Clase 14

Cookies y sesiones

IIC2513 - Tecnologías y Aplicaciones Web

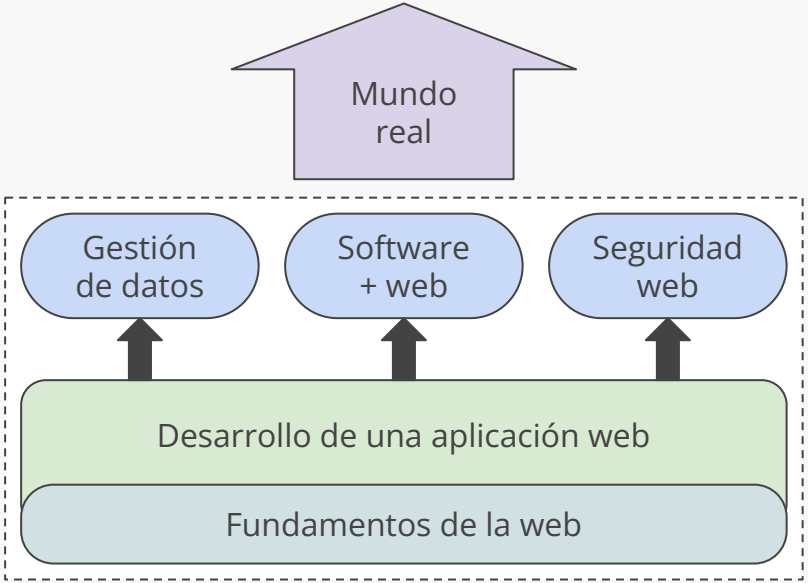
Antonio Ossa Guerra
aaossa@ing.puc.cl

1. El proyecto ya empezó (E0)
2. Reúnanse con sus ayudantes
3. Actividad la próxima clase

Anuncios

Contenidos del curso

(1) Fundamentos de la web
(2) Desarrollo de una app web
(3) Gestión de datos
(4) Software + web
(5) Seguridad web
(6) El mundo real

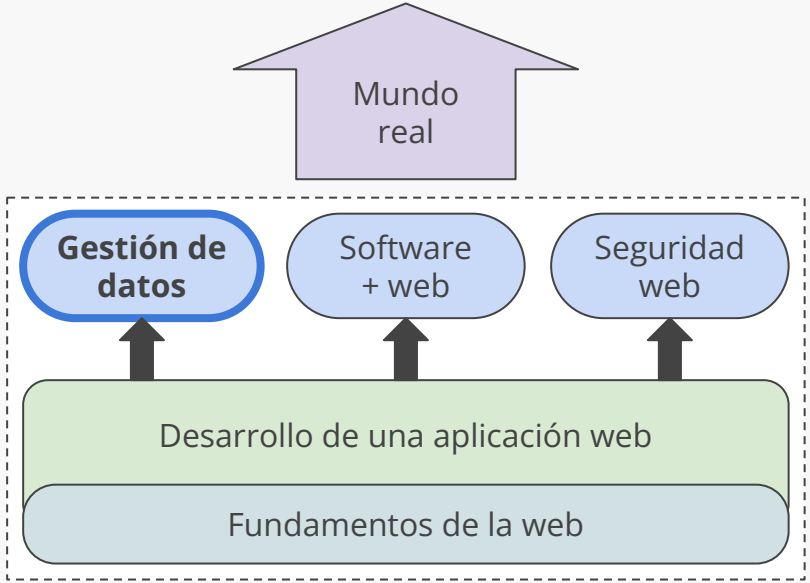


Contenidos del curso

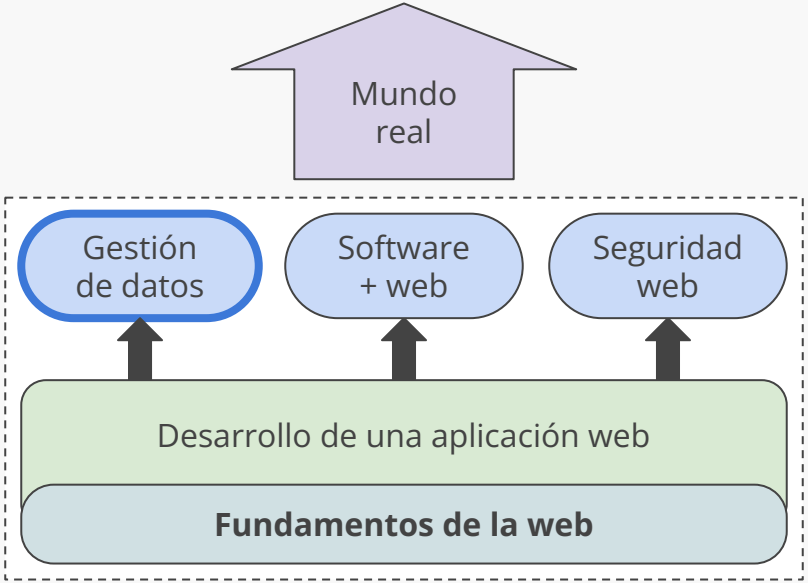
Gestión de datos

Veremos cómo diseñar y manejar ***cookies***, ***sesiones*** y **otros datos** en el navegador...

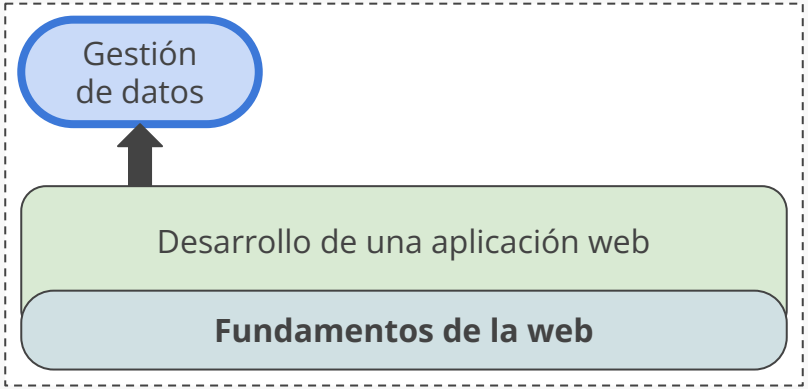
... para habilitar **funcionalidades avanzadas** de manera segura y con técnicas modernas



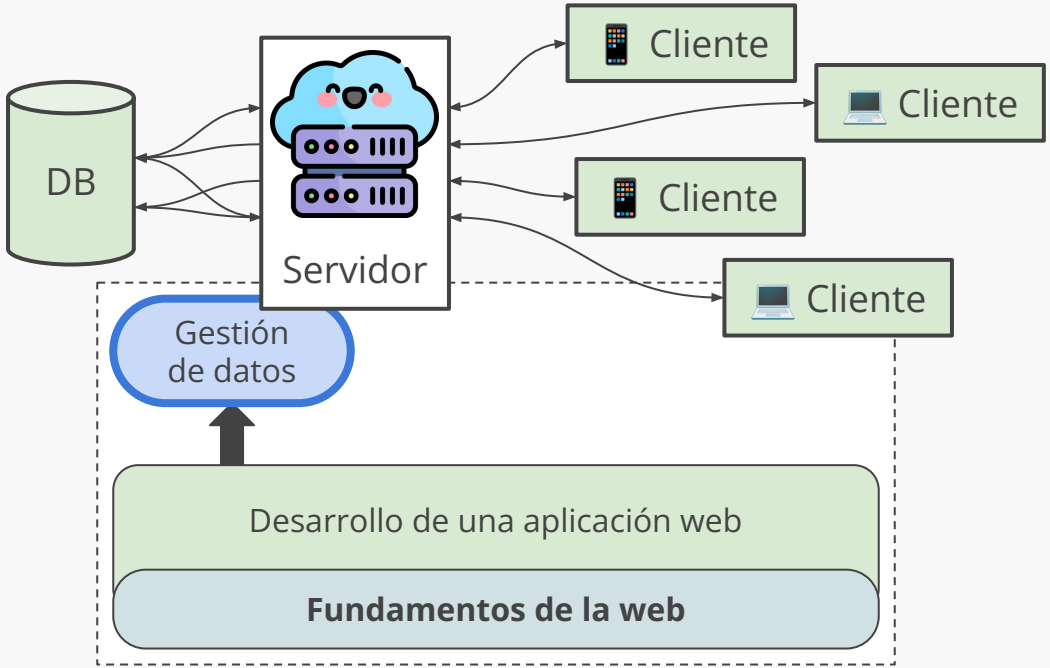
Contenidos del curso



Contenidos del curso



Motivación


Cliente
Cliente
DB
Cliente
Servidor
Cliente
Gestión
de datos
Desarrollo de una aplicación web
Fundamentos de la web

Motivación

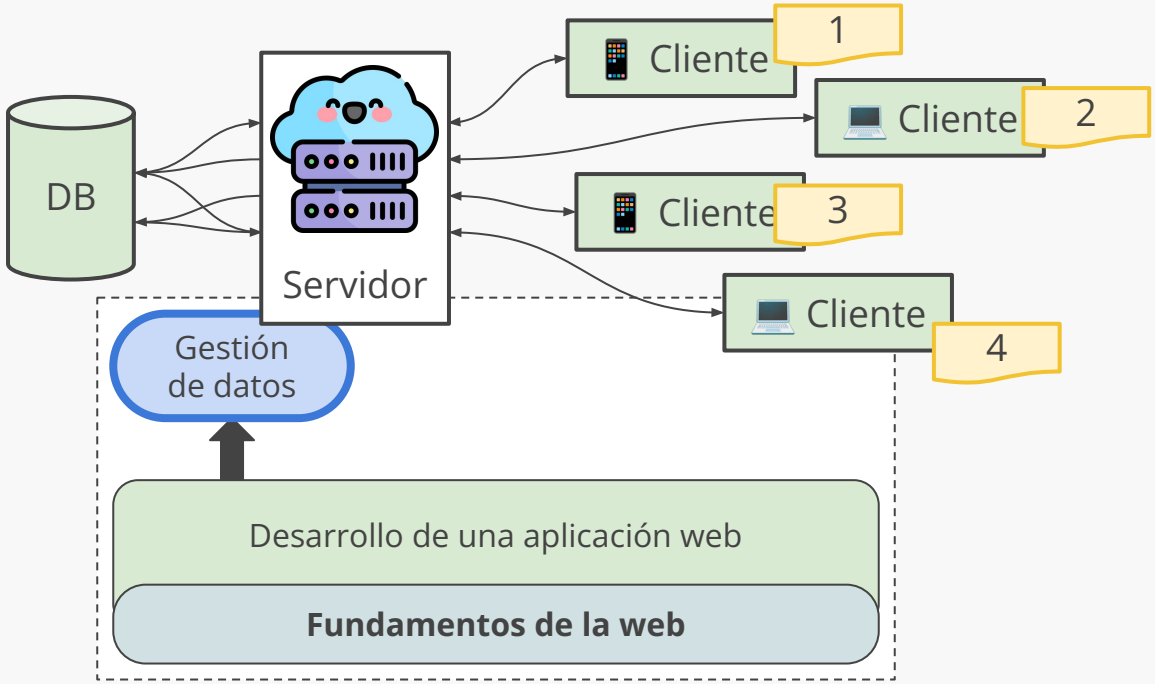


Pregunta#¿Cómo podemos diferenciar entre clientes?

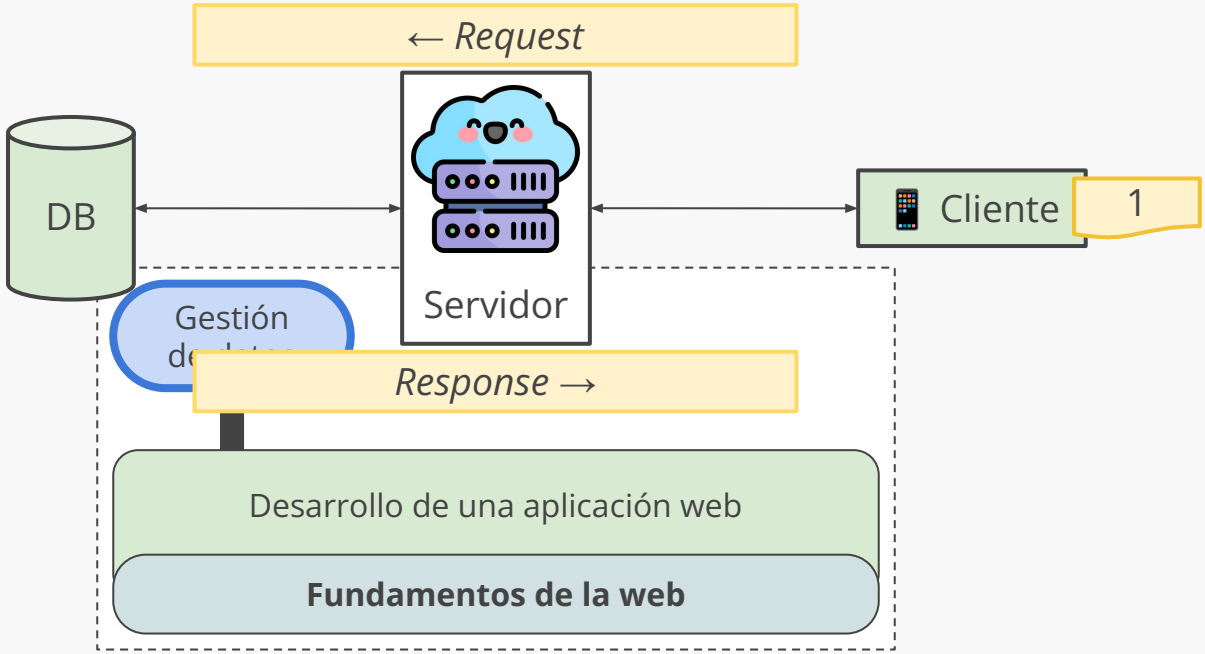
¿Cómo podemos diferenciar entre clientes?

- Nos gustaría autenticar al usuario y tener esa información disponible **cada vez** que procesamos una *request*
- Más generalmente, en una aplicación web nos gustaría **mantener el estado** en cada navegador activo (***session state***)

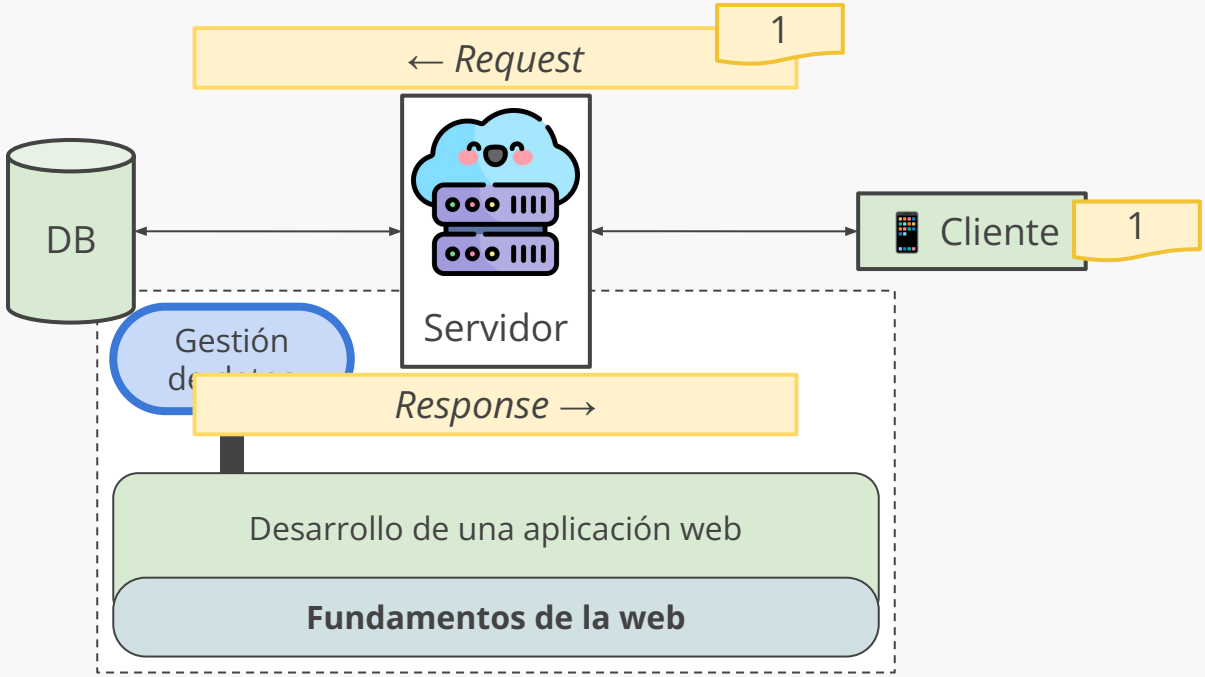
Motivación


DB
Servidor
Cliente
1
Cliente
2
Cliente
3
Cliente
4
Gestión de datos
Desarrollo de una aplicación web
Fundamentos de la web

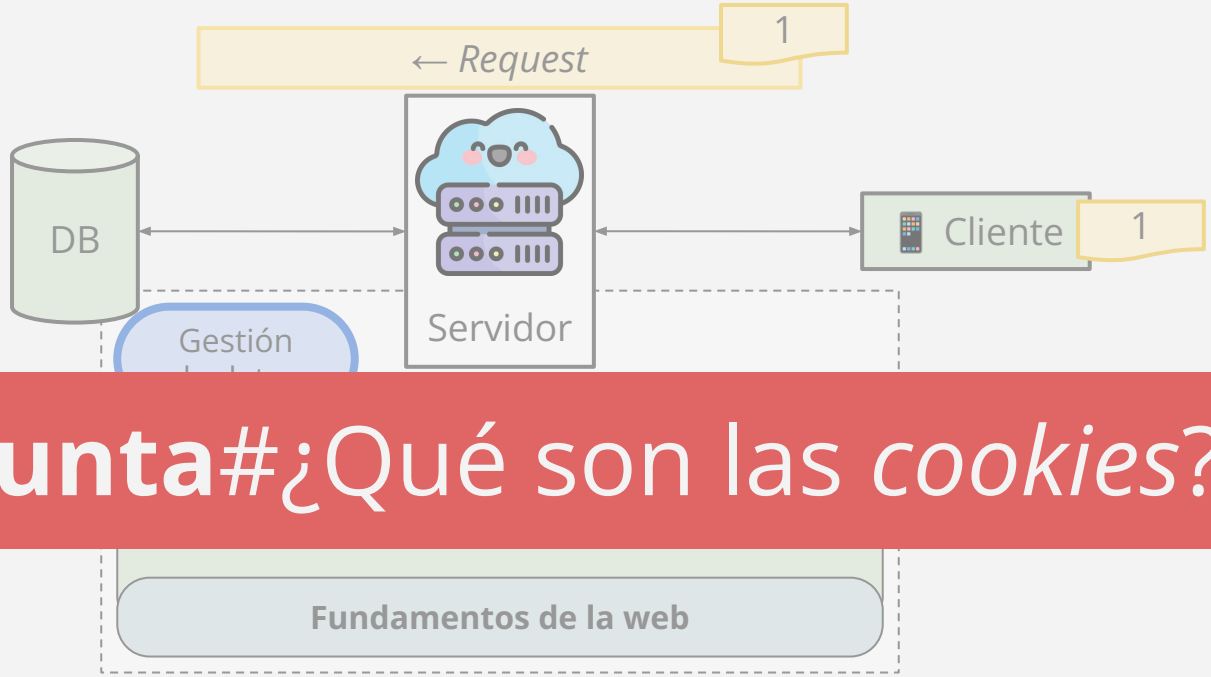
Motivación


← Request
DB
Servidor
Cliente
1
Gestión
Response →
Desarrollo de una aplicación web
Fundamentos de la web

Motivación


1
← Request
DB
Servidor
Cliente
1
Gestión
Response →
Desarrollo de una aplicación web
Fundamentos de la web

Motivación



Pregunta#¿Qué son las *cookies*?

Agenda

✨ ***Cookies* HTTP**

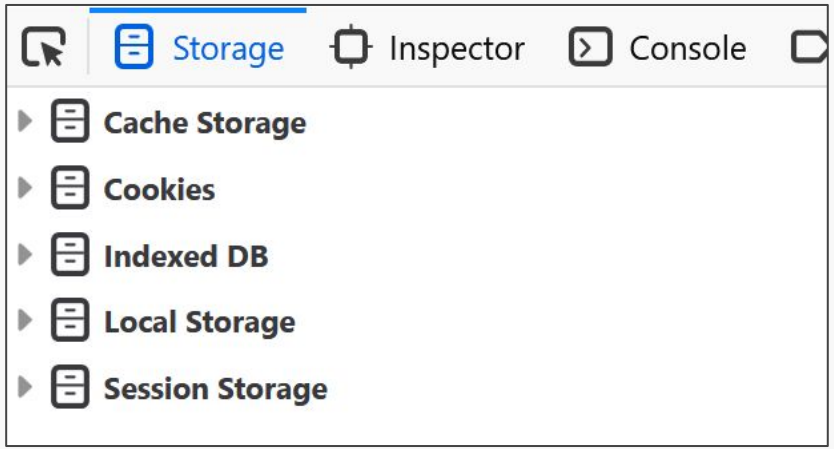
Manejando sesiones

Almacenamiento *client-side*

Cada vez existe más control sobre lo que podemos almacenar del lado del cliente, en almacenamientos de distinta duración y de distinto nivel de complejidad

Elementos que deben conocer:

- Cookies
- Web Storage API
- IndexedDB API
- Cache API



Fuente: Client-side storage - Learn web development | MDN

Almacenamiento *client-side*

- Cookies es la manera más tradicional y conocida de almacenar pequeñas piezas de información del lado del cliente (aunque hay novedades como Cookie Store API)
- La Web Storage API nos permite almacenar datos más pequeños (solo *strings*), con distintos niveles de persistencia, con absoluto control desde JavaScript
- La IndexedDB API permite que el navegador utilice un sistema de base de datos completo para almacenar datos más complejos (objetos de JS)
- La Cache API nos permite almacenar y recuperar solicitudes y respuestas con el fin de entregar respuestas rápidas a solicitudes, pero también puede usarse de almacén

Fuente: Client-side storage - Learn web development | MDN

Almacenamiento *client-side*

- Cookies es la manera más tradicional y conocida de almacenar pequeñas piezas de información del lado del cliente (aunque hay novedades como Cookie Store API)

Clase 15*#Almacenamiento del lado del cliente*

- La IndexedDB API permite que el navegador utilice un sistema de base de datos completo para almacenar datos más complejos (objetos de JS)
- La Cache API nos permite almacenar y recuperar solicitudes y respuestas con el fin de entregar respuestas rápidas a solicitudes, pero también puede usarse de almacén

Fuente: Client-side storage - Learn web development | MDN

¿Qué es REST?

REST = REpresentational State Transfer

Estilo arquitectónico que define una **serie de principios** (o restricciones) que debe cumplir un **servicio web** para que sea escalable y fácil de usar*

(Disertación publicada por Roy Fielding en el año 2000 para conseguir su grado de PhD)

- En su definición, utiliza los conceptos de **recursos y representaciones** de recursos
- Se apoya fuertemente en los **métodos HTTP** para **especificar acciones**

Principios REST

Un servicio que cumpla con todo esto puede ser considerado **RESTful**

- 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 (optional)

Fuente: *REST API Tutorial | What is REST*

Principios REST

- Client-server deben estar separados y poder evolucionar independientemente
- Stateless cada request del cliente debe contener toda la información necesaria para completarla
- Cacheable una respuesta debería poder indicar si es cacheable o no
- Uniform interface
  - Resource-based
    cada recurso se identifica únicamente
  - Manipulation of resources through representations
    las representaciones son uniformes en la respuesta
  - Self-descriptive messages
    cada recurso debe llevar suficiente información para poder procesar el mensaje
  - Hypermedia as the engine of application state (HATEOAS)
    por medio de hipervínculos, el cliente debería poder interactuar con todos los recursos
- Layered system un cliente no debería poder distinguir si hay una implementación por capas, es transparente
- Code on demand (optional) envío de script del servidor al cliente para indicar cómo ejecutar alguna acción

Fuente: *REST API Tutorial | What is REST*

Principios REST

- Client-server deben estar separados y poder evolucionar independientemente
- Stateless cada request del cliente debe contener toda la información necesaria para completarla
- Cacheable una respuesta debería poder indicar si es cacheable o no
- Uniform interface
  - Resource-based
    cada recurso se identifica únicamente
  - Manipulation of resources through representations
    las representaciones son uniformes en la respuesta
  - Self-descriptive messages
    cada recurso debe llevar suficiente información para poder procesar el mensaje
  - Hypermedia as the engine of application state (HATEOAS)
    por medio de hipervínculos, el cliente debería poder interactuar con todos los recursos
- Layered system un cliente no debería poder distinguir si hay una implementación por capas, es transparente
- Code on demand (optional) envío de script del servidor al cliente para indicar cómo ejecutar alguna acción

Fuente: *REST API Tutorial | What is REST*

HTTP stateless

Si una *request* es **independiente de la siguiente**...

¿Cómo podemos **iniciar sesión**
y ser **identificados** en cada *request* a la API?

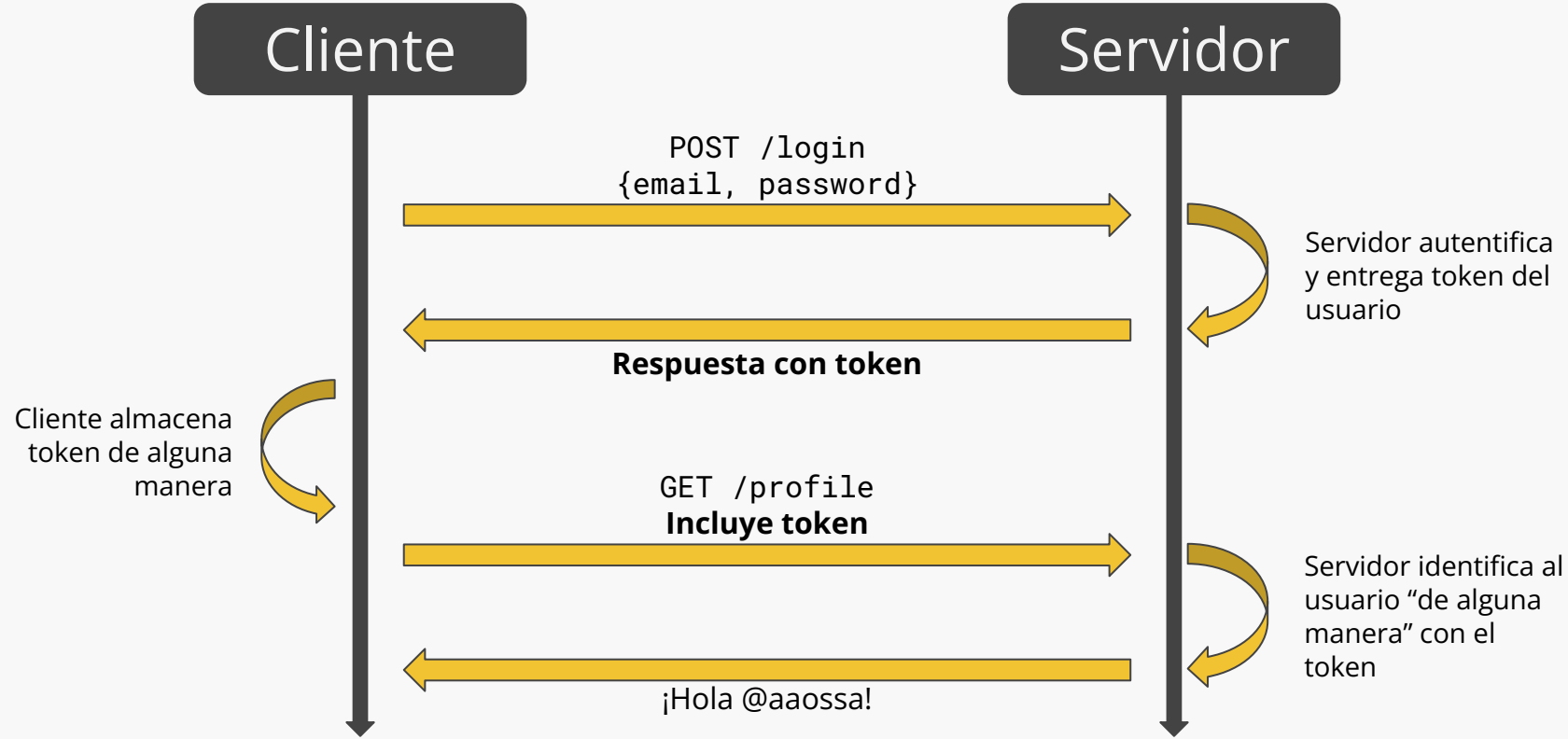
HTTP stateless

Clase 14#*Cookies y sesiones*

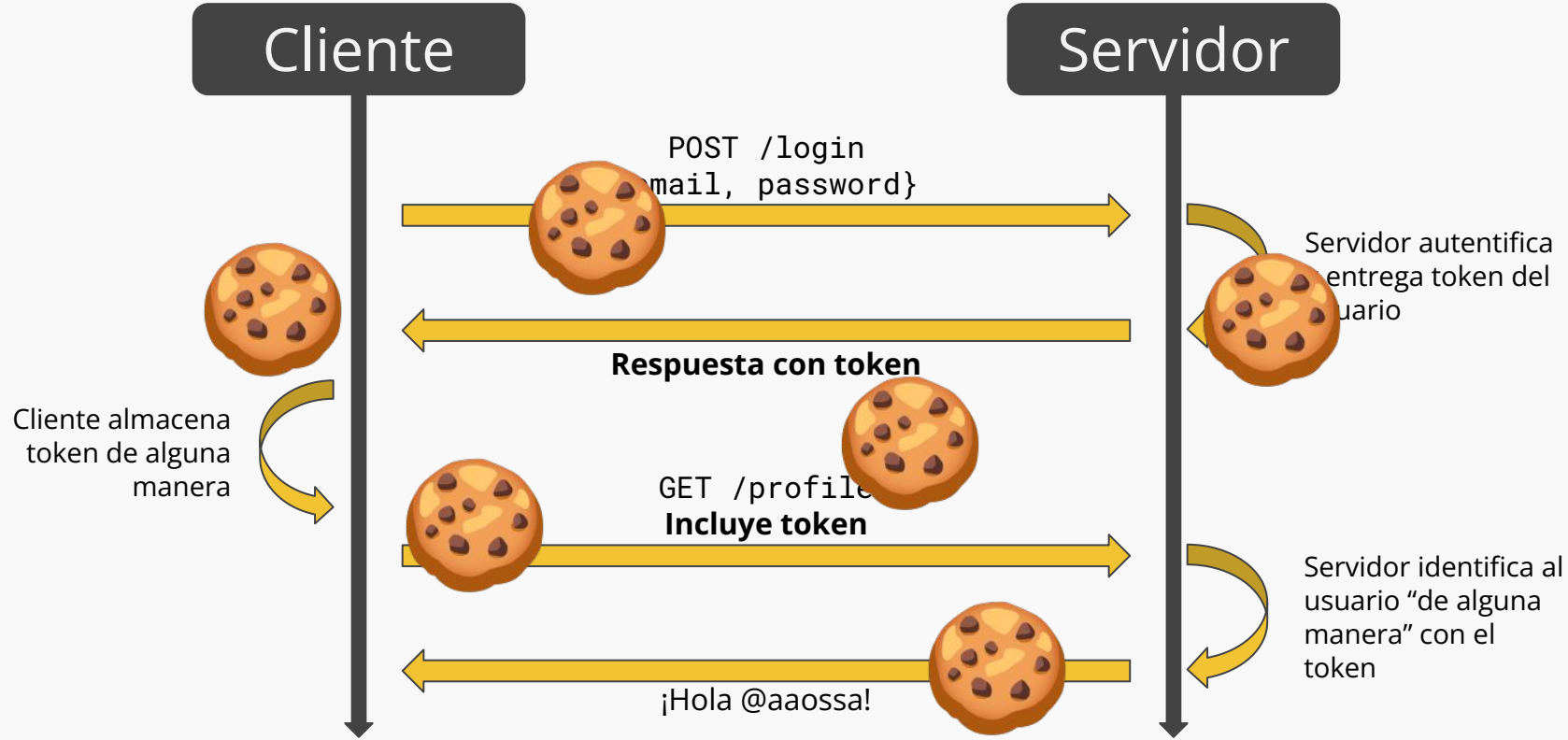
¿Cómo podemos **iniciar sesión**
y ser **identificados** en cada *request* a la API?

Volveremos a visitar esto cuando hablemos de *cookies* y *sessions*

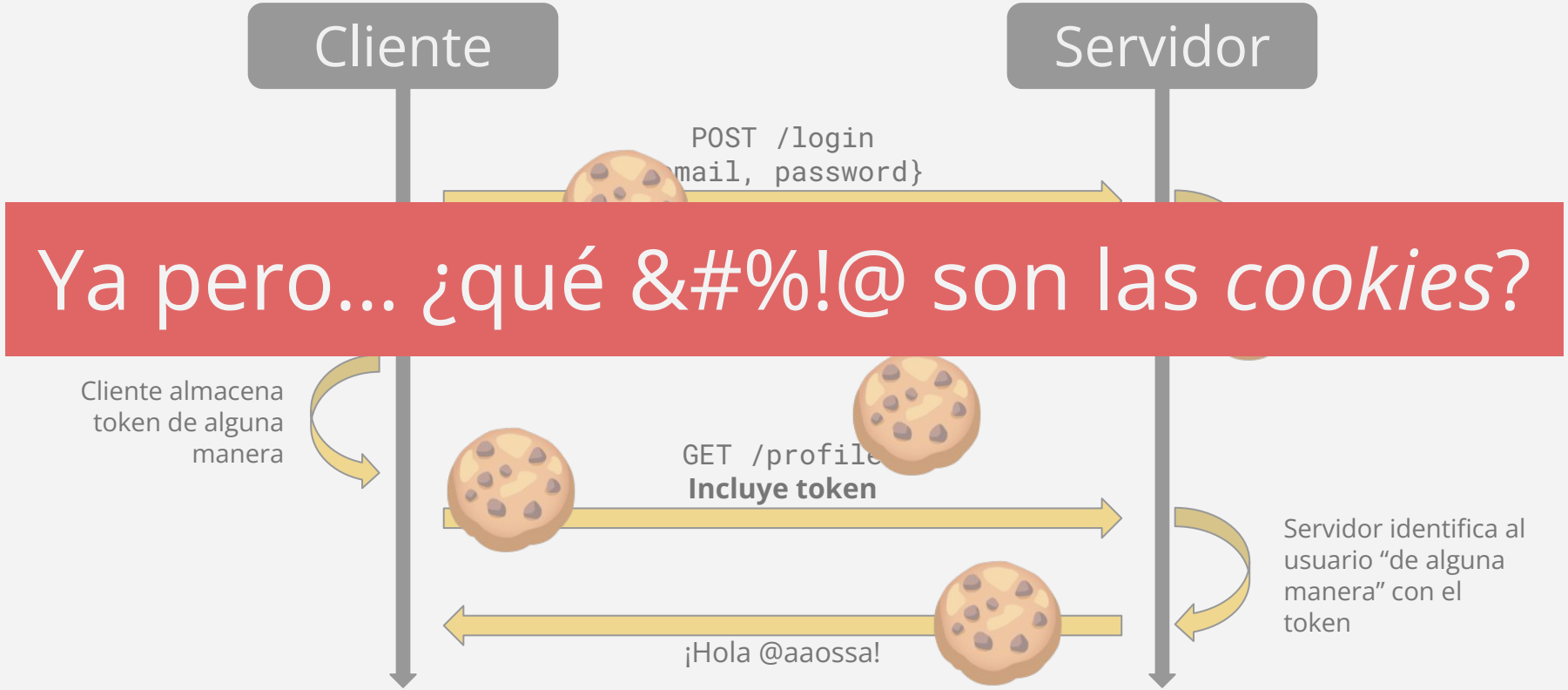
Flujo autenticación



Flujo autenticación



Flujo autenticación



¿Cookies?

Las *cookies* son datos que el **servidor** envía al ***browser***

Objetivo: Mecanismo para que el servidor sepa si **dos requests vienen del mismo cliente**

Algunos usos:

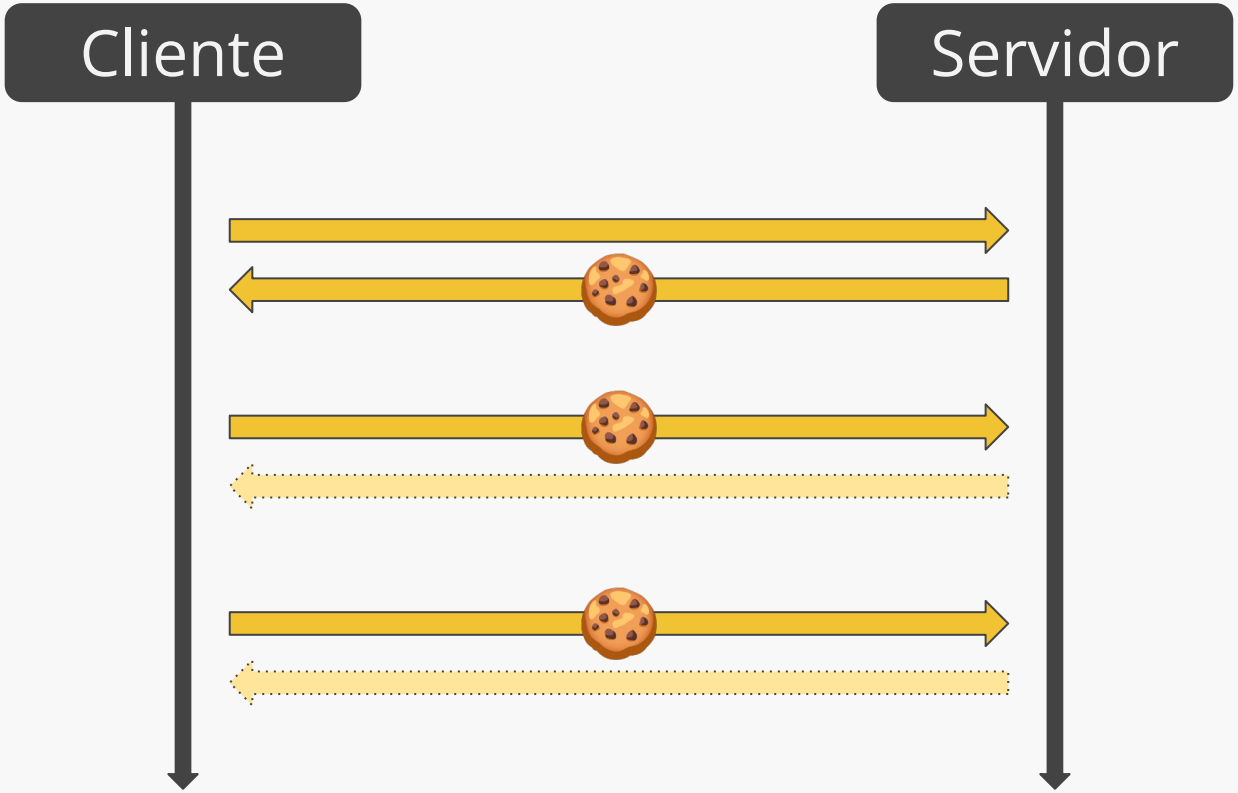
- Manejo de sesión
- Personalización
- Ads
- *Analytics* y *tracking*
- A/B testing en clientes
- Seguridad (como CSRF)
- Coordinación entre accesos
- Recordar preferencias

¿Cómo funcionan?

```
cookieName=cookieValue
```

1. **Servidor envía *cookies*** en *response* y *browser* las almacena
2. *Cookies* son incluidas por *browser* en las **siguientes *requests***

¿Cómo funcionan?



Response headers



Cuando el servidor envía su respuesta HTTP, esta puede incluir el *header* `Set-Cookie` para enviar *cookies* desde el servidor al cliente. Luego, estas *cookies* serán las que el cliente envíe de vuelta al servidor

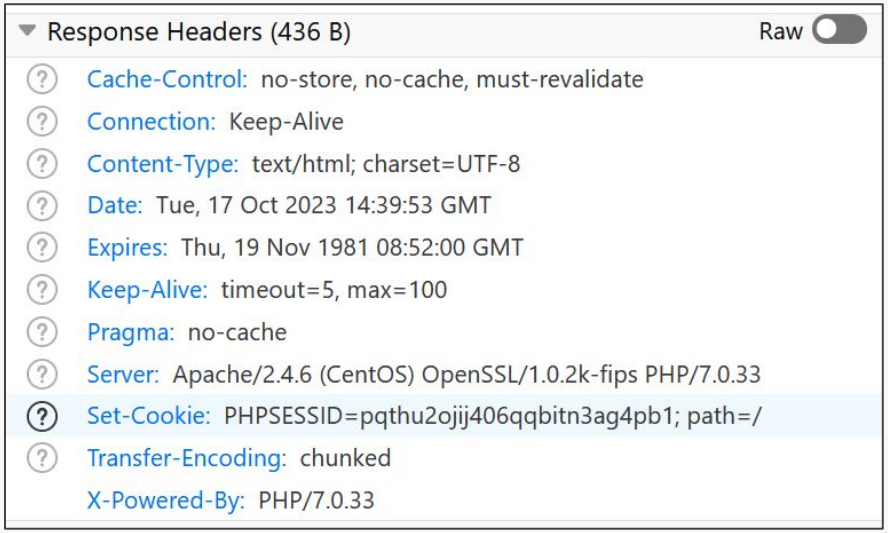
```
Set-Cookie: nombre_cookie=valor_cookie
```

Fuente: *Set-Cookie - HTTP | MDN*

Response headers



Por ejemplo, al solicitar https://buscacursos.uc.cl/ por primera vez o desde un navegador en modo incógnito podremos ver el header `Set-Cookie` que asigna una *cookie* en nuestro navegador



Request headers



El *header* `Cookie` contiene las *cookies* que hayan sido enviadas previamente por el servidor por medio de `Set-Cookie` y almacenadas en el *browser*. Estas *cookies* se incluirán en las *requests* subsecuentes

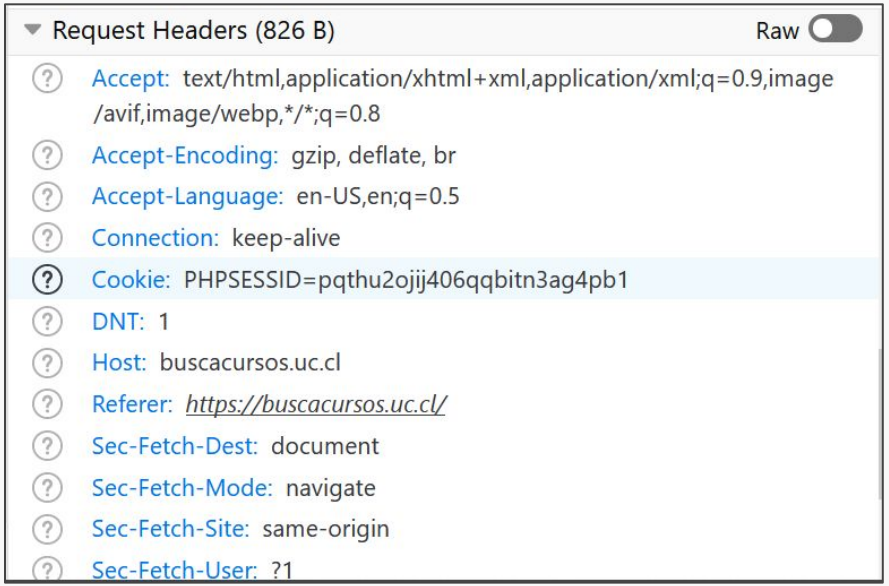
```
Cookie: llame=valor; otra=otro_valor
```

Fuente: *Cookie - HTTP | MDN*

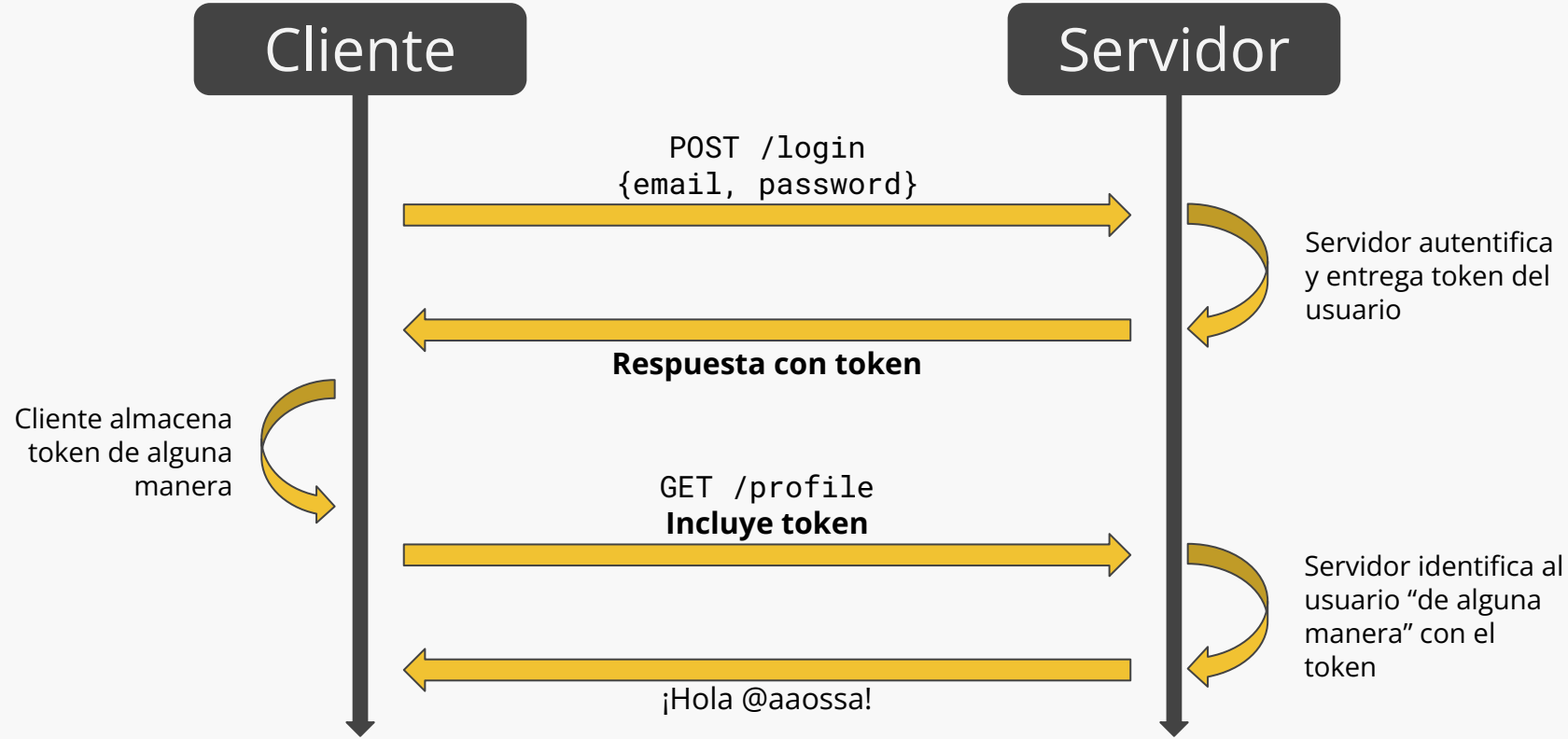
Request headers



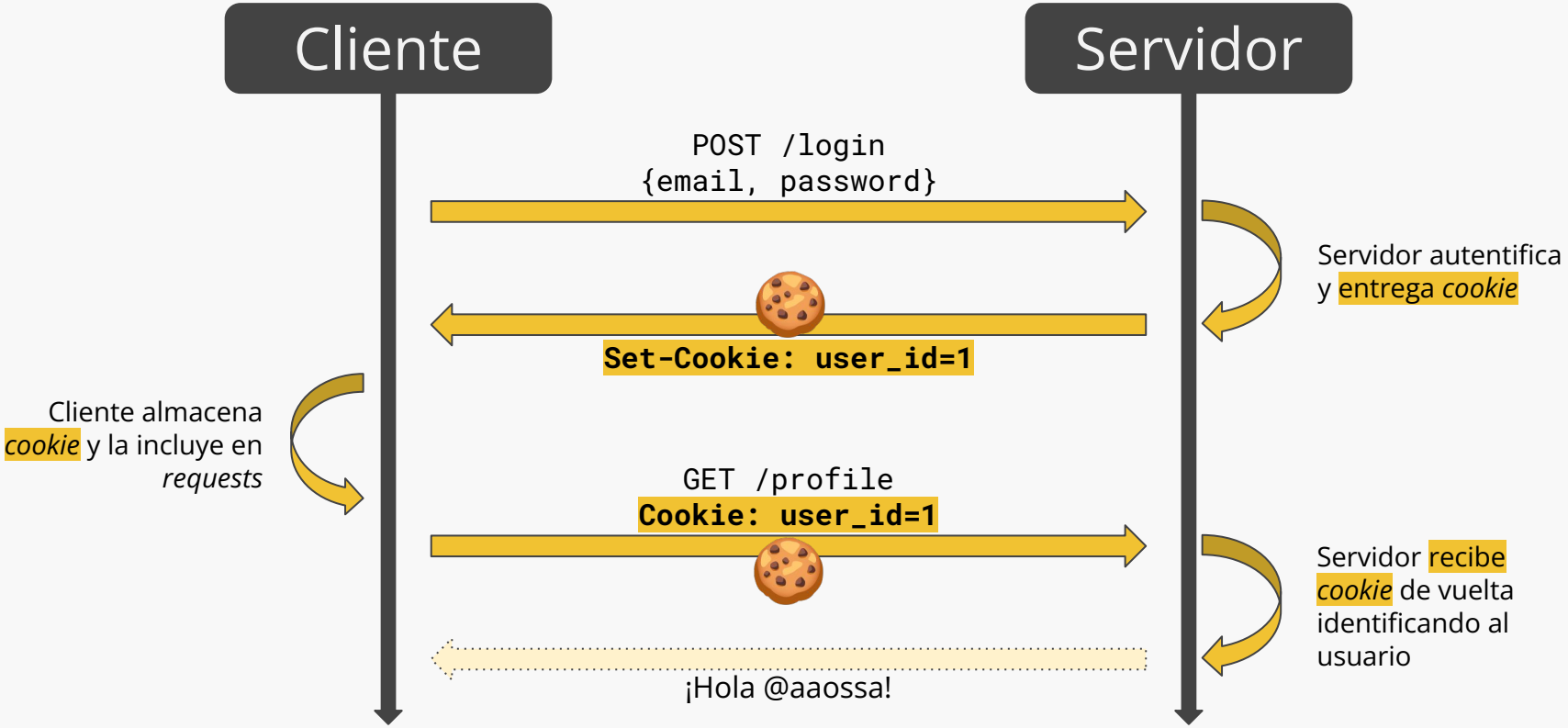
En solicitudes posteriores, por ejemplo al buscar un curso, se incluirá la *cookie* asignada por el servidor como parte de la *request* (en el *header* `Cookie`)



Flujo autenticación



Funcionamiento *headers* para *cookies*



Cookies: manejadas por HTTP *headers*

Response cookie puede incluir una serie de directivas adicionales, tales como:

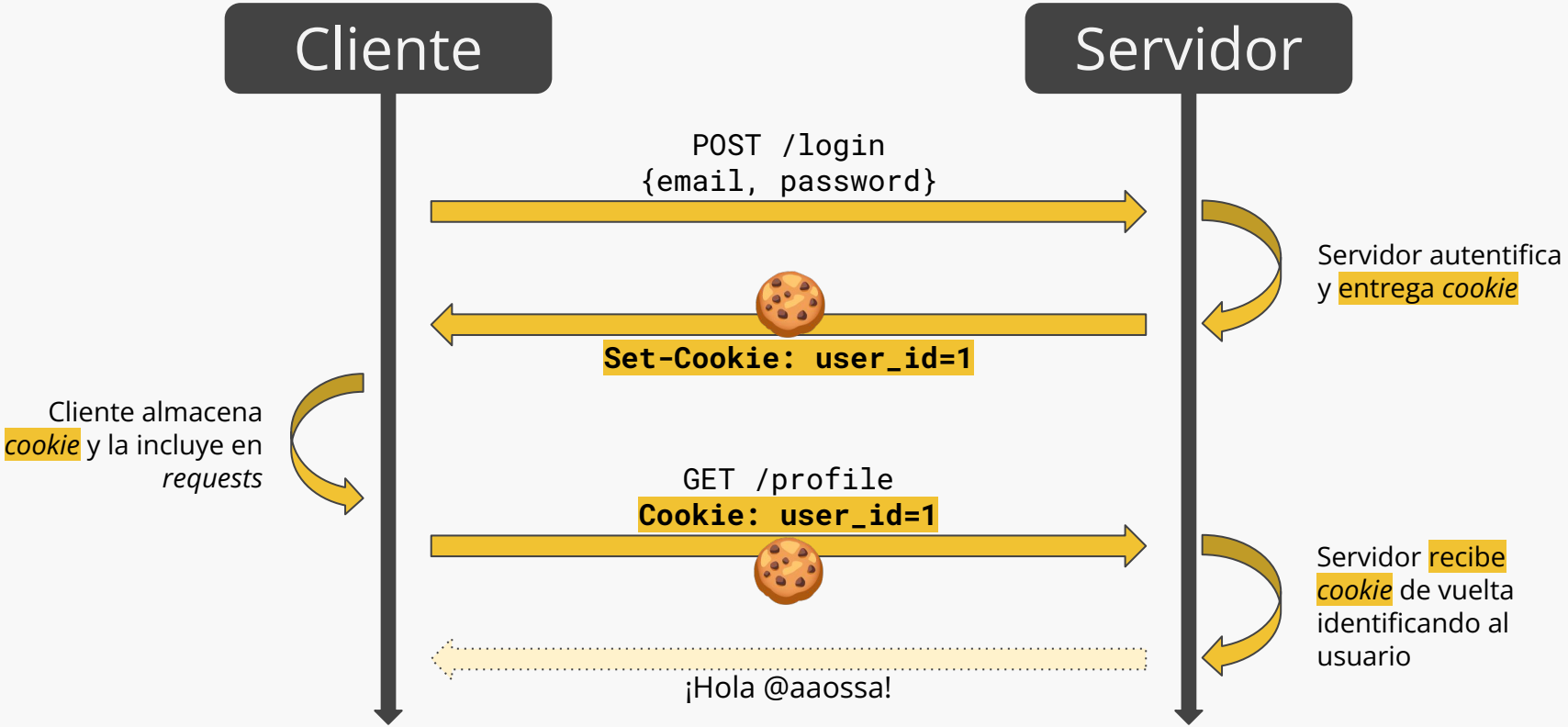
1. `Expires`
   tiempo de vida máximo útil
2. `Domain`
   hosts donde se puede enviar la *cookie*
3. `Secure`
   solo enviar si se usa HTTPS
4. `HttpOnly`
   evita que se pueda acceder desde JS
5. `Path`
   solo incluye *cookie* si estamos en *path*

```
Set-Cookie: cookie=valor
Set-Cookie: cookie=valor; Expires=<date>
Set-Cookie: cookie=valor; Max-Age=<age>
Set-Cookie: cookie=valor; Domain=<domain>
Set-Cookie: cookie=valor; Path=<path>
Set-Cookie: cookie=valor; Secure
Set-Cookie: cookie=valor; HttpOnly
Set-Cookie: cookie=valor; SameSite=Strict

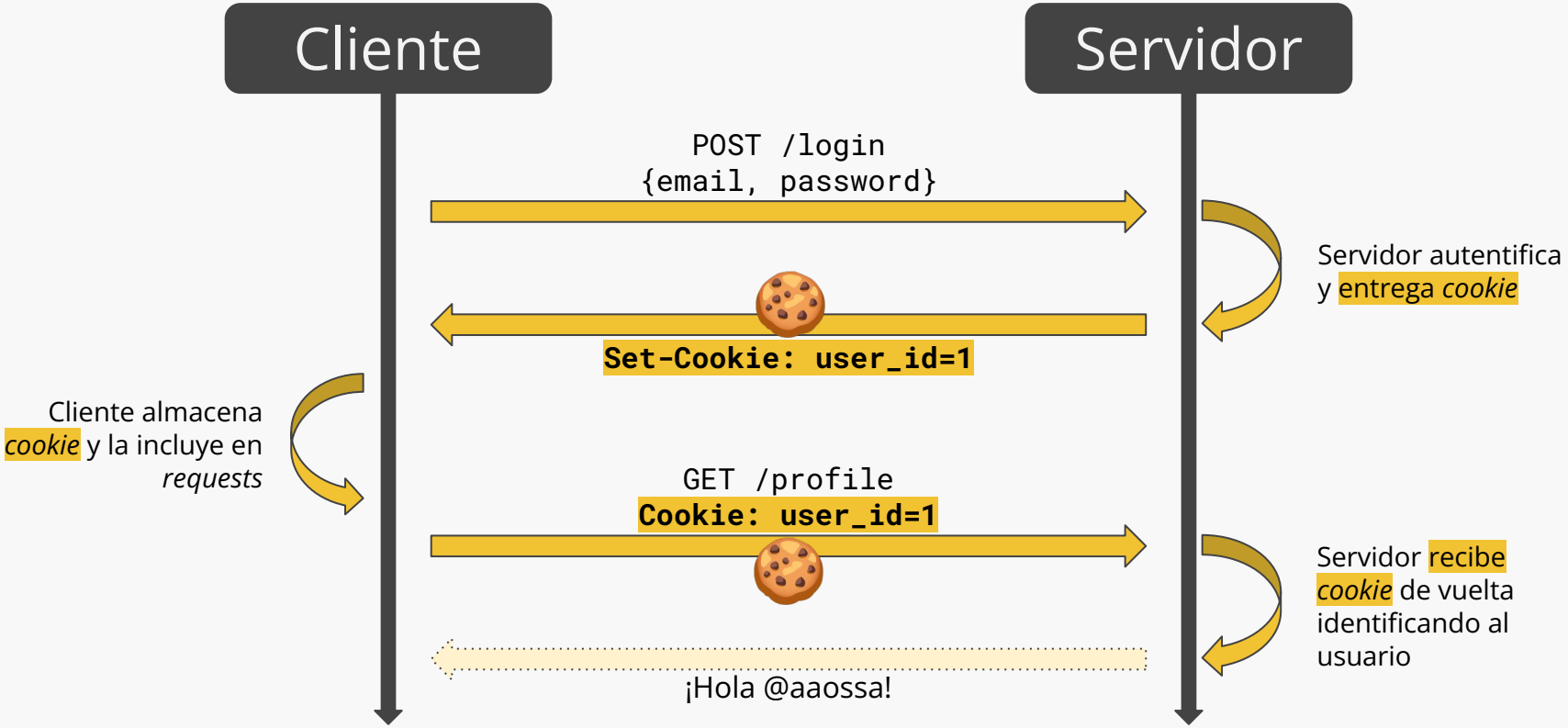
// Es posible usar múltiples directivas
Set-Cookie: cookie=valor;
Domain=<domain-value>; Secure; HttpOnly

// Ejemplo de cookie real
Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct
2021 07:28:00 GMT; Secure; HttpOnly
```

Pregunta#¿Ven alguna vulnerabilidad en este ejemplo?



Pregunta#¿Cómo diferenciar sesiones de un usuario?



Agenda

Cookies HTTP

✨ **Manejando sesiones**

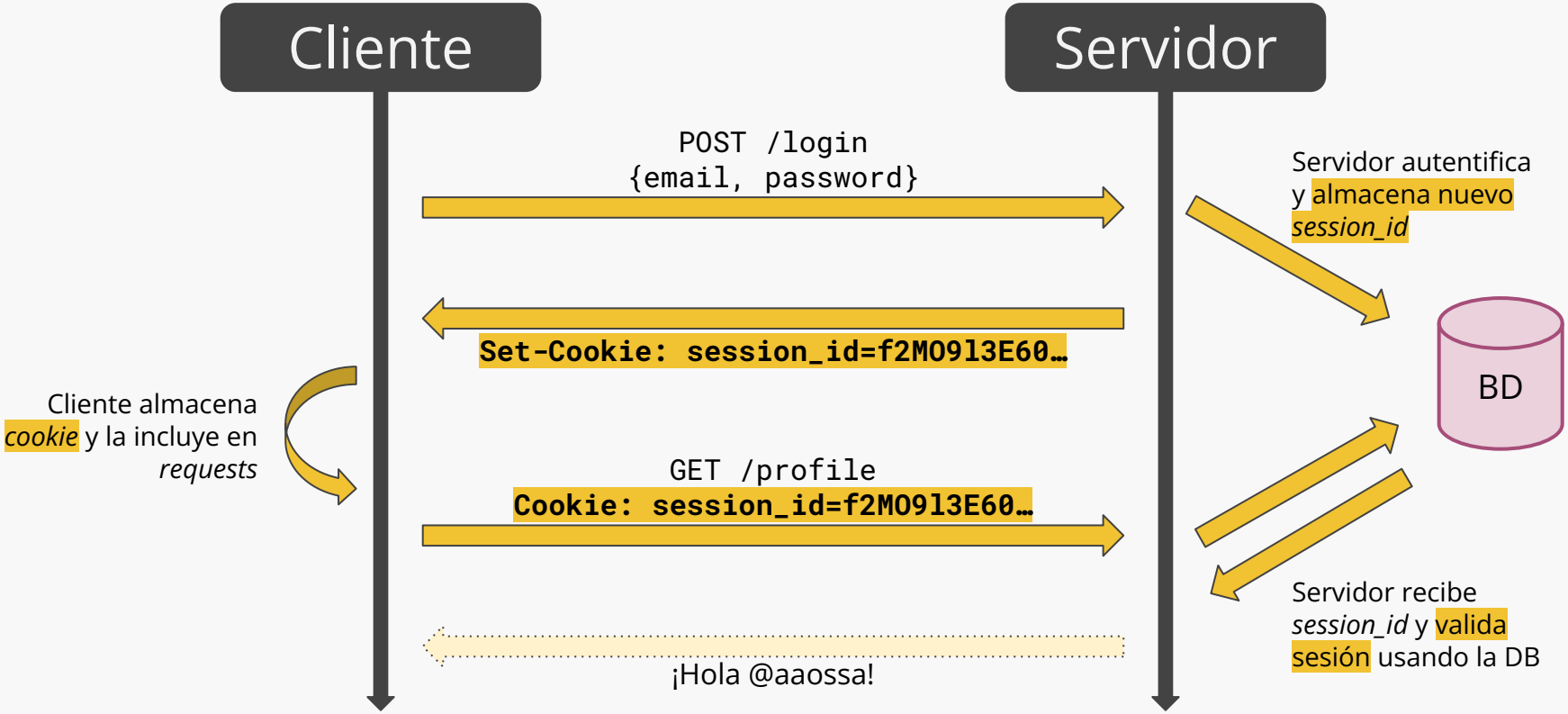
¿Qué es una sesión de usuario?

Forma de **reconocer específicamente** a un usuario en *requests* sucesivos

Estrategias posibles para manejar sesión

1. Enviar datos de sesión (credenciales) en una o más *cookies*
2. Que la *cookie* sólo incluya un identificador (id) de sesión

Funcionamiento de sesión de usuario



¿Diferencia entre *cookies* y *session*?

Si bien tanto *cookies* como *sessions* se utilizan en el contexto de aplicaciones web para almacenar datos a través de distintas páginas de un mismo sitio

La diferencia conceptual es que **las *sessions* son almacenadas en el servidor, mientras que las *cookies* se almacenan en el cliente**

¿Cómo hacerlo en Express?

Utilizamos los objetos de cada *request handler* para trabajar con *cookies*:

```
res.cookie('someKey', 'someValue');
res.send('Cookies set!');
```

De esta manera podemos **leer y escribir propiedades** según necesitemos

¿Cómo hacerlo en koa?

Utilizamos **koa-session**:

```
ctx.session;
ctx.session.someKey = 'someValue';
console.log(ctx.session.someKey);
```

De esta manera podemos **leer y escribir propiedades** según necesitemos

koajs/session: Simple session middleware for Koa:
Código y documentación de koa-session disponible en el repositorio oficial en GitHub

README | Code of conduct | MIT license | Security

koa-session



Simple session middleware for Koa. Defaults to cookie-based sessions and supports external stores.

Requires Node 8.0.0 or greater for async/await support

Installation

```
$ npm install koa-session
```

Notice

6.x changed the default cookie key from `koa:sess` to `koa.sess` to ensure `set-cookie` value valid with HTTP spec.see issue. If you want to be compatible with the previous version, you can manually set `config.key` to `koa:sess` .

Example

View counter example:

Contributors 36



+ 22 contributors

Languages

JavaScript 100.0%

Discusión#¿Cómo se pueden abusar las *cookies*?

Agenda

Cookies HTTP

Manejando sesiones

✨ **¿Qué se viene pronto?**

Próximas fechas importantes: clases y ayudantías

	Clases y ayudantías						
	Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**	**Domingo**
Semana 10 (05/5 - 11/5)		Lanzamiento proyecto (E0)		**Post-receso y repaso**			
Semana 11 (12/5 - 18/5)		HOY		Autenticación y autorización			
Semana 12 (19/5 - 25/5)		Charla GraphQL (Daniela Flores)	Día de las Glorias Navales	*Almacenamiento (cookies, tokens, IndexDB, etc)*			
Semana 13 (26/5 - 01/6)		Tecnologías adicionales		Testing	Sala de ayuda: E1		
Semana 14 (02/6 - 08/6)		Seguridad web		Protocolos seguros			

Próximas fechas importantes: evaluaciones

	Evaluaciones						
	Lunes	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**	**Domingo**
Semana 10 (05/5 - 11/5)	UC: Sin eval.	Inicio E0					
Semana 11 (12/5 - 18/5)		HOY			Fin E0 / Inicio E1		
Semana 12 (19/5 - 25/5)	Control 3		Día de las Glorias Navales				
Semana 13 (26/5 - 01/6)					Fin E1 / Inicio E2		
Semana 14 (02/6 - 08/6)	Control 4						

Clase 14

Cookies y sesiones

IIC2513 - Tecnologías y Aplicaciones Web

Antonio Ossa Guerra
aaossa@ing.puc.cl