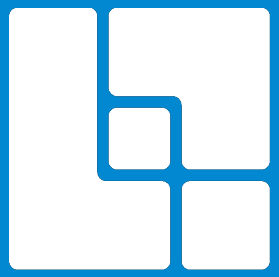


experts
coding

Autenticación en nuestras apps: Fácil, Sencillo y para toda la familia

Quienes somos

Antonio Marín Alberdi

20 años dando caña

Especializado en Arquitectura de Soluciones, diseño de frameworks e implementación de soluciones técnicas.

Destaca en la creación de proyectos locos y pruebas de concepto arriesgadas pero absurdas.

antonio.marin@expertscoding.es

Microsoft
CERTIFIED
Professional Developer
Web Developer 4

Manuel Vilachán

Más de 15 vueltas al sol diseñando cohetes

Arquitecto de Software, muy experto en tecnologías Microsoft y diseño de soluciones técnicas.

Desarrollo software complicado, ayudo a Antonio en sus locas creaciones y en mis ratos libres hago pan.

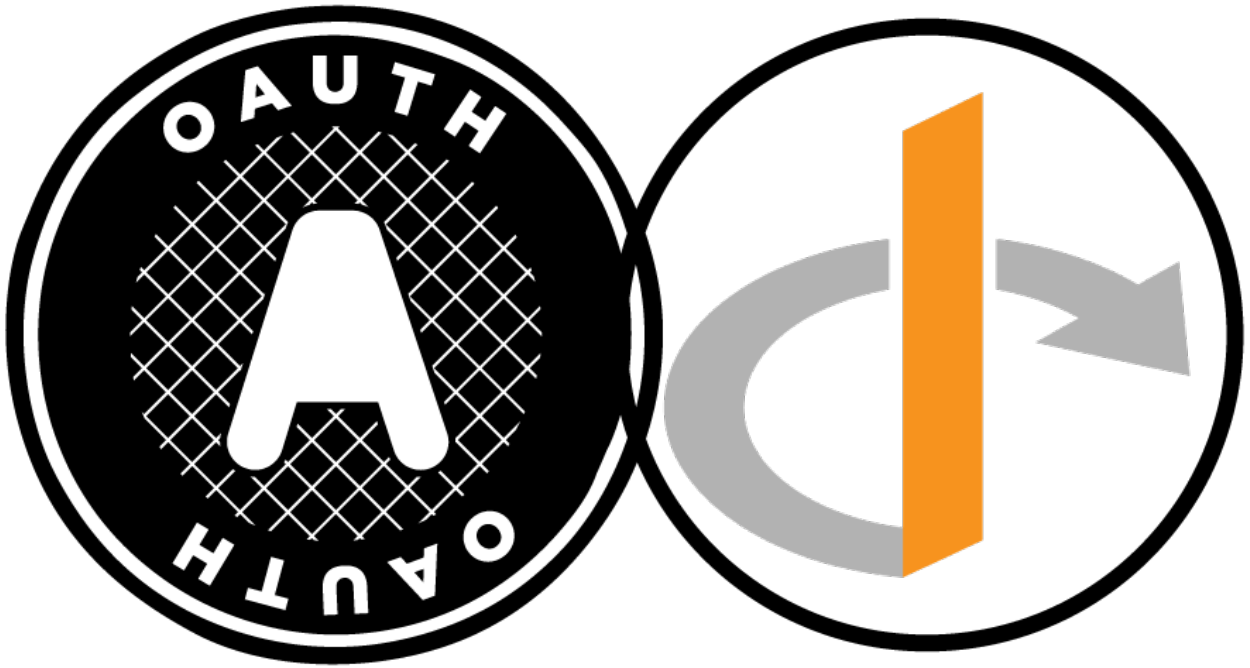
manuel.vilachan@expertscoding.es

¿Dónde están las demos?

https://github.com/expertscoding/T3chfest-2019-Workshop

Autenticación vs. Autorización

Autenticación vs. Autorización



Better Together

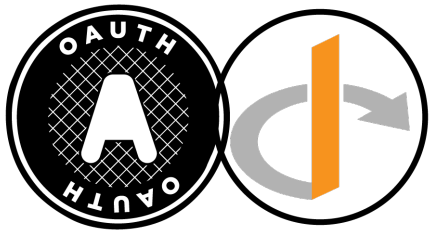
El protocolo – ¿Quién es quién?

¿Qué se define en cada protocolo?

- Endpoints principales
- Flujos de autorización
- Access Token (& Refresh)

- ID Token
- Endpoint UserInfo

El protocolo – Endpoints



- Discovery endpoint (/.well-known/openid-configuration): información acerca de nuestro servicio.

- Authorization endpoint (ej.: /connect/authorize): punto de inicio el flujo de autorización.
  - Parámetros habituales: client id, redirect uri, scope, response type.

- Token endpoint (ej.: /connect/token): donde obtener el token una vez autorizado (casi siempre).
  - Parámetros habituales: grant type, scope, code.

- User Info endpoint (ej.: /connect/userinfo): devuelve información acerca de un usuario para el que se emitió un token, en formato JSON por defecto.
  - Sin parámetros. Requiere de autorización normalmente mediante cabecera HTTP.

Let’s fun

KEEP

CALM

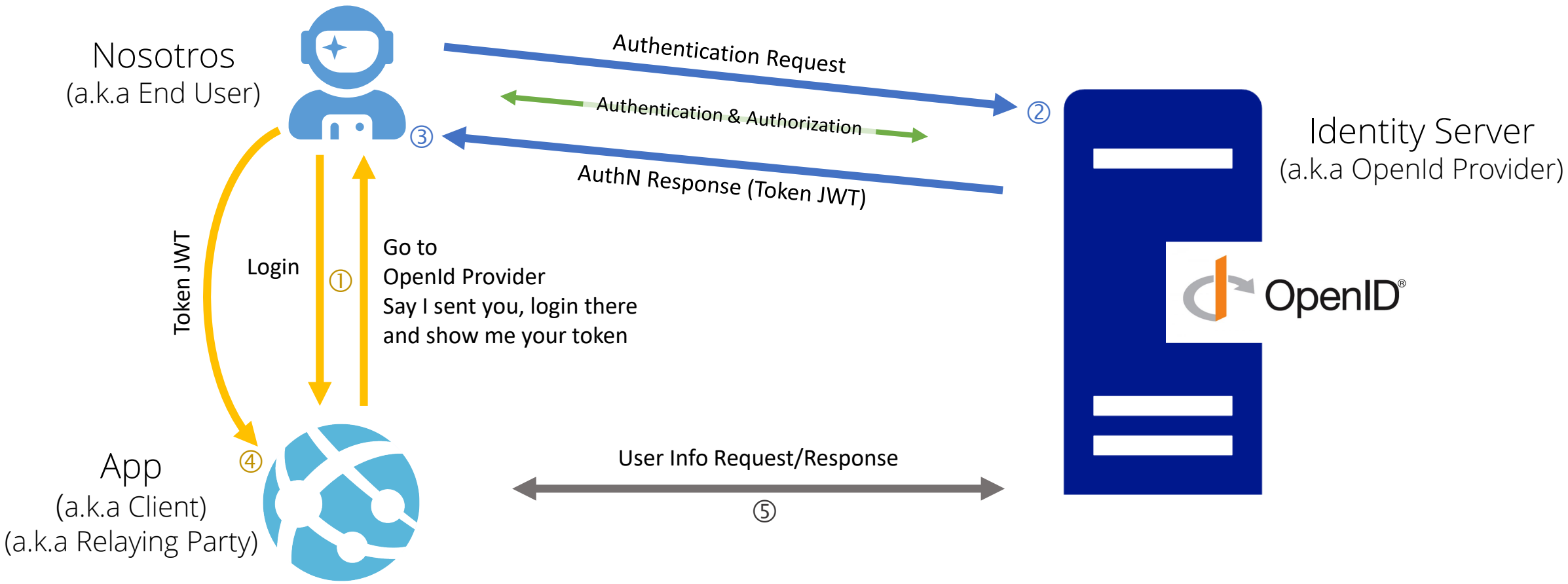
IT IS

CODING

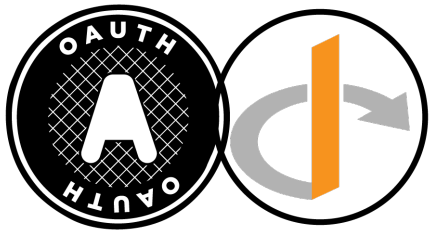
TIME

https://github.com/expertscoding/T3chfest-2019-Workshop

Behind the scene

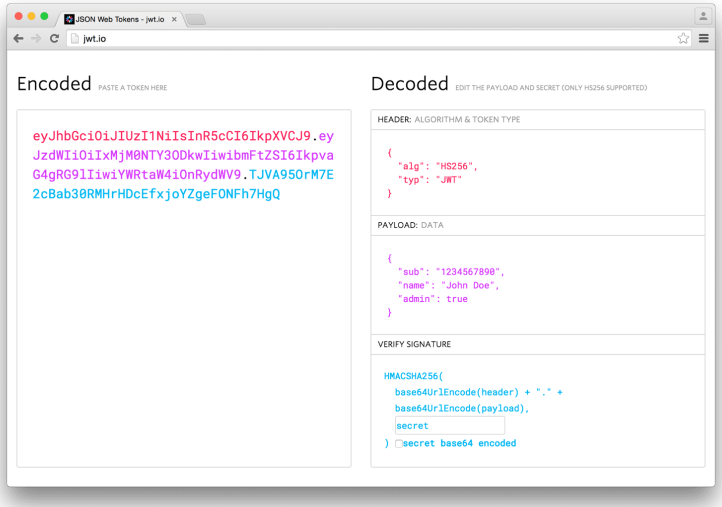


El protocolo – Flujos de autorización

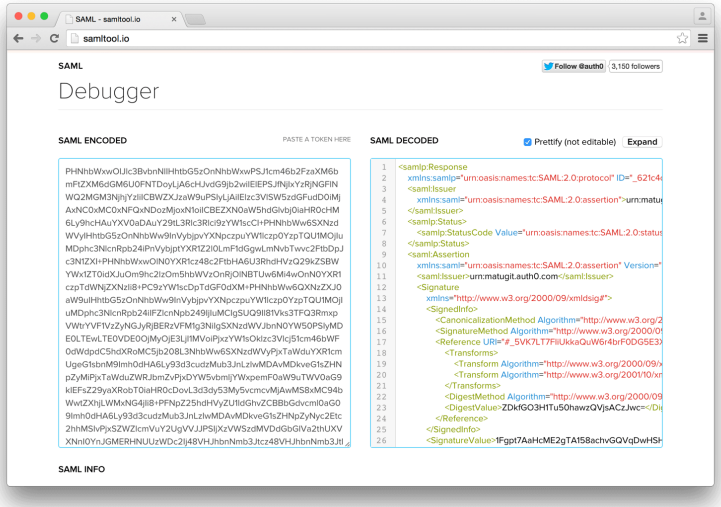


- **Authorization Code:** obtenemos un código que podemos intercambiar posteriormente por un id_token o access_token. Adecuado para clientes con capacidad de guardar secretos de forma segura.
- **Implicit:** todos los tokens se obtienen desde el endpoint de autorización en una sola llamada, no permite la obtención de refresh tokens. Especialmente pensado para aplicaciones “no seguras”.
- **Hybrid:** los tokens se obtienen desde diferentes endpoints. Requiere que el cliente sea capaz de guardar secretos y permite la obtención de refresh tokens.
- **Client Credentials:** Se usa para la autorización a recursos y requiere del almacenamiento seguro del secreto de cliente. No supone la obtención de ningún id_token.
- **Resource Owner:** Intercambio de usuario y password entre cliente y server. Destinado a entornos seguros o dispositivos. No supone tampoco la obtención de ningún id_token.

Anatomía de un Token

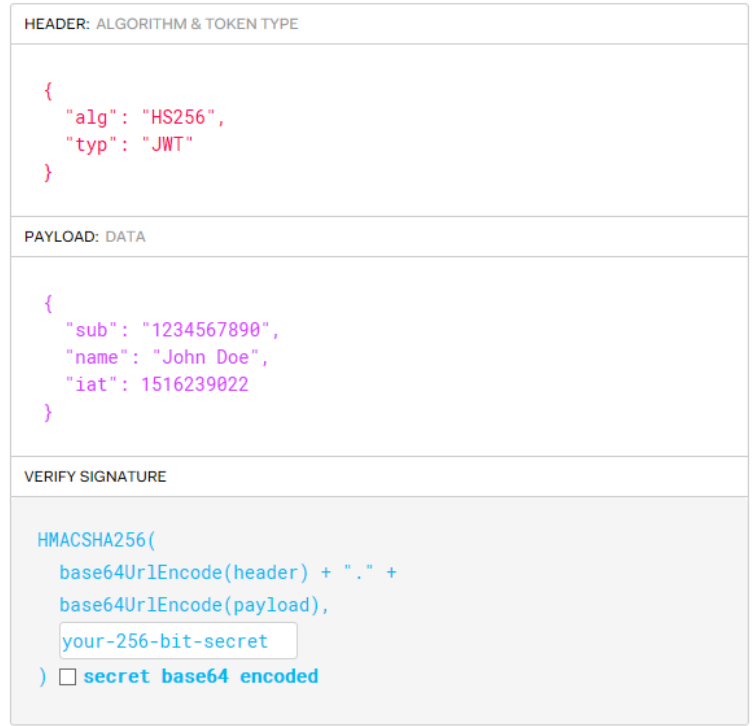


VS



eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c

https://jwt.io/



Anatomía de un id_token

Claims obligatorios:

- iss: issuer
- sub: subject
- aud: audience
- exp: expiration time
- iat: issued at

Claims opcionales:

- auth_time: End-user authentication occurred.
- nonce
- acr: autentication context class reference
- amr: autentication method references
- azp: authorized party

El servidor OidC puede incluir otros claims pertenecientes a los scopes solicitados durante la autorización.

Trabajando con nuestro servicio de autenticación

https://github.com/expertscoding/T3chfest-2019-Workshop

Q & A

github sources

Gracias por tu atención