Vamos construir um sistema de autenticação JWT completo em uma aplicação Quarkus. Este sistema incluirá a geração do token com a chave privada, a configuração do Quarkus para verificar o token e a comunicação do token na sessão do usuário.

Passo 1: Configuração do Projeto

1. Criar um Novo Projeto Quarkus:
   - Use o Quarkus CLI ou Maven para criar um novo projeto:

```
mvn io.quarkus:quarkus-maven-plugin:create \
  -DprojectGroupId=com.example \
  -DprojectArtifactId=jwt-auth \
  -DclassName="com.example.GreetingResource" \
  -Dpath="/hello"
```

2. Adicionar Extensões Necessárias:
   - Adicione as extensões `smallrye-jwt` e `smallrye-jwt-build`:

```
./mvnw quarkus:add-extension -Dextensions="smallrye-jwt,smallrye-jwt-build"
```

Passo 2: Gerar o Par de Chaves

1. Gerar a Chave Privada:
   - No terminal, execute:

```
openssl genpkey -algorithm RSA -out privateKey.pem -pkeyopt rsa_keygen_bits:2048
```

2. Extrair a Chave Pública:
   - No terminal, execute:

```
openssl rsa -pubout -in privateKey.pem -out publicKey.pem
```

3. Adicionar a Chave Pública ao Projeto:
   - Coloque o arquivo `publicKey.pem` em `src/main/resources/META-INF/resources`.

Passo 3: Configurar o Quarkus

1. **Configurar o Arquivo** `application.properties`:
   - Adicione as seguintes propriedades:

```
mp.jwt.verify.publickey.location=META-INF/resources/publicKey.pem
mp.jwt.verify.issuer=https://your-issuer.com
```

Passo 4: Gerar Tokens JWT

1. Criar um Endpoint para Gerar Tokens:
   - Crie uma classe `TokenService` para gerar tokens:

Java

```
package com.example;

import io.smallrye.jwt.build.Jwt;
import javax.enterprise.context.ApplicationScoped;
import java.util.HashSet;
import java.util.Set;

@ApplicationScoped
public class TokenService {

    public String generateToken(String username) {
        Set<String> roles = new HashSet<>();
        roles.add("user");

        return Jwt.issuer("https://your-issuer.com")
                  .upn(username)
                  .groups(roles)
                  .sign();
    }
}
```

Código gerado por IA. Examine e use com cuidado. Mais informações em perguntas frequentes.

2. Criar um Endpoint para Login:
   - Crie um endpoint para autenticar o usuário e retornar o token:

Java

```
package com.example;

import javax.inject.Inject;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("/auth")
public class AuthResource {

    @Inject
    TokenService tokenService;

    @POST
    @Path("/login")
    public Response login(UserCredentials credentials) {
        // Valide as credenciais do usuário (exemplo simplificado)
```

```
        if ("user".equals(credentials.username) &&
"password".equals(credentials.password)) {
            String token =
tokenService.generateToken(credentials.username);
            return Response.ok(token).build();
        }
        return Response.status(Response.Status.UNAUTHORIZED).build();
    }
}

class UserCredentials {
    public String username;
    public String password;
}
```

Código gerado por IA. Examine e use com cuidado. Mais informações em perguntas frequentes.

Passo 5: Proteger Endpoints

1. **Criar Endpoints Seguros**:
   - Use anotações como `@RolesAllowed` para proteger seus endpoints:

Java

```
package com.example;

import javax.annotation.security.RolesAllowed;
import javax.ws.rs.GET;
import javax.ws.rs.Path;

@Path("/secure")
public class SecureResource {

    @GET
    @RolesAllowed("user")
    public String secureEndpoint() {
        return "This is a secure endpoint";
    }
}
```

Código gerado por IA. Examine e use com cuidado. Mais informações em perguntas frequentes.

Passo 6: Comunicação do Token na Sessão do Usuário

1. Enviar o Token no Cabeçalho `Authorization`:
   - Quando o cliente (por exemplo, um frontend) receber o token JWT, ele deve incluí-lo no cabeçalho `Authorization` em todas as requisições subsequentes:

```
Authorization: Bearer <token>
```

2. Verificar o Token no Quarkus:

- O Quarkus automaticamente verifica o token JWT usando a chave pública configurada e valida as reivindicações contidas no token.

Conclusão

Com esses passos, você terá um sistema de autenticação JWT completo em uma aplicação Quarkus, desde a geração do token até a comunicação do token na sessão do usuário. Se precisar de mais alguma coisa ou tiver dúvidas específicas, estou aqui para ajudar! 😊