UTILIZANDO JSON WEB TOKEN – JWT

EM PROJETOS JAVA WEB

UPF – Análise e Desenvolvimento de Sistemas
Programação de Serviços Web
Prof. Jeangrei Veiga

Sumário

Requisitos

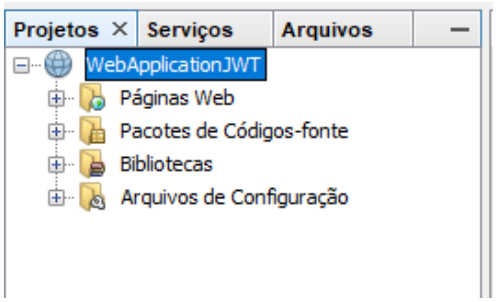
As ferramentas abaixo devem ser instaladas para uso do projeto.

- NetBeans 8.2
- Glassfish 4.0
  Servidor de aplicação Web
- Biblioteca - jjwt-0.9.1
- Biblioteca – Jackson (se utilizar glassfish 4.1 ou superior não é necessário incluir).

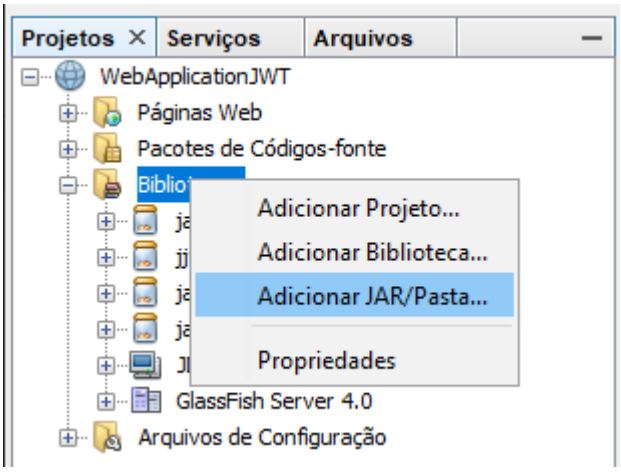
AULA 01

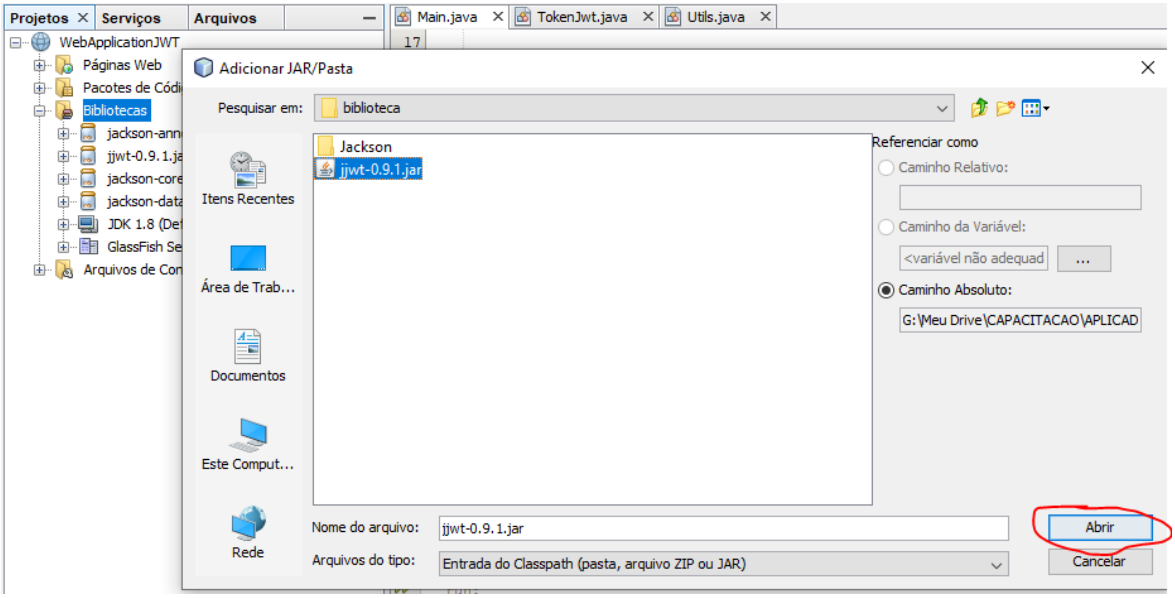
Meu Primeiro Projeto Web JWT (WebApplicationJWT)

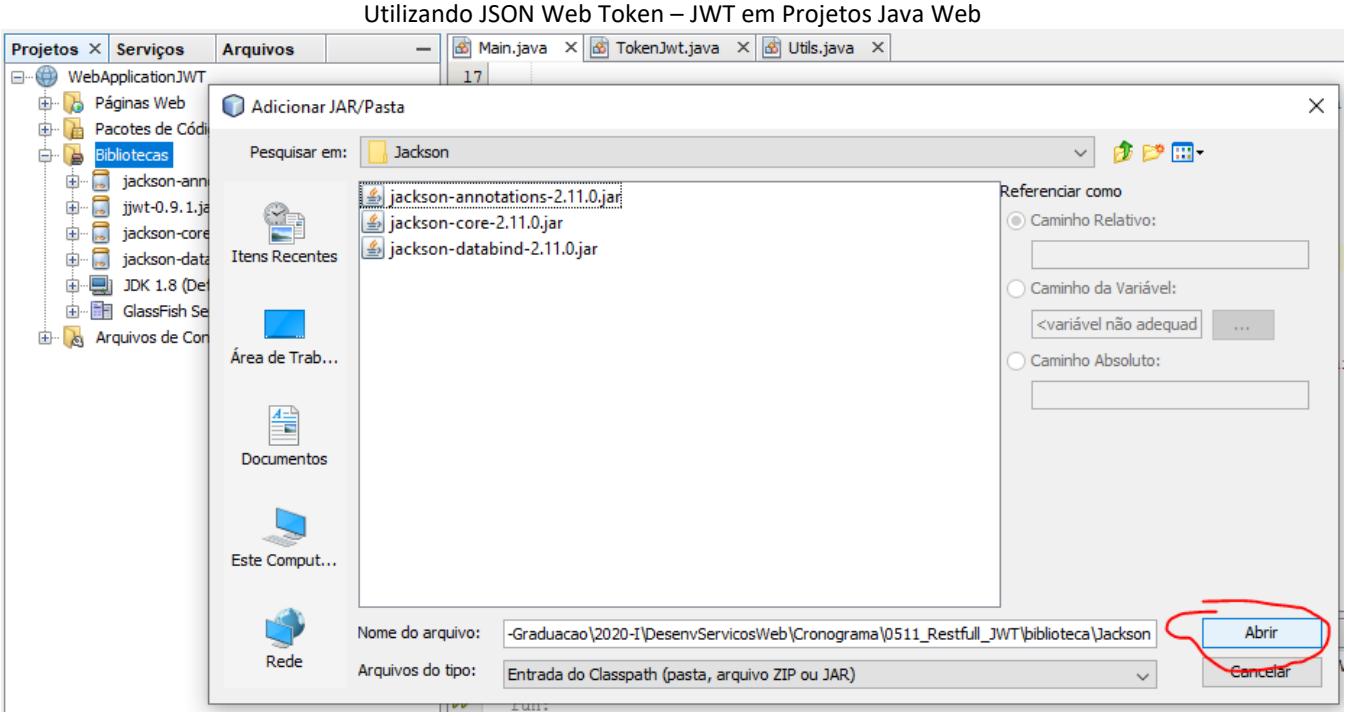
1. Criar o projeto Web.

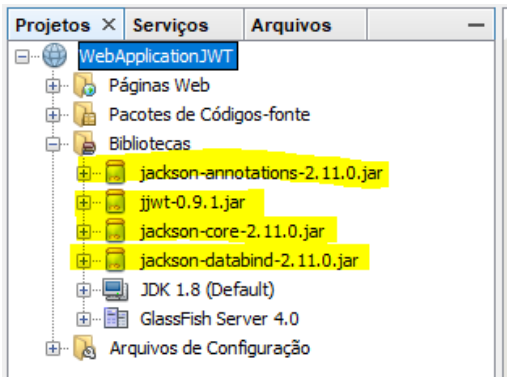


2. Adicionar biblioteca JJWT e Jackson(dependência implícita de JJWT) no projeto:

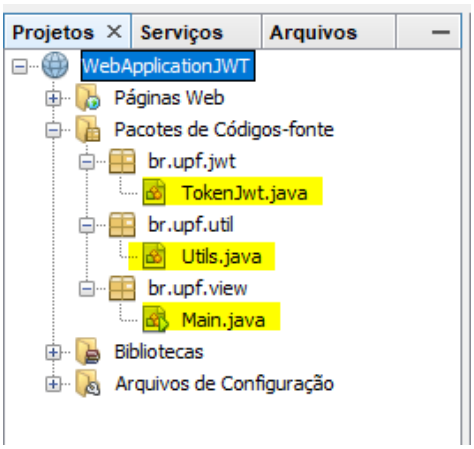








3. Estrutura de classes:



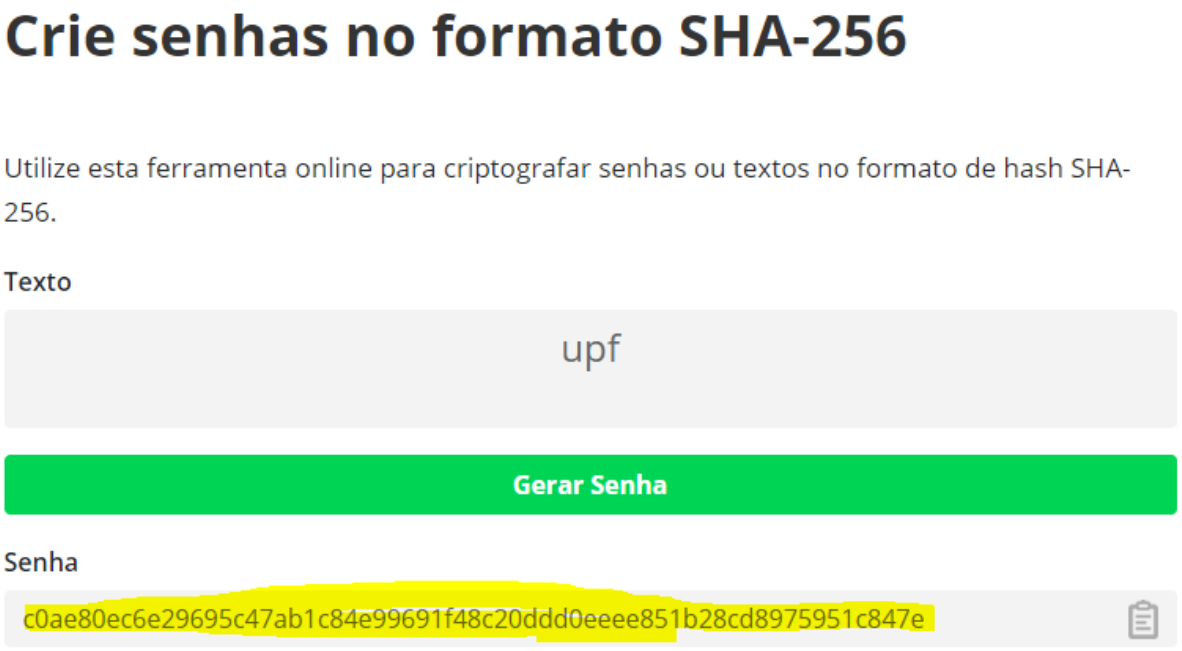
4. Editando a classe Utils.java...

```java
import javax.crypto.spec.SecretKeySpec;
import java.security.Key;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.util.Date;

public class Utils {

    /**
     * Gera uma chave a ser utilizada na assinatura do token.
     * @return
     */
    public static Key gerarChave() {
        /*A string "upf"
        após a execução de SHA-256 "C0AE80EC6E29695C47AB1C84E99691F48C20DDD0EEEE851B28CD8975951C847E"
        e a execução de EncodeBase64.
        Esta string deve ser conhecida apenas por aplicações que precisam gerar um token, validar um token ou
        consumir as informações nele presente.
        Esta senha não pode se tornar pública, pois poderá comprometer totalmente comprometida.
        */
        String keyString = "QzBBRTgwRUM2RTI5NjklQzQ3QUIxQzg0RTk5NjkxRjQ4QzIwREREMEVFRUU4NTFCMjhDRDg5NzU5NTFDODQ3RQ==";
        Key key = new SecretKeySpec(keyString.getBytes(),
                0, keyString.getBytes().length, "HmacSHA256");
        return key;
    }

    /**
     * Representa a validade de um token em minutos.
     * @param tempoEmMinutos
     * @return
     */
    public static Date definirDataDeExpiracao(long tempoEmMinutos) {
        LocalDateTime localDateTime = LocalDateTime.now().plusMinutes(tempoEmMinutos);
        return Date.from(localDateTime.atZone(ZoneId.systemDefault()).toInstant());
    }
}
```

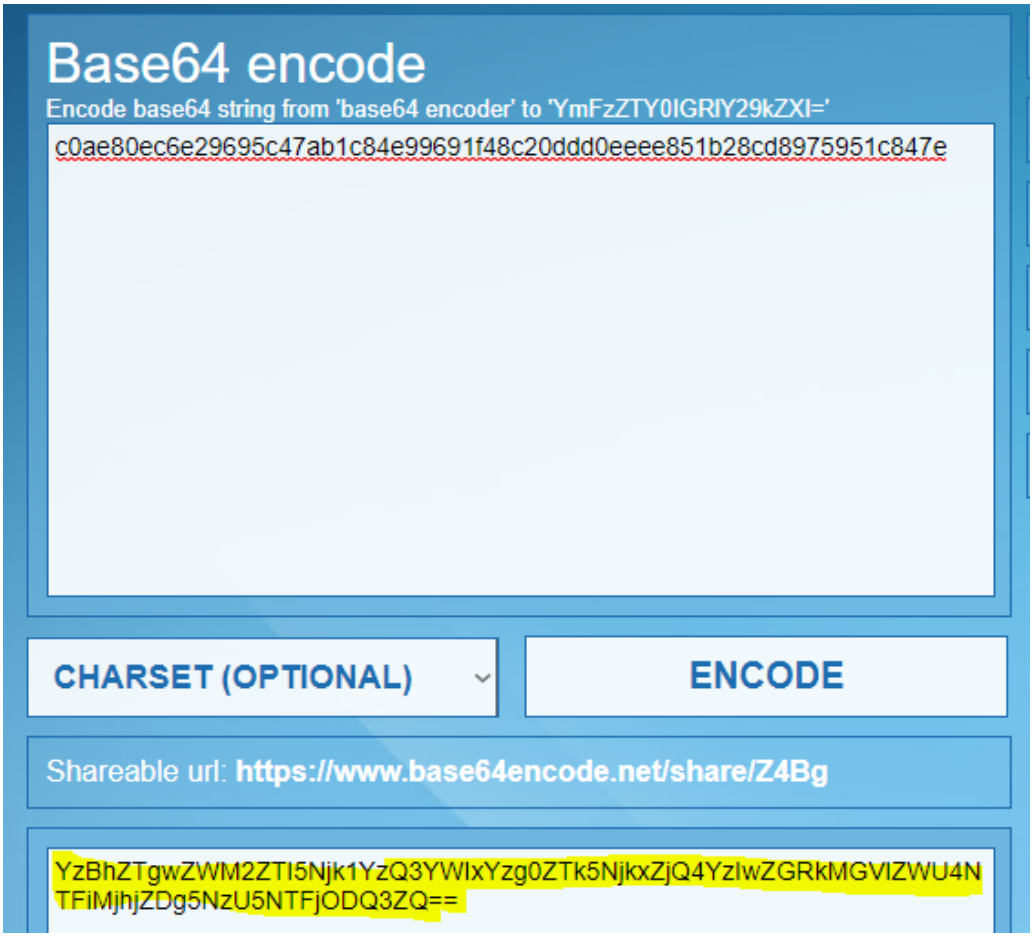
KeyString =
YzBhZTgwZWM2ZTI5Njk1YzQ3YWIxYzg0ZTk5NjkxZjQ4YzIwZGRkMGVlZWU4NTFiMjhjZDg5NzU5NTFjODQ3ZQ==

Obs: o conteúdo da KeyString foi gerado externamente, utilizando:

1. O site: https://www.geradordesenha.com/gerador-online-de-senhas-hash-sha256/ foi utilizado o texto "upf" para gerar a senha, onde foi obtivo o valor (c0ae80ec6e29695c47ab1c84e99691f48c20ddd0eeee851b28cd8975951c847e)



2. O site: https://www.base64encode.net para pegar o resultado gerado no item 1 e aplicar Encoding Base64 e obter o valor (YzBhZTgwZWM2ZTI5Njk1YzQ3YWIxYzg0ZTk5NjkxZjQ4YzIwZGRkMGVlZWU4NTFiMjhjZDg5NzU5NTFjODQ3ZQ==), o qual foi utilizado para compor a variável keyString.



5. Editando a classe TokenJwt.java...

```java
import io.jsonwebtoken.Claims;
import io.jsonwebtoken.Jws;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

import java.security.Key;
import java.util.Date;

public class TokenJwt {

    private Key chave;

    private String jwt; //uma string que vai armazenar o JWT

    /**
     * Construtor que recebe a cahve como parâmetro, pois sem este não é possível
     * gerar o token.
     * @param chave
     */
    public TokenJwt(Key chave) {
        this.chave = chave;
    }


    /**
     * Utilizado para validar o token
     * @return
     */
    public boolean tokenValido() {
        try {
            Jwts.parser().setSigningKey(chave).parseClaimsJws(jwt);
            return true;
        } catch (Exception e) {
            return false;
        }
    }
```

```
39
40      /**
41       * Utilizado para recuperar o assunto do Token
42       * @return
43       */
44      public String recuperarSubjectDoToken() {
45          Jws<Claims> claimsJws = Jwts.parser().setSigningKey(chave).parseClaimsJws(jwt);
46          return claimsJws.getBody().getSubject();
47      }
48
49      /**
50       * Utilizado para recuperar quem é o emissor do token
51       * @return
52       */
53      public String recuperarIssuerDoToken() {
54          Jws<Claims> claimsJws = Jwts.parser().setSigningKey(chave).parseClaimsJws(jwt);
55          return claimsJws.getBody().getIssuer();
56      }
57
58          /**
59       * Método responsável por gerar o token
60       * @param nomeUsuario
61       * @param dataExpiracao
62       * @return
63       */
64      public String gerarToken(String nomeUsuario, Date dataExpiracao) {
65          jwt = Jwts.builder()
66                  .setHeaderParam("typ","JWT") //definindo parâmetros do cabeçalho
67                  .setSubject(nomeUsuario) //assunto do token
68                  .setIssuer("upf") //quem é o emissor do token
69                  .setIssuedAt(new Date()) //data de criação
70                  .setExpiration(dataExpiracao) //data de expiração do token
71                  .signWith(SignatureAlgorithm.HS256, chave) //assinar o token
72                  .compact(); //contruir o JWT
73          return jwt;
74      }
75  }
```

6. Editando a classe Main.java...

```java
package br.upf.view;

import br.upf.jwt.TokenJwt;
import br.upf.util.Utils;
import java.util.Date;

public class Main {
    public static void main(String args[]) throws InterruptedException{
        TokenJwt token = new TokenJwt(Utils.gerarChave());

        System.out.println("*** Gerando o JSON Web Token ***");
        //definindo 1 minuto como tempo de expiração
        Date dataDeExpiracao = Utils.definirDataDeExpiracao(1L);
        String jwt = token.gerarToken("jeangrei", dataDeExpiracao);
        System.out.println("JWT : " + jwt);
        System.out.println("  ***  ==============   ***");

        System.out.println("**  Aguardando 00:01:01 minutos para testar validade do token ***");
        Thread.sleep(60001L);

        if(token.tokenValido()){
            System.out.println("--> Token válido!");
            System.out.println("Nome do usuário presente no token: " + token.recuperarSubjectDoToken());
            System.out.println("Emissor do token: " + token.recuperarIssuerDoToken());
        } else{
            System.out.println("----> Token INVÁLIDO!");
        }
    }
}
```

7. Testando a classe Main.java

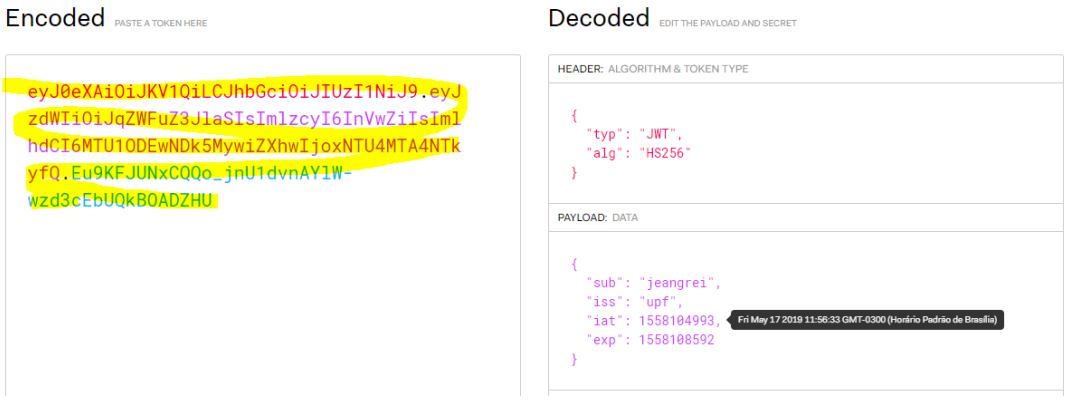
O teste abaixo realiza simulação onde a data de vencimento expira, por isso ocorre falha na validação do Token

```
Saída
Console do Depurador | WebApplicationJWT (run)
run:
*** Gerando o JSON Web Token ***
JWT : eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqZWFuZ3JlaSIsImlhdCI6MTU4OTIyNDUyMiwiZXhwIjoxNTg5MjI0NTgyfQ.3XsxWK-J3zX34S3YaLUMmPOGmVtKiINcc0_0bXn0A24
  ***  ==============   ***
**  Aguardando 00:01:01 minutos para testar validade do token ***
----> Token INVÁLIDO!
CONSTRUÍDO COM SUCESSO (tempo total: 1 minuto 2 segundos)
```

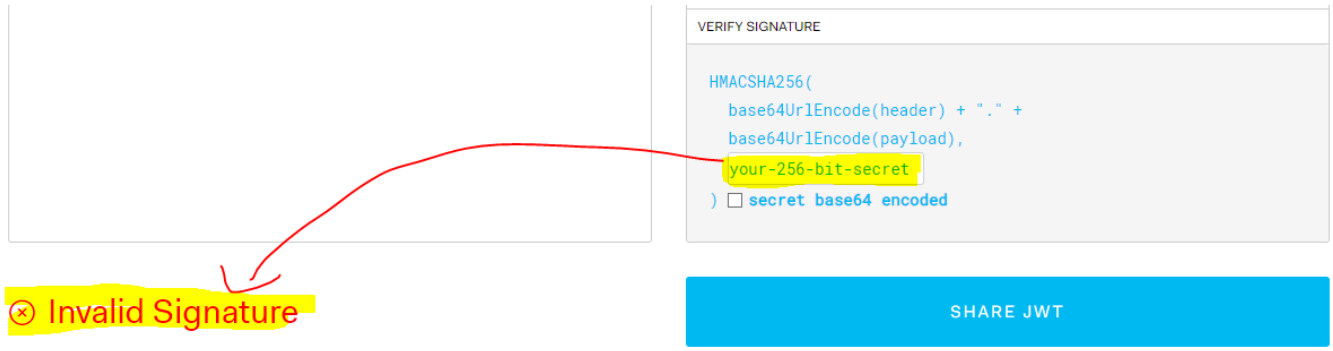
O teste abaixo realiza simulação onde a data de vencimento do token NÃO EXPIRA. Foi comentado a linha 23 e 24 da classe Main.java, onde define uma pausa na execução da Thread.

```
Saída
Console do Depurador | WebApplicationJWT (run) | WebApplicationJWT (run) #2
*** Gerando o JSON Web Token ***
JWT : eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqZWFuZ3JlaSIsImlhdCI6MTU4OTIyNDk3OSwiZXhwIjoxNTg5MjI1MDM4fQ.4Lhe_woQ4DPQwCM5EFRZWJkI2_tz-m2Whx7DsHfblpA
  ***  ==============   ***
--> Token válido!
Nome do usuário presente no token: jeangrei
Emissor do token: upf
CONSTRUÍDO COM SUCESSO (tempo total: 2 segundos)
```

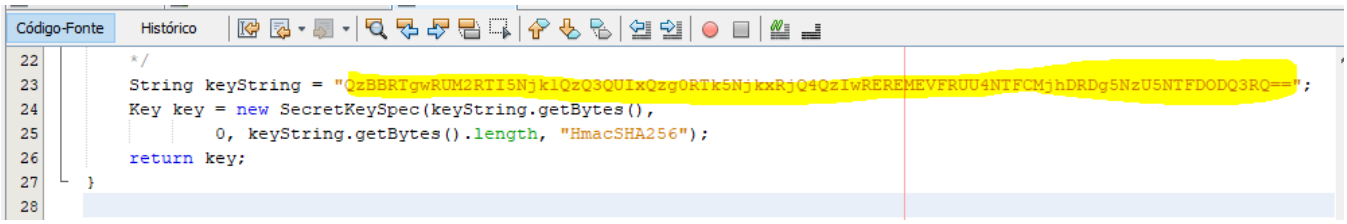
8. Validando o Token no site http://jwt.io



Assinatura é inválida, pois não bate com a assinatura que consta no campo de verificação:



Para validar a assinatura, basta passar a chave secreta que temos na classe Utils.java

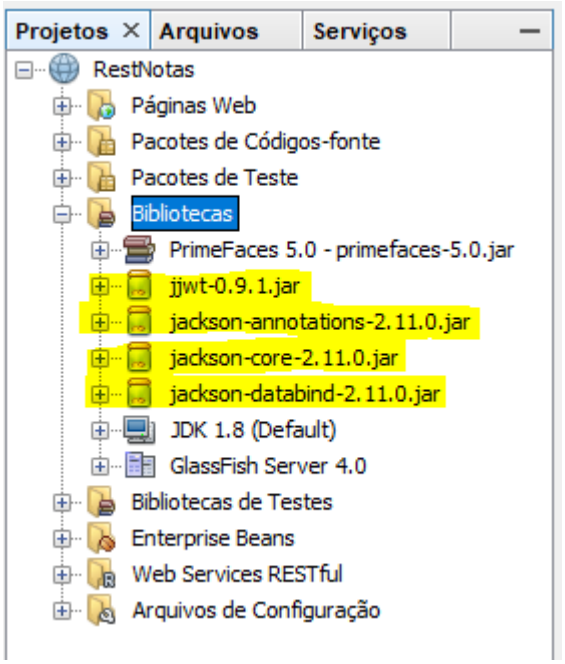


A verificação será realizada com sucesso...

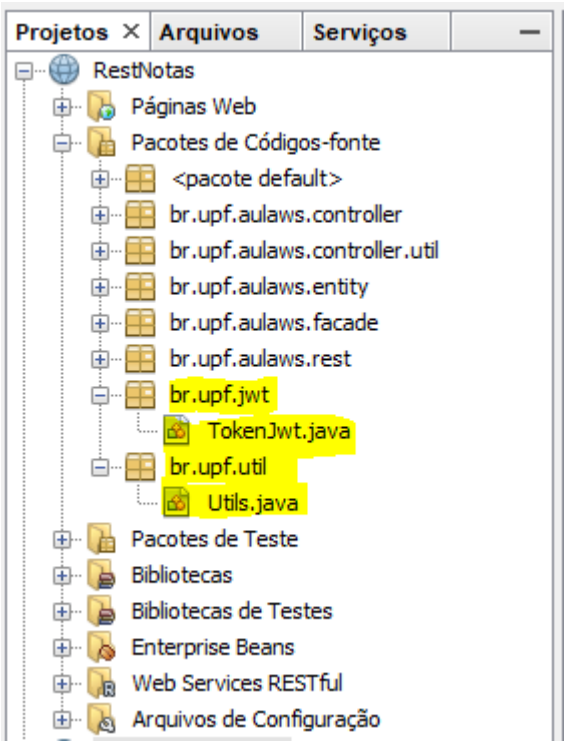
AULA 02

Implementar JWT no Projeto RestNotas

1. Abrir o projeto Web RestNotas e adicionar as bibliotecas do JWT e Jackson.



2. Importar no projeto a classe Utils.java e a classe TokenJWT.java criada anteriormente neste tutorial.



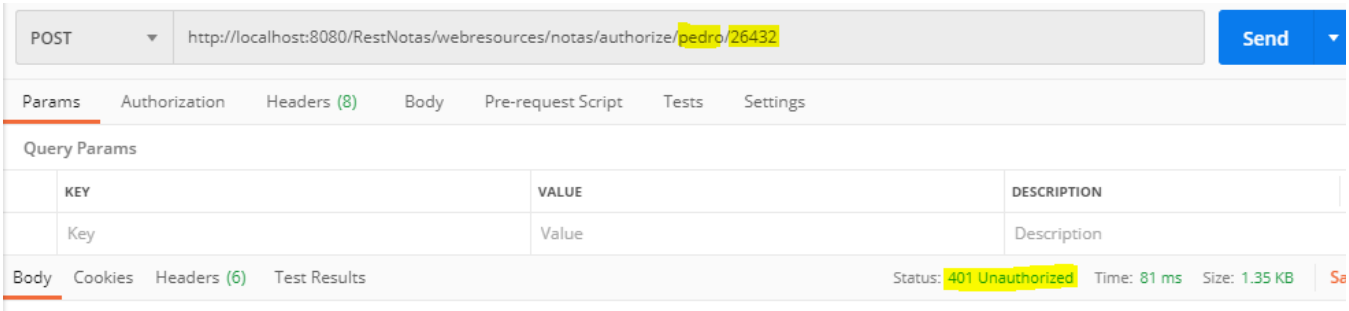
3. Na classe NotasService.java incluir o método que valida o usuário e senha e gera um token válido, recebendo parâmetros pela URI.

NotasService.java

```
153         * Método responsável por autenticar o usuário no sistema e gerar um token e
154         * retornar para o cliente.
155         *
156         * @param user
157         * @param password
158         * @return
159         */
160        @POST
161        @Path("/authorize/{user}/{password}")
162        @Produces(MediaType.APPLICATION_XML)
163        public Response getAuthorizeResource(@PathParam("user") String user,
164                @PathParam("password") String password) {
165            TokenJwt token = new TokenJwt(Utils.gerarChave());
166
167            if (user != null && password != null && user.equals("maria") && password.equals("123")) {
168                //definindo 10 minuto como tempo de expiração
169                Date dataDeExpiracao = Utils.definirDataDeExpiracao(10L);
170                String jwt = token.gerarToken("jeangrei", dataDeExpiracao);
171            //    System.out.println("JWT : " + jwt);
172                return Response.ok(jwt).build();
173            } else {
174                return Response.status(Response.Status.UNAUTHORIZED).build();
175            }
176        }
```

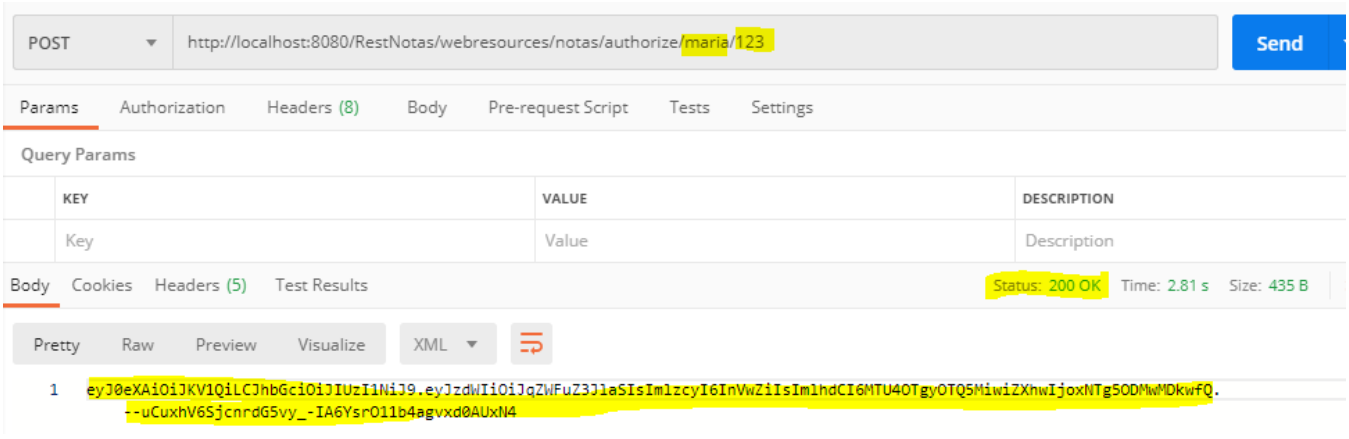
Testes com POSTMAN para verificar se o serviço esta disponível e funcional:

- Passando usuário e senha incorretos.



Status 401 UNAUTHORIZED

- Passando usuário e senha corretos e recebendo como retorno o token.



Chave gerada:

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqZWFuZ3JlaSIsImlzcyI6InVwZiIsImlhdCI6MTU4OTgyOTQ5MiwiZXhwIjoxNTg5ODMwMDkwfQ.--uCuxhV6SjcnrdG5vy_-IA6YsrO11b4agvxd0AUxN4

Status 200 e o retorno do token.

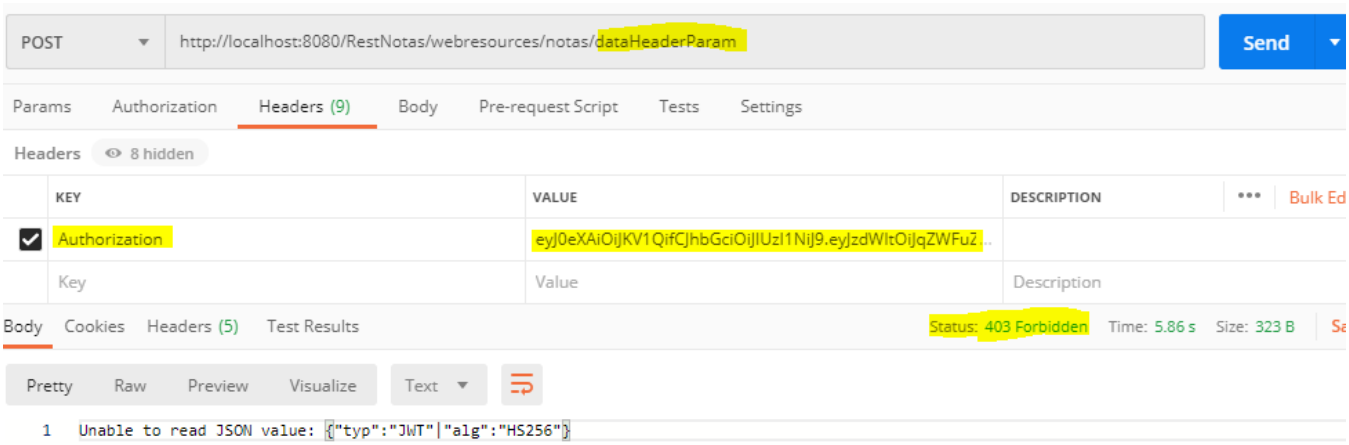
4. Na classe NotasService.java, incluir o método que recebe o token como parâmetro utilizando a HeaderParam.

NotasService.java

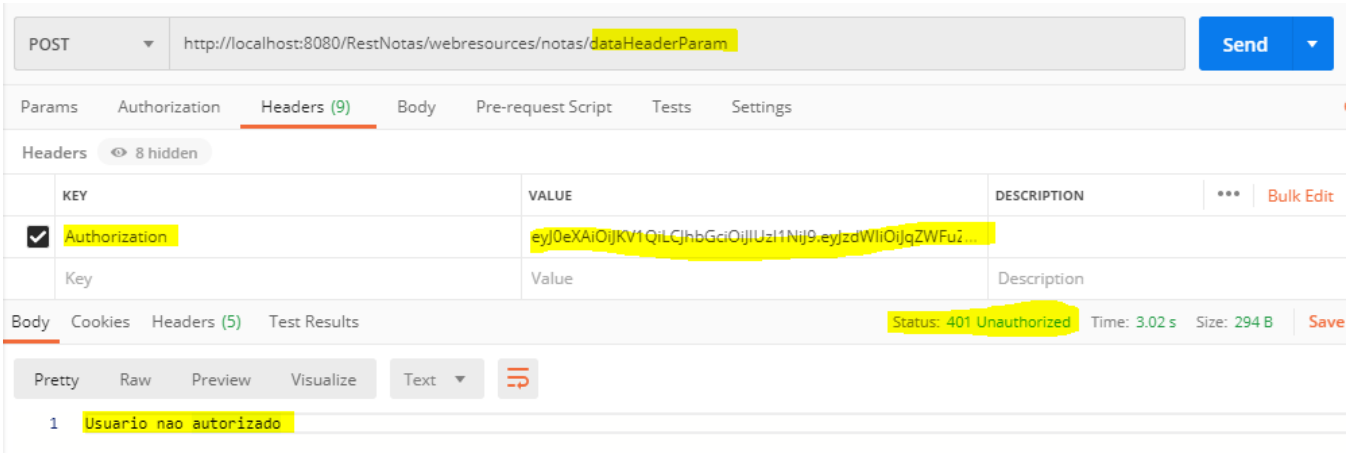
```
183     /**
184      * Método que utiliza a HeaderParam para receber parâmetros
185      * @param authHeader
186      * @return
187      */
188     @POST
189     @Path("/dataHeaderParam")
190     @Produces(MediaType.TEXT_PLAIN)
191     public Response getDataHeaderParam(@HeaderParam("authorization") String authHeader) {
192         Key key = Utils.gerarChave(); //chave de verificação...
193         if (authHeader != null) {
194             try {
195                //Gerando a Claims
196                 Jws<Claims> claims = Jwts.parser().setSigningKey(key).parseClaimsJws(authHeader);
197                 //verifica se a pessoa que consta no subject da Claims no momento da criação do  token
198                 //é a mesma que consta para comparação.
199                 if (!claims.getBody().getSubject().equals("pedro")) {
200                     //retorna status de usuário não autorizado
201                     return Response.status(401).entity("Usuario nao autorizado").build();
202                 }
203                 //retorna status de usuário autorizado
204                 return Response.ok("Usuário autorizado!").build();
205             } catch (Exception e) {
206                 return Response.status(403).entity(e.getMessage()).build();
207             }
208         } else {
209             return Response.status(403)
210                     .entity("Nenhum cabeçalho de autenticação presente!").build();
211         }
212     }
```

Testes com POSTMAN para verificar se o serviço esta disponível e funcional:

- Passando um Token inválido (com token válido, foi alterado um caracter do mesmo)



- Passando um Token válido e verificando se a pessoa que consta no subject da Claims no momento da criação do token (Jeangrei) é a mesma que consta para comparação (Pedro).



- Passando um Token válido e verificando se a pessoa que consta no subject da Claims no momento da criação do token (Jeangrei) é a mesma que consta para comparação (Jeangrei). É necessário alterar na classe NotasService.java linha 199 o nome da pessoa, no caso para "Jeangrei".

```
192     public Response getDataHeaderParam(@HeaderParam("authorization") String authHeader) {
193         Key key = Utils.gerarChave(); //chave de verificação...
194         if (authHeader != null) {
195             try {
196                 //Gerando a Claims
197                 Jws<Claims> claims = Jwts.parser().setSigningKey(key).parseClaimsJws(authHeader);
198                 //verifica se a pessoa que consta no subject da Claims no momento da criação do  token
199                 //é a mesma que consta para comparação.
200                 if (!claims.getBody().getSubject().equals("jeangrei")) {
```

Testando no Postman novamente...