

Programación III

API REST - SLIM - JWT

Clase 10

Temas a Tratar

- Autenticación con Tokens
- JWT

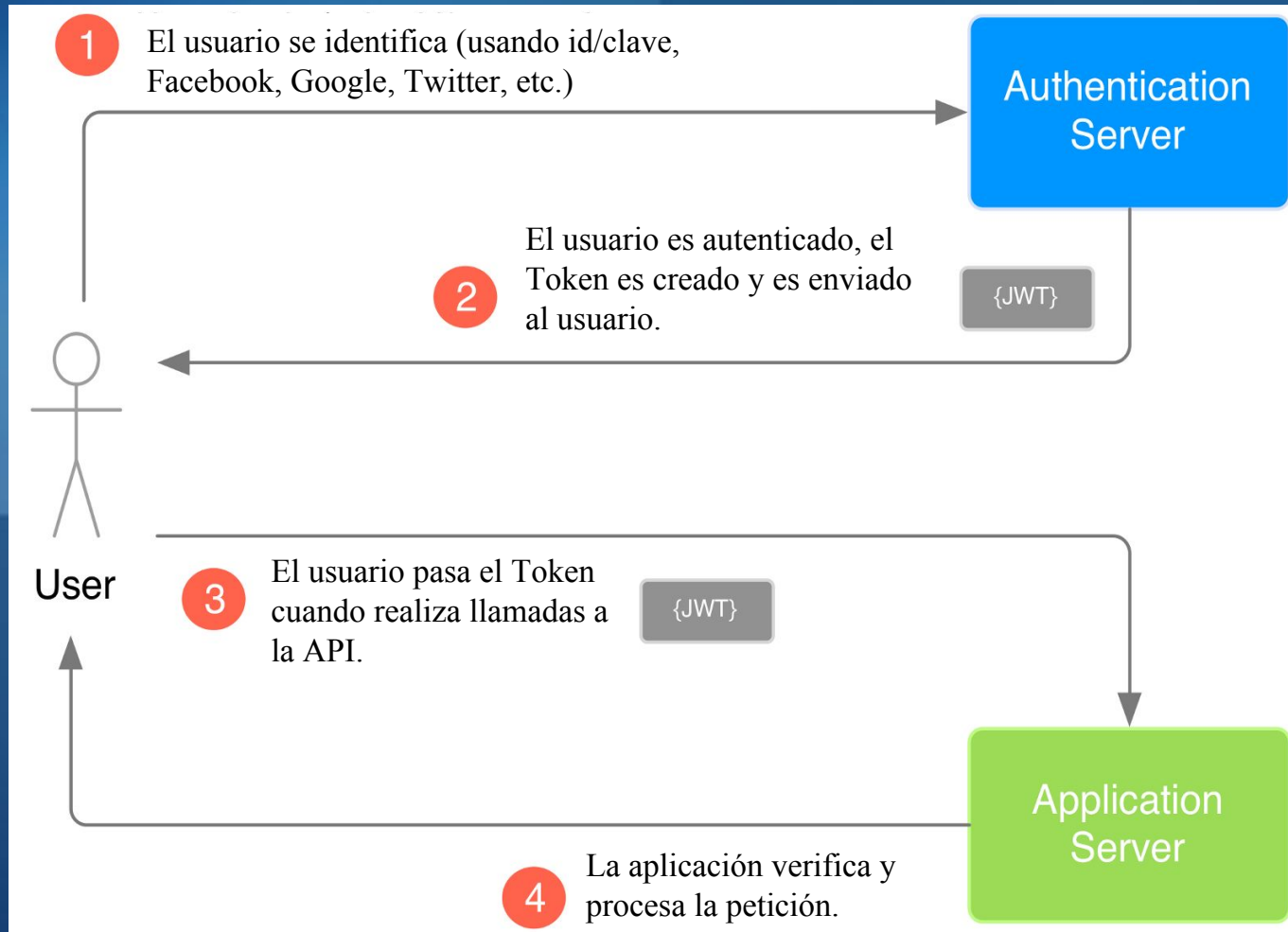
Autenticación con Tokens (1/3)

- Una de las nuevas tendencias es la autenticación por medio de *Tokens* y que el backend sea un API RESTful.
- Funcionamiento:
 - El usuario se autentica con usuario/contraseña o a través de un proveedor (como Twitter, Facebook o Google).
 - A partir de entonces, cada petición HTTP que haga el usuario va acompañada de un *Token* en la cabecera.
 - Este Token no es más que una firma cifrada que le permite al API identificar al usuario.
 - Pero este Token no se almacena en el servidor, si no del lado del cliente (en el *localStorage* o *sessionStorage*) y el API es el que se encarga de descifrar ese Token y redirigir el flujo de la aplicación en un sentido u otro.

Autenticación con Tokens (2/3)

- Como los tokens son almacenados en el lado del cliente, no hay información de estado y la aplicación se vuelve totalmente escalable.
- Se puede usar el mismo API para diferentes aplicaciones (Web, Mobile, Android, iOS, ...)
 - Solo hay que enviar los datos en formato JSON y cifrar/descifrar tokens en la autenticación y posteriores peticiones HTTP, a través de un MIDDLEWARE.
- También añade más seguridad.
 - Al no utilizar cookies para almacenar la información del usuario, se evita ataques CSRF (*Cross-Site Request Forgery*) que manipulen la sesión que se envía al backend.

Autenticación con Tokens (3/3)



Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

JWT (1/3)

- Un JSON Web Token (o JWT) es un estándar abierto ([RFC-7519](#)) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.
- Un JWT está compuesto por 3 partes:
 - el encabezado (header),
 - el payload
 - y la firma (signature)

```
header.payload.signature
```

Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

JWT - Header

- La primera parte es la cabecera del token, que a su vez tiene otras dos partes:
 - El tipo, en este caso un JWT
 - y la codificación utilizada. Comúnmente es el algoritmo *HMAC SHA256*.

El contenido sin codificar es el siguiente:

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

Codificado...

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

JWT - Payload (1/2)

- EL *Payload* está compuesto por los llamados JWT Claims donde irán colocados los atributos que definen al token.
- Los más comunes a utilizar son:
 - sub: Identifica el sujeto del token. Ej. Id de usuario.
 - iat: Identifica la fecha de creación del token, válido si se quiere poner una fecha de caducidad. En formato de tiempo UNIX.
 - exp: Identifica a la fecha de expiración del token. Se calcula a partir del iat. También en formato de tiempo UNIX.

```
{
  "sub": "54a8ce618e91b0b13665e2f9",
  "iat": "1424180484",
  "exp": "1425390142",
  "admin": true,
  "rol": 1
}
```

JWT - Payload (2/2)

- Al payload se le pueden agregar más campos, incluso personalizados.

```
{  
  "sub": "54a8ce618e91b0b13665e2f9",  
  "iat": "1424180484",  
  "exp": "1425390142",  
  "admin": true,  
  "rol": 1  
}
```

- Codificado...

```
eyJzdWIiOiI1NGE4Y2U2MThlOTFmGIXMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiaXhwIjojMTQyNTM5MDE0IiwiaSImFkbWluIjp0cnVlLCJyb2wiOjF9
```

JWT - Signature

- La firma es la tercera y última parte del JWT.
- Está formada por los anteriores componentes (Header y Payload) cifrados en *Base64* con una clave secreta (almacenada en nuestro backend).
- Así sirve de *Hash* para comprobar que todo está bien.

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

Codificado...

```
KnvuXrazg-Iqm24UFz_nij125eSjPsxiR2KrhLZLv_Y
```

JWT Completo

- El JWT una vez codificado tendrá el siguiente aspecto:

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI1NGE4Y2U2MThlOTFiMGIXMzY2NWUyZjkiLCJpYXQiOiIxNDI0MTgwNDg0IiwiaXhwIjoibTQyNTM5MDE0IiwiaWFkbWluIjp0cnV1LCJyb2wiOjF9.KnvvXrazg-Iqm24UFz_nij125eSjPsxiR2KrhLZLv_Y
```

- Para verificar el JWT dirigirse hacia jwt.io.

JWT Ciclo de Vida

Ciclo de vida de un token JWT



Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

Nota (1/2)

- Es importante entender que el propósito de usar JWT NO es ocultar u ofuscar datos de ninguna manera.
- El motivo por el que se utiliza JWT es para demostrar que los datos enviados fueron realmente creados por una fuente auténtica.
- Los datos dentro de un JWT están codificados y firmados, no cifrados.

Nota (2/2)

- El propósito de codificar datos es transformar la estructura de los mismos.
- Los datos firmados permiten que el receptor verifique la autenticidad de la fuente de los datos.
- Entonces, codificar y firmar datos NO protege los datos.
- Por otro lado, el objetivo principal del cifrado es proteger los datos y evitar el acceso no autorizado.

Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

Definiciones (1/2)

- La codificación es para mantener la usabilidad de los datos y puede revertirse empleando el mismo algoritmo que codifica el contenido, es decir, no se utiliza ninguna clave.
- El cifrado es para mantener la confidencialidad de los datos y requiere el uso de una clave (mantenida en secreto) para volver a texto plano.

Definiciones (2/2)

- Hashing es para validar la integridad del contenido mediante la detección de todas las modificaciones de los mismos mediante cambios obvios en la salida hash.
- La ofuscación se usa para evitar que las personas entiendan el significado de algo, y se usa a menudo con la ayuda de una computadora para evitar la ingeniería inversa exitosa y/o el robo de la funcionalidad de un producto.

Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

JWT en Slim - Crear

- La creación de un JWT se realiza por medio del método estático *encode* de la clase *Firebase\JWT*.

```
$app->post("/jwt/crearToken[/]", function (Request $request, Response $response, array $args) : Response {

    $datos = $request->getParsedBody();
    $ahora = time();

    //PARAMETROS DEL PAYLOAD -- https://tools.ietf.org/html/rfc7519#section-4.1 --
    //SE PUEDEN AGREGAR LOS PROPIOS, EJ. 'app'=> "API REST 2021"
    $payload = array(
        'iat' => $ahora,           //CUANDO SE CREO EL JWT (OPCIONAL)
        'exp' => $ahora + (30),    //INDICA EL TIEMPO DE VENCIMIENTO DEL JWT (OPCIONAL)
        'data' => $datos,         //DATOS DEL JWT
        'app' => "API REST 2021"  //INFO DE LA APLICACION (PROPIO)
    );

    //CODIFICO A JWT (PAYLOAD, CLAVE, ALGORITMO DE CODIFICACION)
    $token = JWT::encode($payload, "miClaveSecreta", "HS256");

    $newResponse = $response->withStatus(200, "Éxito!!! JSON enviado.");

    //GENERO EL JSON A PARTIR DEL ARRAY.
    $newResponse->getBody()->write(json_encode($token));

    //INDICO EL TIPO DE CONTENIDO QUE SE RETORNARÁ (EN EL HEADER).
    return $newResponse->withHeader('Content-Type', 'application/json');

});
```

Temas a Tratar

- Autenticación con Tokens
- JWT
 - ¿Qué es JWT?
 - Partes del JWT.
 - Notas.
 - Definiciones.
 - Crear en Slim.
 - Verificar en Slim.

JWT en Slim - Verificar

- La verificación del JWT se realiza por medio del método estático *decode* de la clase *Firebase\JWT*.

```
$app->post("/jwt/verificarToken[/]", function (Request $request, Response $response, array $args) : Response {  
  
    $datos = $request->getParsedBody();  
    $token = $datos['token'];  
  
    $retorno = new stdClass();  
    $status = 200;  
  
    try {  
        //DECODIFICO EL TOKEN RECIBIDO  
        JWT::decode(  
            $token,                                //JWT  
            "miClaveSecreta",                       //CLAVE USADA EN LA CREACION  
            ['HS256']                               //ALGORITMO DE CODIFICACION  
        );  
  
        $retorno->mensaje = "Token OK!!!";  
    }  
    catch (Exception $e) {  
  
        $retorno->mensaje = "Token no válido!!! --> " . $e->getMessage();  
        $status = 500;  
    }  
  
    $newResponse = $response->withStatus($status);  
  
    $newResponse->getBody()->write(json_encode($retorno));  
  
    return $newResponse->withHeader('Content-Type', 'application/json');  
});
```




Ejercitación