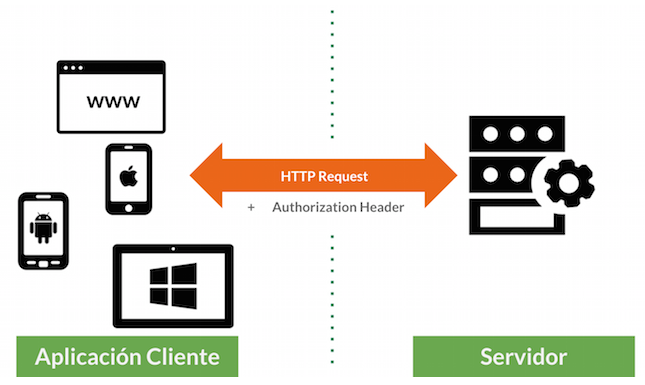
**JWT**

JSON Web Token es un estándar abierto basado en JSON para transmitir información con la identidad y claims de un usuario entre un cliente/servidor.



El JWT está conformado de 3 partes: header.payload.signature.

* Header: contiene la información del algoritmo con el cual se creó el JWT.

{ “alg”: “HS256”, “typ”:”JWT” }

* Payload: está compuesto por claims(privilegios) los cuales contienen información como el usuario para quien fue generado, su fecha de creación, fecha de expiración y cualquier otro atributo que debe ser agregado token.

{ “sub”:”BancaDigitalDesarrollo”, “iat”:1224140965, “exp”: 4561140965 }

* Signature está compuesto por el header y el payload, ambos cifrados en Base64 con una clave secreta.

tokenSinFirma = base64Encode(header) + “.” + base64Encode(payload)

firma = SHA256(key, tokenSinFirma)

token = tokenSinFirma + “.” + firma

El token obtenido sería el siguiente:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

Para la creación del JWT se sugiere el uso de llaves asimétricas, evitando con ello el uso de claves en duro dentro de un archivo properties tanto del lado del servidor como del cliente.

**Llaves asimétricas**

Las llaves deben ser creadas con openssl

Crea llave privada

genrsa -out privateApigeeKey.pem 1024

Crea llave publica

rsa -in privateApigeeKey.pem -pubout -out publicApigeeKey.pem

Cambio de formato

pkcs8 -topk8 -inform PEM -in privateApigeeKey.pem -out private\_key.pem -nocrypt

**TLS 2 Vías**

La información que se consulto para implementación del TLS de 2 vias en Spring fue la siguiente:

https://www.baeldung.com/x-509-authentication-in-spring-security

https://www.baeldung.com/java-ssl-handshake-failures

https://asardana.com/2016/10/11/spring-boot-mutual-authentication-2-way-ssltls/

<https://www.youtube.com/watch?v=bymtJYK8oto>

https://warlord3112.wordpress.com/2018/04/30/mutual-tls2-way-between-consul-and-spring-boot-application/

https://www.novatec-gmbh.de/en/blog/spring-boot-applications-tls-http2/

https://github.com/codependent/spring-boot-ssl-mutual-authentication

Adicional a esto, te comparto las instrucciones que se utilizaron para la creación de los Java Keystore en ambiente de desarrollo, ya que para ambiente productivo se utilizaron los que nos creo la gente de CA.

 Usa solo el apartado de **Certificados Autofirmados**