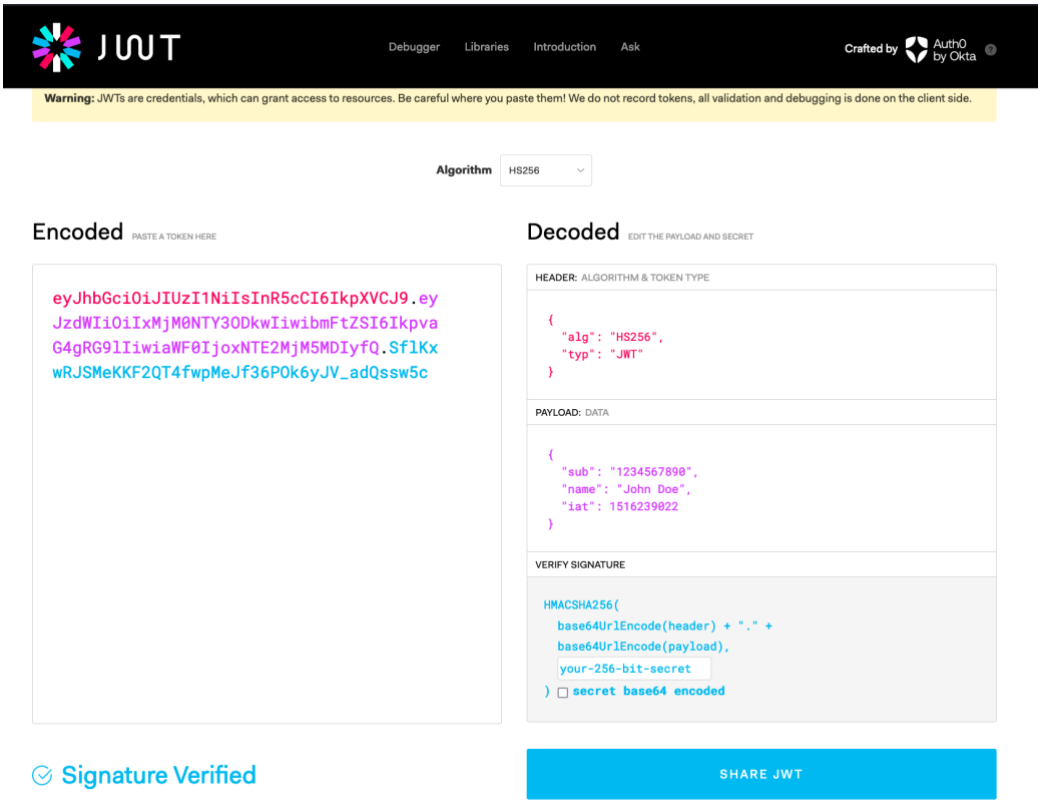
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

***Juntando todos os 3 elementos (header, payload e signature), vamos ter algo parecido com a seguinte imagem:***




Se desejar ver como estes elementos JWT funcionam em tempo real e colocar esses conceitos em prática, podemos utilizar a plataforma online jwt.io Debugger para decodificar, verificar e gerar JWTs.

***Hiperligação:*** https://jwt.io/#debugger-io



## *Parte 2 – Implementação de JWT Tokens em Python*

Passo 1 - Antes de mais, vamos começar por instalar o módulo do **pyjwt**. Como tal, vamos abrir o terminal do editor Visual Studio e correr a seguinte instrução:

```
pip install pyjwt
```

De seguida, vamos criar um novo ficheiro com o nome "tokens_jwt.py". Neste ficheiro, vamos organizar o nosso código por 4 partes:

- Parte 1 – **variáveis globais** (que vão indicar qual o algoritmo e senha secreta);
- Parte 2 - método para **criar** tokens jwt;
- Parte 3 - método para **validar** tokens jwt;
- Parte 4 - **método principal onde invocamos os métodos dos passos 2 e 3**;

Antes disso, vamos começar por importar os módulos necessários, mais especificamente:

- Módulo do jwt (proveniente da instalação do módulo pyjwt);
- Módulo do datetime (data e tempo, e importar 3 componentes essenciais: data, tempo e fuso horário)

```
1   import jwt
2   from datetime import datetime, timedelta, timezone
```

***Parte 1 - Variáveis globais (que vão indicar qual o algoritmo e senha secreta)***

```
# Chave secreta (informação super importante :) )
secret_key = 'supersecretkey'

# Algoritmo de codificação
algorithm = 'HS256'
```

***Parte 2 - Método para criar Tokens jwt***

```
#metodo para criar token JWT
def create_jwt_token(payload):
    # codificar os dados que recebemos, mais especificamente,
    # 1 - as informações que recebmos do payload
    # 2 - qual a chave secreta
    # 3 - qual o algoritmo a utilizar
    token = jwt.encode(payload, secret_key, algorithm=algorithm)

    #retorna a token gerada
    return token
```

### *Parte 3 - Método para validar Tokens jwt*

```python
#metodo para validar token JWT existentes
def validate_jwt_token(token_to_validate):
    try:
        # descodificar os dados de um token
        decoded_payload = jwt.decode(token_to_validate, secret_key, algorithms=[algorithm])
        #se tudo correr bem, retorna a token descodificada
        return decoded_payload

    #caso a assinatura já tenha o tempo expirado
    except jwt.ExpiredSignatureError:
        print("Token expirado.")
    #caso a assintura seja inválida
    except jwt.InvalidTokenError:
        print("Token Invalido. Acesso negado.")

    #retorna nada, pois não conseguiu fazer a operação do try
    return None
```

### *Parte 4 - Método principal onde passamos as informações (programa principal)*

```python
#metodo principal
def main():
    #codificar a informação para token (corpo da informação)
    payload = {
        'iss': 'https://www.teste.pt/',
        'user_id': 123,
        'username': 'teste',
        'role': 'admin',
        'exp': datetime.now() + timedelta(minutes=1) # Token a expirar em 1 minuto
        #(pode ser microsegundos (microseconds), segundos (seconds), horas (hours) ou semanas (weeks) )
    }

    #invocar metodo criar tokens e levar a informação do payload
    #depois de processar todo o metodo recebe o resultado com a token
    token = create_jwt_token(payload)
    print("JWT TOKEN: ", token)

    #validar e descodificar token
    decoded_payload = validate_jwt_token(token)

    # Se descodificar com sucesso, mostra os dados
    if decoded_payload:
        print("Decoded Payload:", decoded_payload)


#invocar o metodo principal
main()
```

### *Resultado da execução do programa:*

```
JWT TOKEN:  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL3d3dy50ZXN0ZS5wdC8iLCJ1c2VyX2lkIjoxMjMsInVzZXJuYW1lIjoidGVzdGUiLCJyb2xlIjoiYWRtaW4iLCJleHAiOjE3MjA1MzQ5NjR9.TxVlBIFGybyb2bPe0Jez4OIJ40P2fyUzbVCeokqOfCo
Decoded Payload: {'iss': 'https://www.teste.pt/', 'user_id': 123, 'username': 'teste', 'role': 'admin', 'exp': 1720534964}
```