



Protegiendo tu API REST con JWT en aplicaciones .NET





- 1. Conceptos básicos de seguridad**
- 2. Cookies vs Tokens**
- 3. Definiciones de JWT**
- 4. Fundamentos de JWT**
- 5. Anatomía de JWT**
- 6. Estructura de un Token**
- 7. Nuestros amigos debuggers**
- 8. Ciclo de vida de un Token**
- 9. Librerías y vulnerabilidades**
- 10. Vamos a la acción**



UN REPASO A CONCEPTOS BASICOS

Autenticación

Recepción hotel

Login con password

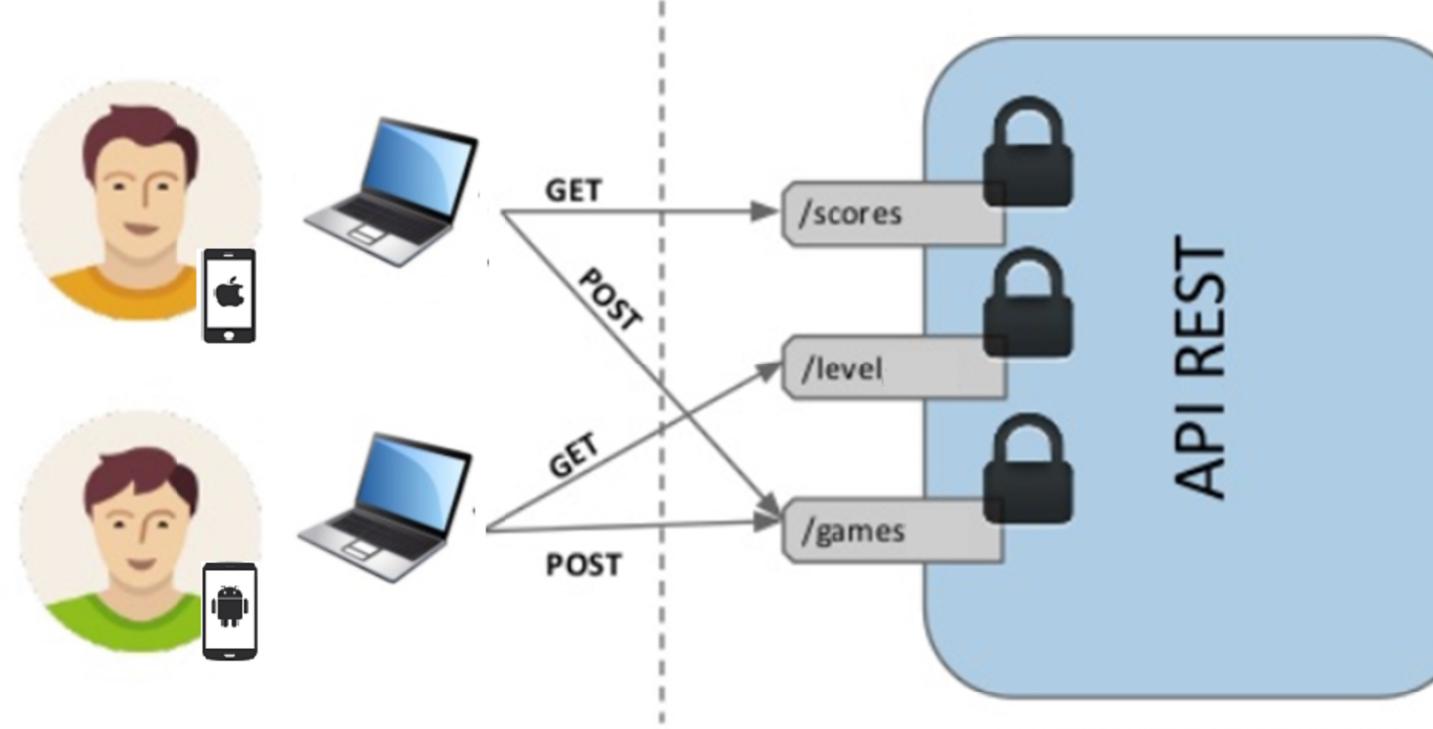
HTTP 401 Unauthorized

Autorización

Llave de la habitación

Permisos de acceso

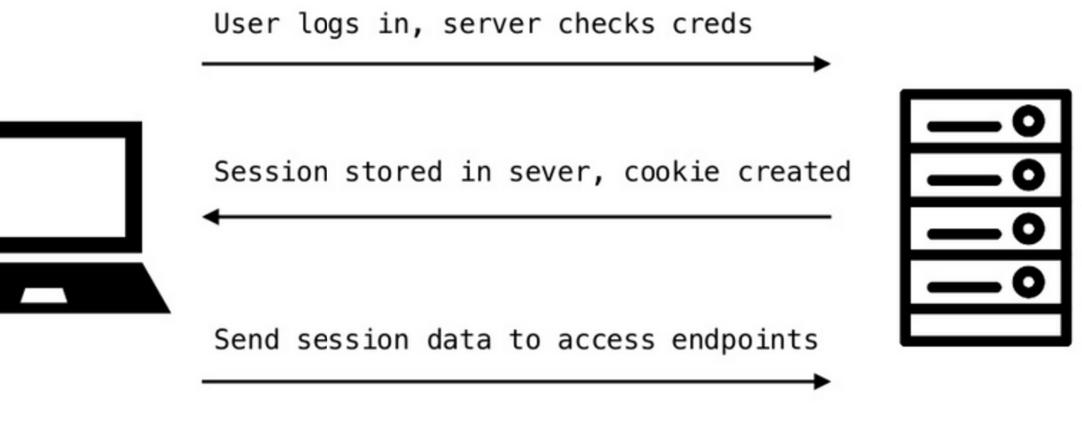
HTTP 403 Forbidden



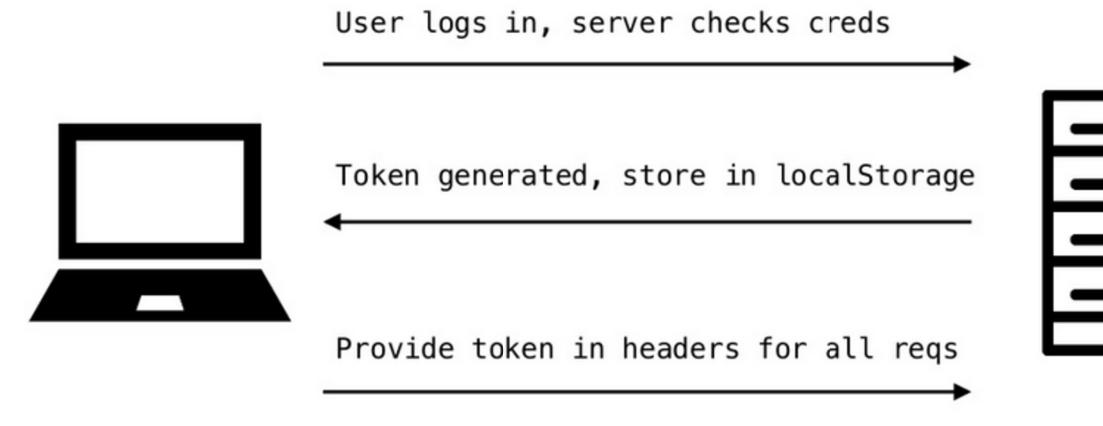


Cookies vs Tokens

Traditional Authentication Systems



Token-Based Authentication Systems



cookies: WebForms, asp.net mvc, etc

Sesión: Necesitan ser guardadas en el servidor, -Escalable

Datos: Contiene la sessionId, AUTH del usuario

Tokens: Api key, OAuth2, openID, SSO, etc.

Sesión: Se almacena en cada cliente, +Escalable

Datos: Contiene información del usuario



JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties.

JSON Web Token (JWT) Specification

Abstract

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

ON Web Token

JSON Web Token (abreviado **JWT**) es un **estándar abierto** basado en **JSON** propuesto por **IETF (RFC 7519)** para la creación de **tokens de acceso** que permiten la propagación de identidad y privilegios o *claims* en inglés. Por ejemplo, un servidor podría generar un token indicando que el usuario tiene privilegios de administrador y proporcionarlo al cliente. El cliente entonces podría utilizar el token para probar que está actuando como un administrador en el cliente o en otro sistema. El token está firmado por la clave del servidor, así que el cliente y el servidor son ambos capaz de verificar que el token es legítimo. Los JSON Web Tokens están diseñados para ser compactos, poder ser enviados en las URLs -*URL-safe*- y ser utilizados en escenarios de **Single Sign-On (SSO)**. Los privilegios de los JSON Web Tokens puede ser utilizados para propagar la identidad de usuarios como parte del proceso de **autenticación** entre un **proveedor de identidad** y un **proveedor de servicio**, o cualquiera otro tipo de privilegios requeridos por procesos empresariales.^{1 2 3 4}

https://es.wikipedia.org/wiki/JSON_Web_Token

<https://tools.ietf.org/html/rfc7519>

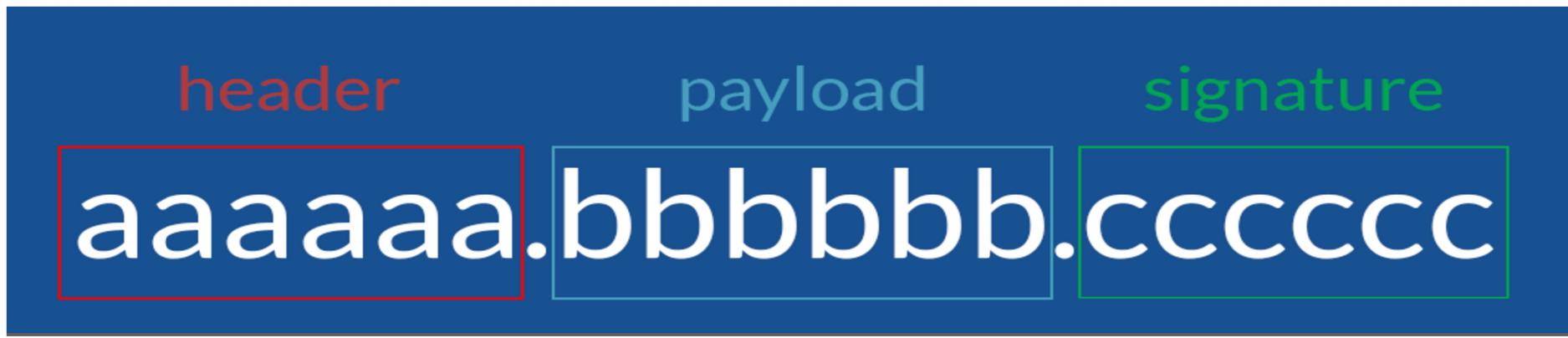


- Desktop, Mobile & Web Ready!!
 - Ligero: podemos codificar gran cantidad de datos y pasarlo como una cadena.
 - Self-container: Delegamos mantener el estado al cliente.
 - Stateless: Creamos servicios optimizados desacoplados del servidor .
 - Scalable: Los webserver pueden escalar sin problemas y aumentar en rendimiento.
 - La información es confiable porque está firmada digitalmente.
 - "*Authorization: Bearer token*" es la forma más común de enviarlo (existen otras).
 - ¡Nos olvidamos de cookies!
-
- JWT sirve para transmitir información de un usuario garantizando integridad de datos entre un cliente/servidor mediante una cadena de texto codificada en Base64.
-
- **Los JWT son mecanismos para transferir datos, no para asegurarlos**
 - **Los JWT son seguros cuando se utiliza conjuntamente con HTTPS**

RECORDAR: Siempre, debemos usar HTTPS entre el cliente/servidor para encriptar las peticiones.



ANATOMIA DEL TOKEN



HEADER: Indica el algoritmo y tipo de Token.

PAYLOAD: Datos de usuario/claims

SIGNATURE: la firma, para verificar que el token es válido.



EN MI LOCAL FUNCIONA

ESTRUCTURA DEL TOKEN (Claims)

Registered claims

jti	→ Id del token: String
iss	→ Issuer (emisor): StringOrUri
aud	→ Audiencia: StringOrUri
sub	→ Subject (tema): StringOrUri
iat	→ Cuándo se creó: NumericDate
exp	→ Cuándo expira: NumericDate
nbf	→ Tiempo hasta validez: NumericDate

Claims: No son obligatorios.

Claims: Se recomienda seguir este formato.

Claims: No todas las librerías .NET los implementan.

jti: Es muy útil para usar Tokens de un solo uso y evitar ataques.

Especificación: <https://tools.ietf.org/html/rfc7519>



JWT – ESTRUCTURA DEL TOKEN

eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9 . eyJuYW1lIjoiSldUNElLCJ1c2VmdWwiOnRydWV9 . nOlg86mQIADPd24_FnflpkWpE74SSFxsMtcfSmlEjeA

Header

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

Payload

```
{  
  "name": "JWT4B",  
  "useful": true  
}
```

Signature

```
HMACSHA256  
(  
  base64UrlEncode(Header)  
  + ". "  
  + base64UrlEncode(Payload)  
)
```

HEADER: Algoritmo Hash HS256 y token JWT.

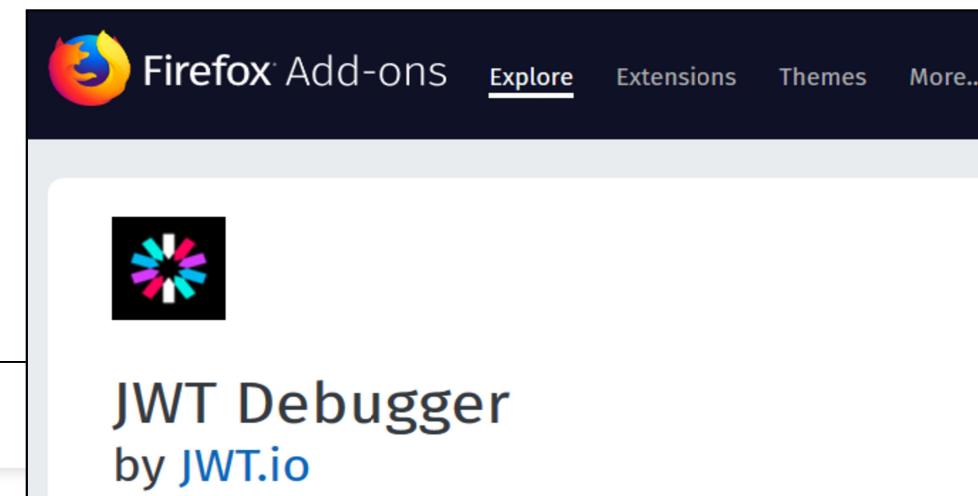
PAYLOAD: Datos de nombre usuario y lo que necesite nuestra API para validar la petición, recordar que nosotros generamos el token y podemos incluir todos los atributos que queramos.

SIGNATURE: Firma para la integridad del Token

Aquí lo importante es el “SECRET” con el que firmamos y que ahora explicaremos.



NUESTRO AMIGOS DEBUGGERS





Debugger

Encoded PASTE A TOKEN HERE
JhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzIi0iIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRlIwiYWRtaW4iOnRydWV9.TJVA950rM7E2cBab3MHrHcEfijoYZgeF0NFh7HgQ

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

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

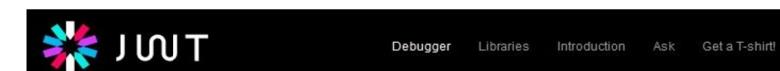
PAYOUT: DATA

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

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
)  Secret base64 encoded
```

Signature Verified



Debugger

Encoded PASTE A TOKEN HERE
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwidWRtaW4iOnRydWV9.TJVA950rM7E2cBab30RMHrHcEfijoYZgeF0NFh7HgQ

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

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

PAYOUT: DATA

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

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
   otro-secret
)  Secret base64 encoded
```

Invalid Signature

MO: Lo importante es el **SECRET** con el que firmamos el token y no debemos darlo a nadie.



JWT LIBRERIAS Y VULNERABILIDADES

<https://docs.microsoft.com/en-us/security-updates>

Security Advisories and Bulletins

10/11/2017 • 2 minutes to read • Contributors 

In this library you will find the following security documents that have been released by the Microsoft Security Response Center (MSRC). The MSRC investigates all reports of security vulnerabilities affecting Microsoft products and services, and releases these documents as part of the ongoing effort to help you manage security risks and help keep your systems protected.

- [Security Bulletins](#)
- [Security Bulletin Summaries](#)
- [Security Advisories](#)
- [Microsoft Vulnerability Research Advisories](#)
- [Acknowledgments](#)
- [Glossary](#)



JWT LIBRERIAS Y VULNERABILIDADES

Libraries for Token Signing/Verification

Warning: Critical vulnerabilities in JSON Web Token libraries with asymmetric keys. [Learn more ▾](#)

.NET	.NET	.NET (RT)
<input checked="" type="checkbox"/> Sign	<input checked="" type="checkbox"/> HS256	<input checked="" type="checkbox"/> HS256
<input checked="" type="checkbox"/> Verify	<input checked="" type="checkbox"/> HS384	<input checked="" type="checkbox"/> HS384
<input checked="" type="checkbox"/> iss check	<input checked="" type="checkbox"/> HS512	<input checked="" type="checkbox"/> HS512
<input checked="" type="checkbox"/> sub check	<input checked="" type="checkbox"/> RS256	<input checked="" type="checkbox"/> RS256
<input checked="" type="checkbox"/> aud check	<input checked="" type="checkbox"/> RS384	<input checked="" type="checkbox"/> RS384
<input checked="" type="checkbox"/> exp check	<input checked="" type="checkbox"/> RS512	<input checked="" type="checkbox"/> RS512
<input checked="" type="checkbox"/> nbf check	<input checked="" type="checkbox"/> ES256	<input checked="" type="checkbox"/> ES256
<input checked="" type="checkbox"/> iat check	<input checked="" type="checkbox"/> ES384	<input checked="" type="checkbox"/> ES384
<input checked="" type="checkbox"/> jti check	<input checked="" type="checkbox"/> ES512	<input checked="" type="checkbox"/> ES512

[Microsoft](#) 302 [View Repo](#)

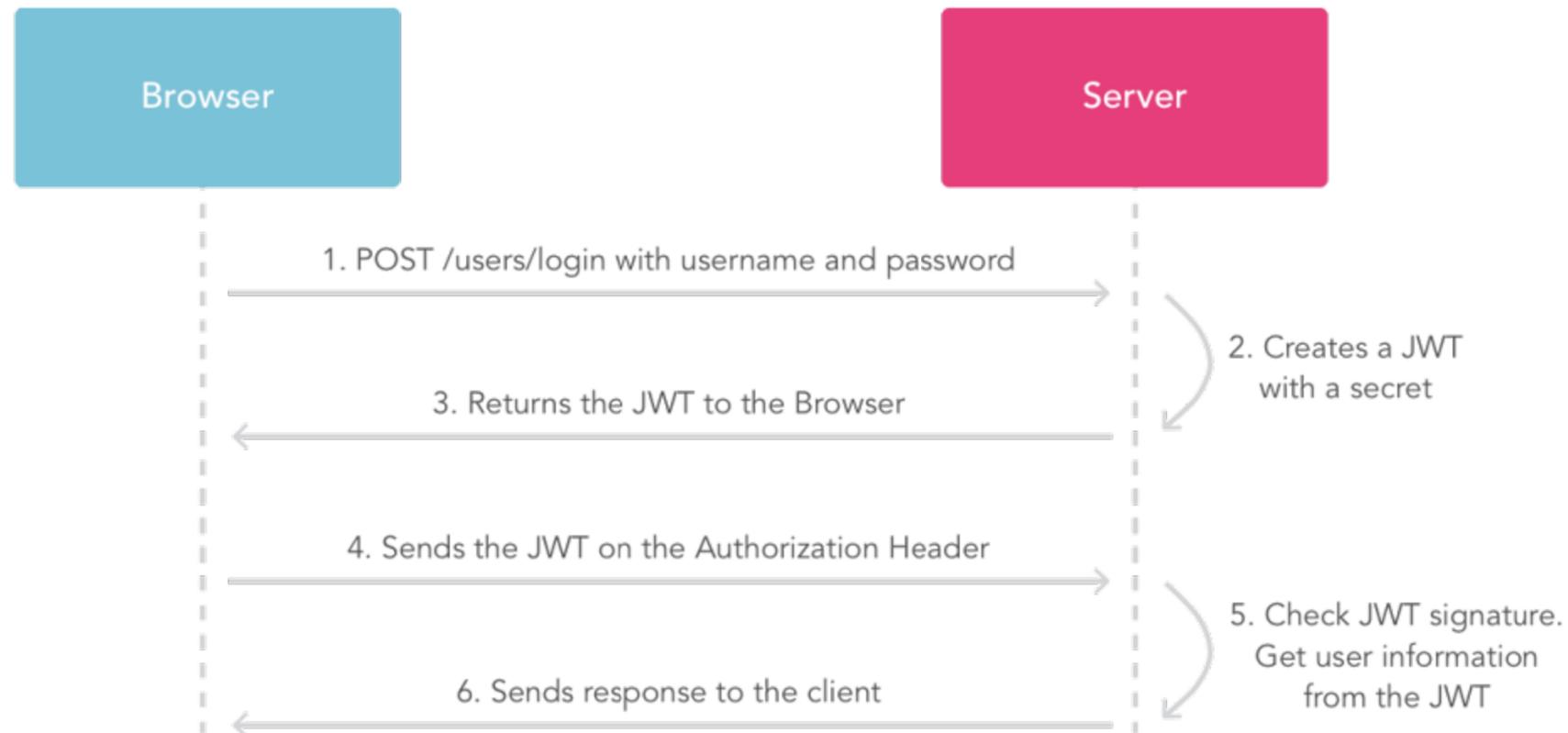
[jose-jwt](#) 378 [View Repo](#)

[jose-rt](#)

Vulnerabilities: <https://docs.microsoft.com/en-us/security-updates/securityadvisories/2017/3214296>

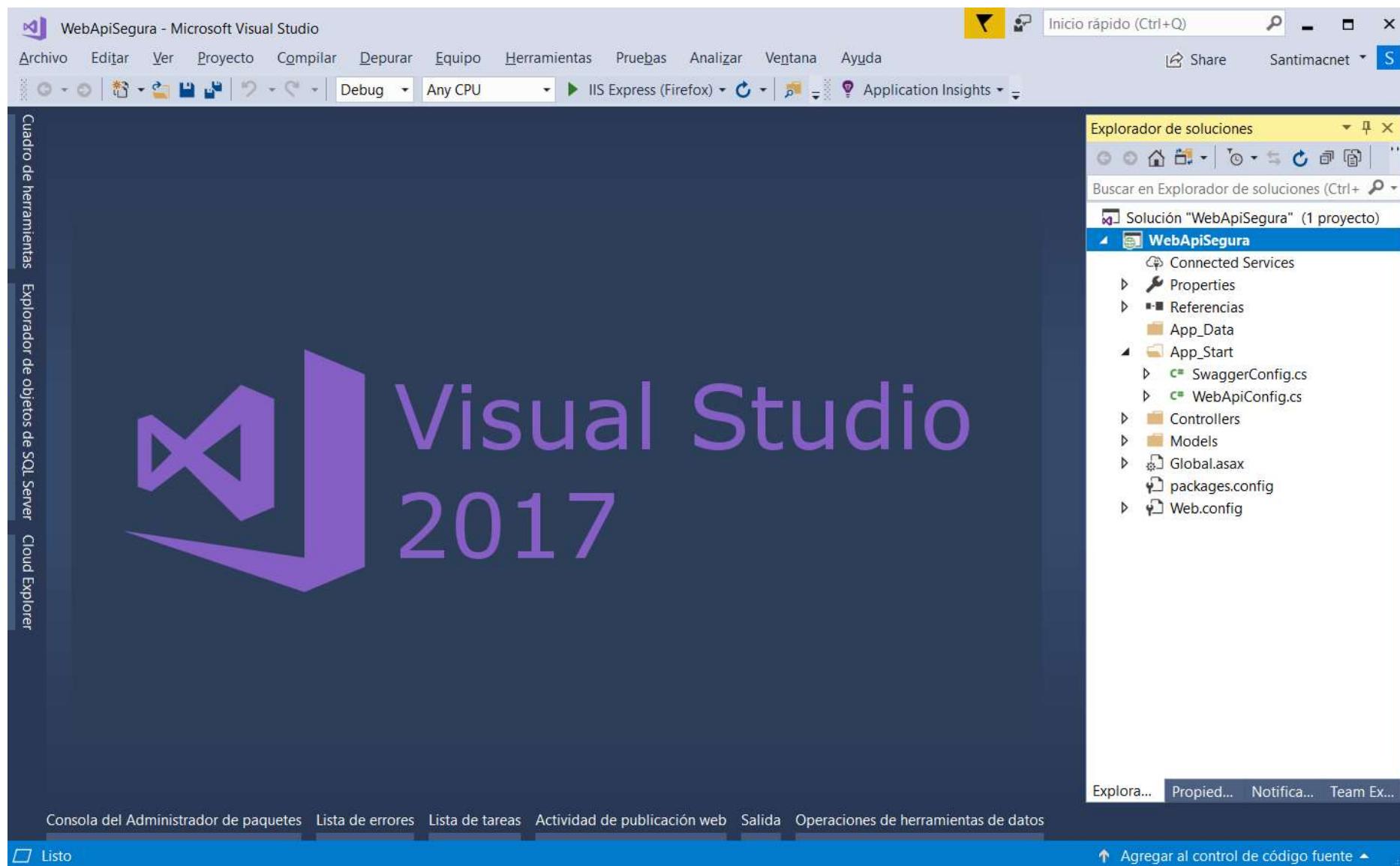


Ciclo de vida de un Token

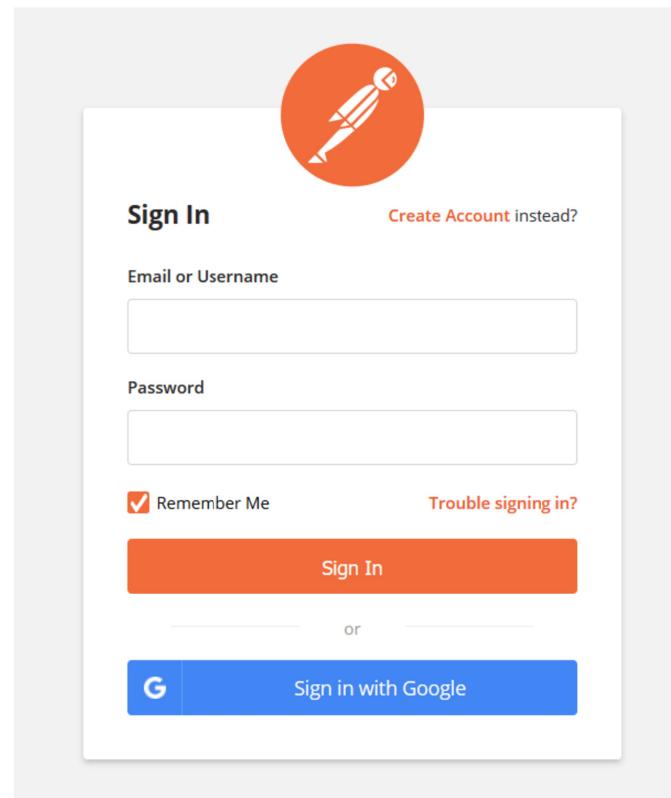




DEMO ASP.NET WEB API



DEBUGGERS



CUESTIONES CLAVE

- Que pasa cuando recibo el token en mi controlador
- Que pasa cuando caduca el token en mi aplicación
- Que pasa cuando realizamos logout en app/web
- Que pasa con mis token en devlocal y producción
- Que pasa si tengo varias API REST que usan JWT
- Quien y donde se gestionan los usuarios de mi API

PREGUNTAS



REFERENCIAS



<https://enmilocalfunciona.io/construyendo-una-web-api-rest-segura-con-json-web-token-en-net-parte-i/>

<https://enmilocalfunciona.io/construyendo-una-web-api-rest-segura-con-json-web-token-en-net-parte-ii/>

<https://enmilocalfunciona.io/construyendo-una-web-api-rest-segura-con-json-web-token-en-net-parte-iii/>

<https://github.com/santimacnet/WebAPI-Segura-JWT>

<https://jwt.io>

<https://www.jsonwebtoken.io>

<https://tools.ietf.org/html/rfc7519>

<https://docs.microsoft.com/en-us/security-updates/securityadvisories/2017/3214296>

<https://auth0.com/blog/ten-things-you-should-know-about-tokens-and-cookies/>