Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

APIs y servicios web

IIC2513 - Tecnologías y Aplicaciones Web

Sebastián Vicencio R.
2do Semestre 2020

API

Application Programming Interface

API

¿Qué es?

Es una interfaz o intermediario entre una aplicación o software y quien desee interactuar con la aplicación.

- **Se dice que una aplicación “expone” funcionalidades mediante una API**
- **Quien desee acceder a estas funcionalidades tiene que “consumir” la API**

API

¿Qué es?

Establecer comunicación entre aplicaciones, ya que quien consume la API puede perfectamente ser otra aplicación

```
document.querySelector('.some-class')
```

```
GET https://api.github.com/orgs/IIC2513-2020-2/repos
```

API Web

API que utiliza HTTP

API Web

¿Qué es?

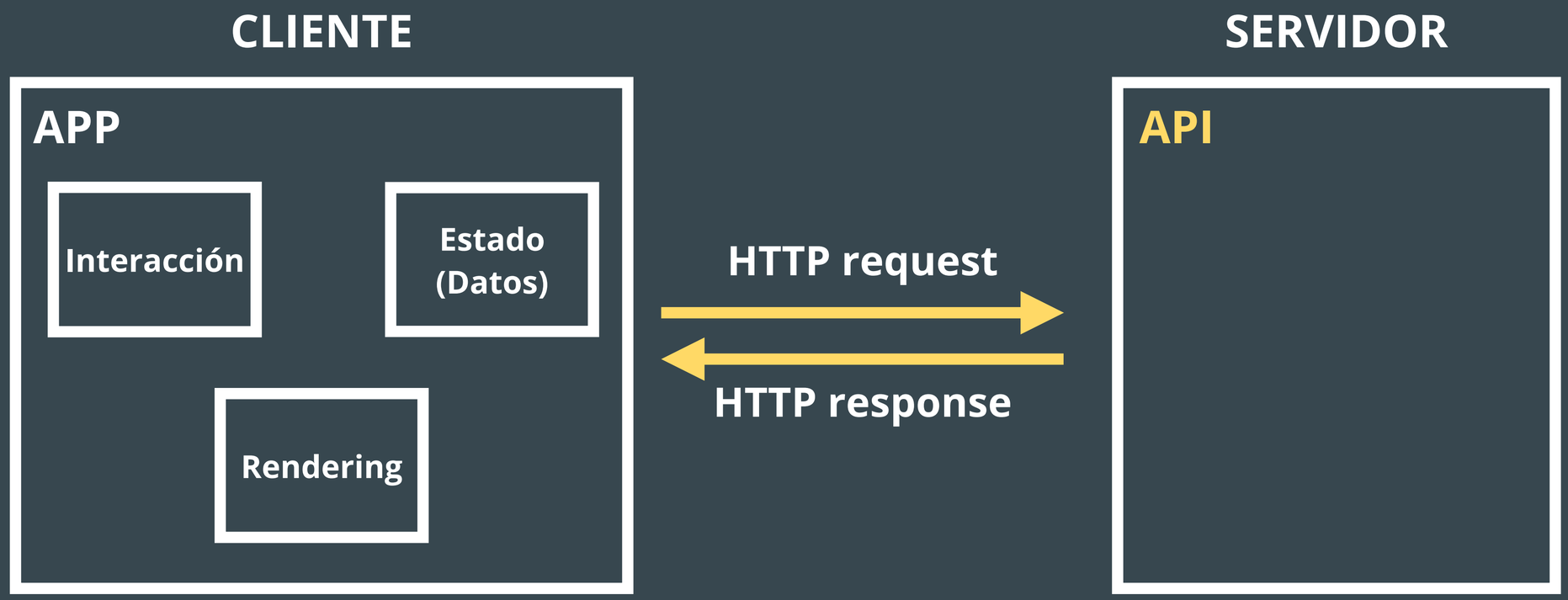
Es una API que se accede utilizando la infraestructura Web, es decir, el protocolo HTTP

- En su forma más genérica, el cliente hace un request y recibe datos como respuesta
- No necesariamente un documento HTML

API Web

¿Dónde está inserta?

Las URLs suelen llamarse endpoints



API Web

Características

Se conocen de forma genérica como "web services"

- Las respuestas contienen datos estructurados que son presentados en una manera estándar. Posibles de ser interpretados por diferentes tipos de aplicaciones
- Los formatos más utilizados para una API son JSON y XML

Arquitecturas WS

Cómo especificar un servicio web

Arquitecturas web services

¿Qué es?

Existen diferentes arquitecturas o estilos para especificar un servicio web

Lo que tienen en común:

- **Scope (qué vamos a consultar)**
- **Acción/operación (cómo vamos a consultar)**
- **Datos (con qué vamos a consultar)**

Arquitectura RPC

Remote Procedure Call

Arquitectura RPC

¿Qué es?

Llamar a un procedimiento (función) de forma remota

El medio utilizado es internet (puede ser HTTP o incluso SMTP)

- **Analogía: llamar a una función en JavaScript**
- **Necesitamos nombre del procedimiento y argumentos**

XML-RPC

Ejemplo

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
   <methodName>sample.sum</methodName>
   <params>
      <param>
         <value><int>17</int></value>
      </param>

      <param>
         <value><int>13</int></value>
      </param>
   </params>
</methodCall>
```

Fuente: TutorialsPoint

SOAP

Ejemplo

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

<?xml version = "1.0"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-envelope"
   SOAP-ENV:encodingStyle = "http://www.w3.org/2001/12/soap-encoding">

   <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotations">
      <m:GetQuotation>
         <m:QuotationsName>MiscroSoft</m:QuotationsName>
      </m:GetQuotation>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fuente: TutorialsPoint

JSON-RPC

Ejemplo

```
POST / HTTP/1.1
HOST: api.example.com
Content-Type: application/json

{“jsonrpc": "2.0", "method": "subtract", "params": {"subtrahend": 23,
"minuend": 42}, "id": 3}
```

Fuente: JSON-RPC Spec

Arquitectura REST

REpresentational **S**tate **T**ransfer

Arquitectura REST

¿Qué es?

Estilo arquitectónico que define una **serie de principios** (o restricciones) que debe cumplir un **servicio web**

Disertación publicada por **Roy Fielding** en el año 2000 para conseguir grado de PhD

Arquitectura REST

Características

Utiliza los conceptos de recursos y representaciones de estos recursos

Se apoya fuertemente en los métodos HTTP para especificar acciones

Arquitectura REST

Principios

- Client-server
- Stateless
- Cacheable
- Uniform interface
  - Resource-based
  - Manipulation of Resources Through Representations
  - Self-descriptive Messages
  - Hypermedia as the Engine of Application State (HATEOAS)
- Layered system
- Code on Demand (opcional)

RESTful WS

¿Cómo se ven?

RESTful web services

¿Cómo se ven?

REST se apoya fuertemente en HTTP

- Scope en **path**
- Operación en **método HTTP**
- Datos en **body**

RESTful web services

¿Cómo se ven?

```
GET     /users
POST    /users
GET     /users/2
PUT     /users/2
PATCH   /users/2
DELETE  /users/2
```

Ejemplo RESTful WS

Github API

Consumir una API

El consumidor

Consumir APIs

¿Quiénes consumen APIs?

Frontend webapps

Backend webapps

Mobile apps

Smart devices apps

Consumir APIs

¿Cómo acceder a una API programáticamente?

Librerías que permitan hacer **requests HTTP**

Browser

~~XMLHttpRequest~~

fetch

Node

node-fetch

axios

superagent

Documentación API

¿Cómo consumir una API?

La mayoría de las APIs REST incluyen una documentación

Puede entenderse como un manual de uso de la API

La calidad de la documentación influye en la experiencia de los consumidores de la API

Documentación API

Formato

Vimos un ejemplo con Github API

`GET` `/repos/{owner}/{repo}/commits`

Parameters

Name	Type	In	Description
`accept`	string	header	Setting to `application/vnd.github.v3+json` is recommended.
`owner`	string	path	
`repo`	string	path	
`sha`	string	query	SHA or branch to start listing commits from. Default: the repository's default branch (usually `master`).
`path`	string	query	Only commits containing this file path will be returned.
`author`	string	query	GitHub login or email address by which to filter by commit author.
`since`	string	query	Only show notifications updated after the given time. This is a timestamp in ISO 8601 format: `YYYY-MM-DDTHH:MM:SSZ`.
`until`	string	query	Only commits before this date will be returned. This is a timestamp in ISO 8601 format: `YYYY-MM-DDTHH:MM:SSZ`.
`per_page`	integer	query	Results per page (max 100)
`page`	integer	query	Page number of the results to fetch.

Documentación API

Formato

¿Qué suele incluir?

- Método HTTP + endpoint (path)
- Parámetros request
- Response
- Errors
- Ejemplos

Default response

Status: 200 OK


```
[
  {
    "id": "6dcb09b5b57875f334f61aebed695e2e4193db5e",
    "tree_id": "6dcb09b5b57875f334f61aebed695e2e4193db5e",
    "message": "Fix all the bugs",
    "timestamp": "2016-10-10T00:00:00Z",
    "author": {
      "name": "Monalisa Octocat",
      "email": "mona@github.com"
    },
    "committer": {
      "name": "Monalisa Octocat",
      "email": "mona@github.com"
    }
  },
  {
    "id": "6dcb09b5b57875f334f61aebed695e2e4193db5e",
    "tree_id": "6dcb09b5b57875f334f61aebed695e2e4193db5e",
    "message": "Fix all the bugs",
    "timestamp": "2016-10-10T00:00:00Z",
    "author": {
      "name": "Monalisa Octocat",
      "email": "mona@github.com"
    },
    "committer": {
```

Documentación API

Formato OpenAPI

Estándar que **automatiza** el proceso de **documentación** de una API REST

API en documento JSON o YAML

- Estructura definida
- Entendida por humanos y máquinas
- Ejemplo Github API

Exponer una API

El proveedor

Exponer una API

¿Cómo construir una API en koa?

Definir rutas que respondan en JSON

- Rutas que respondan en HTML y JSON
- Rutas API aisladas
- Una aplicación diferente que sea directamente una API

Exponer una API

Rutas que responden en HTML y JSON

Asignar a `ctx.body` un objeto JavaScript

- **Response será convertido a string**
- **Response incluirá header `Content-Type: application/json`**

Exponer una API

Rutas que responden en HTML y JSON

Método `ctx.accepts` de koa

```
switch (ctx.accepts(['html', 'json'])) {
  case 'html':
    await ctx.render(/* ... */);
    break;
  case 'json':
    ctx.body = { /* JSON response */ };
    break;
  default:
    break;
}
```

Header request

```
Accept: application/json
```

Exponer una API

Rutas que responden en HTML y JSON

Código con varias rutas, todas con `ctx.accepts`

Puede ser poco mantenible

```
router.get('event', '/:id', async (ctx) => {
  const { event } = ctx.state;
  switch (ctx.accepts(['html', 'json'])) {
    case 'html':
      await ctx.render('events/show', {
        event,
      });
      break;
    case 'json':
      ctx.body = event;
      break;
    default:
      break;
  }
});

router.get('event-attendances', '/:id/attendances', async (ctx) => {
  const { event: { attendees } } = ctx.state;
  switch (ctx.accepts(['html', 'json'])) {
    case 'html':
      await ctx.render('events/attendances', { attendees });
      break;
    case 'json':
      ctx.body = event.attendees.map(({ id, email }) => ({ id, email }));
      break;
    default:
      break;
  }
});

router.post('event-create-attendance', '/:id/attendances', async (ctx) => {
  const { currentUser, event } = ctx.state;
  /* Route code */
  switch (ctx.accepts(['html', 'json'])) {
    case 'html':
      ctx.redirect(ctx.router.url('event', event.id));
      break;
    case 'json':
      ctx.status = 201;
      ctx.body = { message: 'Attendee added' };
      break;
    default:
      break;
  }
});
```

Exponer una API

Rutas API aisladas

Conjunto de rutas "separadas" del resto de la aplicación

Montamos las rutas de la API bajo otro "namespace"

```
// app.js
const apiRoutes = require('./routes/api');

/* ... */

app.use(apiRoutes.routes());
```

Exponer una API

Rutas API aisladas

```
// routes/api/index.js
const KoaRouter = require('koa-router');

const router = new KoaRouter({ prefix: '/api' });

router.get('/', async (ctx) => {
  ctx.body = { message: 'savetalk API' }
});

// router.use('/events', events.routes());

module.exports = router;
```

- Sólo respondemos JSON
- Podría ser necesario obtener datos de diferente manera
- Podemos aplicar middlewares específicos para la API

Autenticación API

¿Cómo funciona?

Autenticación API

¿Sirve lo que ya hemos visto?

Método tradicional para autenticar usuarios es mediante sesión (cookies)

¿Podemos utilizar cookies?

Sí, si sabemos que la API será utilizada desde un browser

Autenticación API

¿Sirve lo que ya hemos visto?

¿Qué pasa si queremos **consumir la API** desde una **app móvil**, o incluso desde otra **backend app**?

APIs incluyen otros **mecanismos de autenticación**

El más conocido es **autenticación basada en token**

Autenticación API

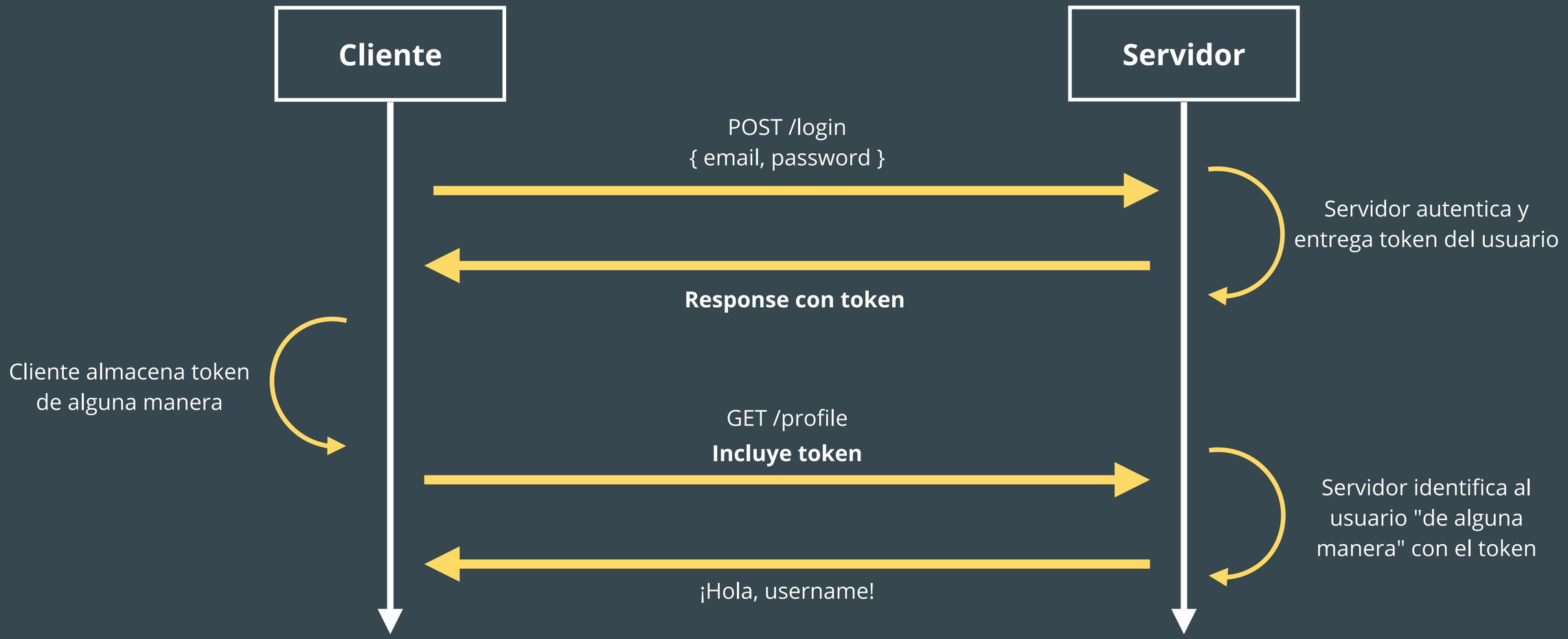
Autenticación basada en token

Access token

String con caracteres "random" que sirve para identificar a un usuario

Autenticación API

Flujo autenticación



Autenticación API

JSON Web Tokens

Estándar que permite transmitir información en un objeto JSON de forma segura entre dos entidades

Especificación del estándar RFC7519 del año 2015

Autenticación API

JSON Web Tokens

Estructura

Header . Payload . Signature

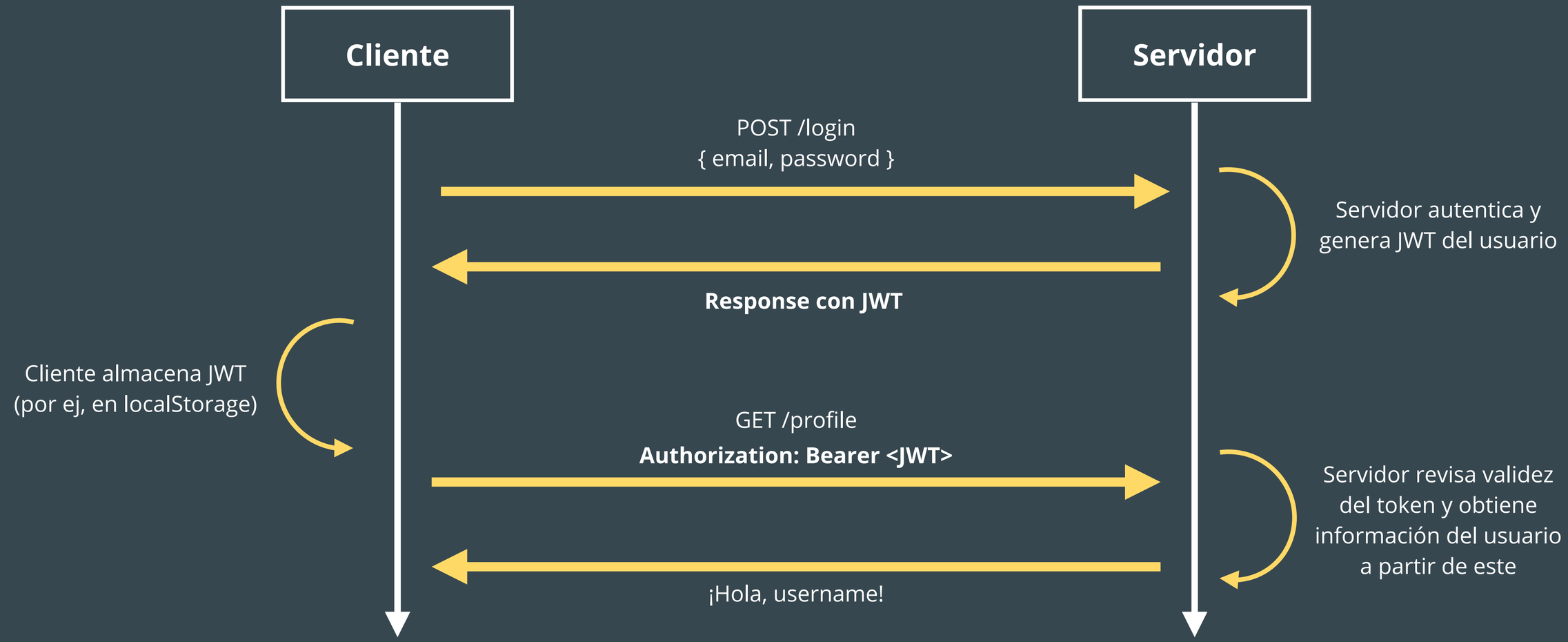
Base64

Base64

Header (Base64)
+
Payload (Base64)
Signed with secret + algorithm

Autenticación API

Flujo autenticación con JWT



Para decodificar JWT: jwt.io

Autenticación API

¿Y el sign out?

- Con cookies basta con **borrar la cookie**
  Browser automáticamente **no la enviará más**
- **Token es almacenado en browser**. Podría seguir enviándolo en el futuro
- Hay que **"revocar" el token**
  Distintas estrategias que no veremos

Para profundizar, aquí hay un paper respecto a revocación

Autenticación API

¿Cómo la manejamos en Node (koa)

jsonwebtoken

Generar JWT al hacer login

koa-jwt

Verificar JWT en cada request que llega al servidor

Referencias

- Perry Eising (Medium) - "What exactly IS an API?"
- W3C - "Web Services overview - Design Issues"
- Phil Sturgeon (Smashing Magazine) - "Understanding RPC Vs REST For HTTP APIs"
- Roy Fielding - “Architectural Styles and the Design of Network-based Software Architectures”
- REST API Tutorial - “What Is REST?”

Referencias

- Keshav Vasudevan (Swagger Blog) - “What is API Documentation, and Why It Matters?”
- Swagger - “OpenAPI Specification - Version 3.0.3”
- Swagger - “About Swagger Specification”
- Auth0 - “Get Started with JSON Web Tokens”
- Auth0 - “Token Based Authentication Made Easy”