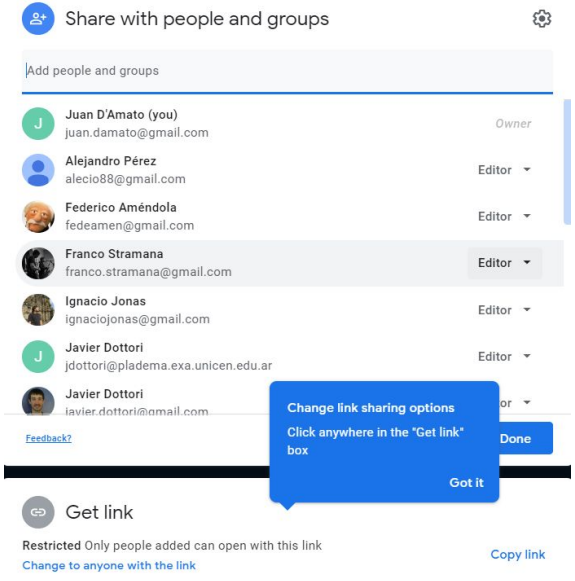
SEGURIDAD

WEB 2 -
TUDAI

Que pasa cuando compartimos la información?

- *Queremos que el acceso sea* ***SEGURO***
- No queremos que todos **modifiquen** datos
- A los que si, le brindamos ***permisos***



¿Qué es **seguridad?**

La **seguridad de la información** es el conjunto de medidas preventivas y reactivas de las organizaciones y de los sistemas tecnológicos que permiten resguardar y *proteger* la información buscando mantener la ***confidencialidad***, *la* ***disponibilidad*** *e* ***integridad*** *de los datos* y de la misma.

Seguridad

La mayoría de los problemas de seguridad en los sitios web se encuentran a **nivel aplicación** y son el resultado de **escritura defectuosa** de código.

Programar aplicaciones **web seguras** requiere tener en cuenta los **riesgos** que puede correr la **información** contenida, solicitada y recibida.

Mecanismos de Seguridad

Asegurar el acceso a la información -> **AUTENTICACIÓN**

Protección de los datos críticos (contraseñas, tarjetas de crédito) -> **ENCRIPTACIÓN**

Comunicación segura -> **PROTOCOLOS**

Encriptación

Es el proceso de hacer **ilegible** información importante. Una vez encriptada sólo puede leerse al aplicar una **clave**.

Encriptación

Codificación

La **codificación** es un proceso de **transformación**, convirtiendo una estructura a otra, de acuerdo a una especificación conocida. No requieren parámetros para aplicarse.

Por ejemplo:

- Image enconder base64. Ver : https://www.base64-image.de/
- SHA1
- Video encoders/decoders

Por qué encriptar?

Para mantener la clave “oculta” y que ni otros *usuarios*, ni el *administrador* del sistema pueda conocerla

https://bcrypt-generator.com/

Importante!!!! Ahora en nuestra BBDD vamos a guardar la contraseña “encriptada”

Encriptación - Protocolos

MD5 (Message-Digest Algorithm 5): Es una función hash de 128 bits. Es usado para **firmas digitales**.

MD5("") = d41d8cd98f00b204e9800998ecf8427e

SHA (Secure Hash Algorithm): Familia de funciones hash de cifrado. SHA-0 y SHA-1 producen una salida de 160 bits (20 bytes) de un mensaje que puede tener un tamaño máximo de 2**64 bits. Es parecido al MD5 pero la función de **compresión** es más **compleja**.

Encriptación en PHP

En PHP contamos con diferentes funciones de encriptación.

```php
<?php
$clave = "12345";
$clave_encriptada = password_hash ($clave , PASSWORD_DEFAULT );
echo "La clave $clave encriptada es la siguiente: $clave_encriptada";
?>
```

Encriptando “12345” con php se obtiene:

$2y$12$vcP6PqTDVHRswJcqzpIpwu5LEbs3TT8hO6ZzppzPQIuXOgcBlW5Ta

DEMO 1

Encriptación - Ataques

Para una clave de 12 dígitos, escrita con un teclado con 97 caracteres, en un ataque por fuerza bruta habría que realizar:

9712 = 693.842.360.995.438.000.295.041** comprobaciones.

- Para MD5, la salida es de 128 bits, sería necesario realizar:

2128=3,402823669 * 10**38 operaciones.**

Con ataques basados en búsqueda de colisiones:

- Para MD5, la salida es de 128 bits, hay que operar sobre la mitad de bits:

264=18.446.744.073.709.551.616 operaciones.**

- Para el algoritmo SHA 1, cuya salida es de 160 bits:

280=1.208.925.819.614.629.174.706.176 operaciones.**

100.000 de procesadores de 1 Ghz tardarían más de 38.000 años !!!

Autenticación

Autentificación / Autenticación

Es el proceso de **verificar la identidad** digital del remitente de una comunicación como una petición para conectarse.

El remitente autenticado puede ser una **persona** que usa un dispositivo, un **dispositivo** por sí mismo o un **programa** del dispositivo.

Autorización

Proceso por el cual la red de datos **autoriza** al usuario identificado a **acceder** a determinados recursos de la misma.

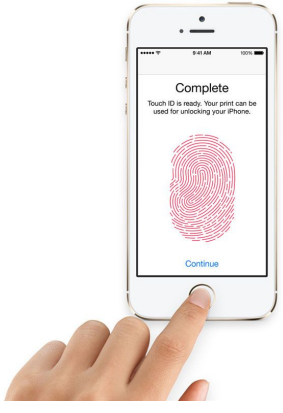
Un **ROL** podría estar compuesto por niveles de autorización dentro del sistema.

Mecanismos de autenticación de un usuario

Por password

Por token

Por datos biométricos



Cómo modelamos en la BBDD la autenticacion?

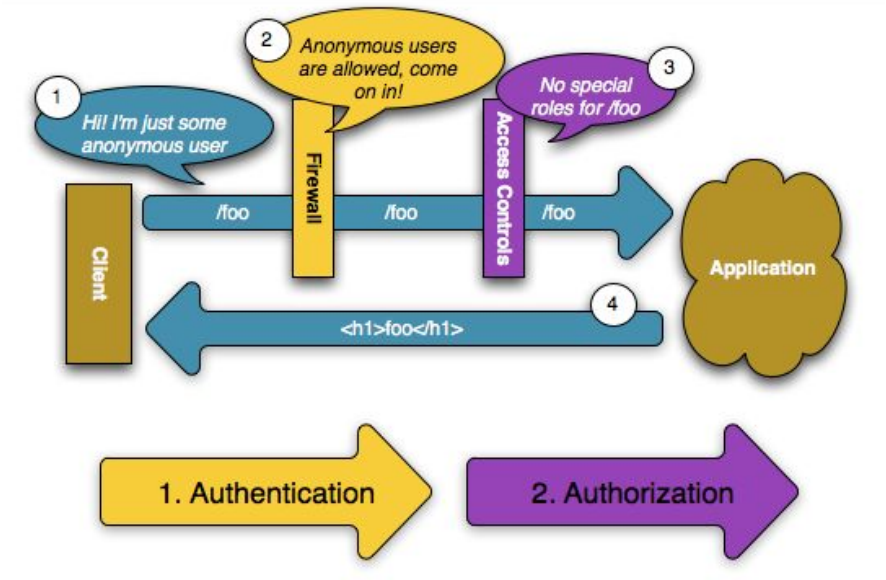
- Tabla de usuarios
  - Id
  - Email
  - Password
  - rol

- El passWord va a estar **encriptado**
- Dado un usuario, verificamos que el password coincida

Autenticación - Autorización

1. Conocer qué usuario
2. Verificar sus datos (UP)
3. Dar acceso al RECURSO solicitado.

(Role based Access control)



Implementar un mecanismo de autenticación?

- Recordemos : HTTP es “stateLess”
- Tenemos que implementar nuestro mecanismo:
  - Por **manejo de sesiones**
  - Por **API**

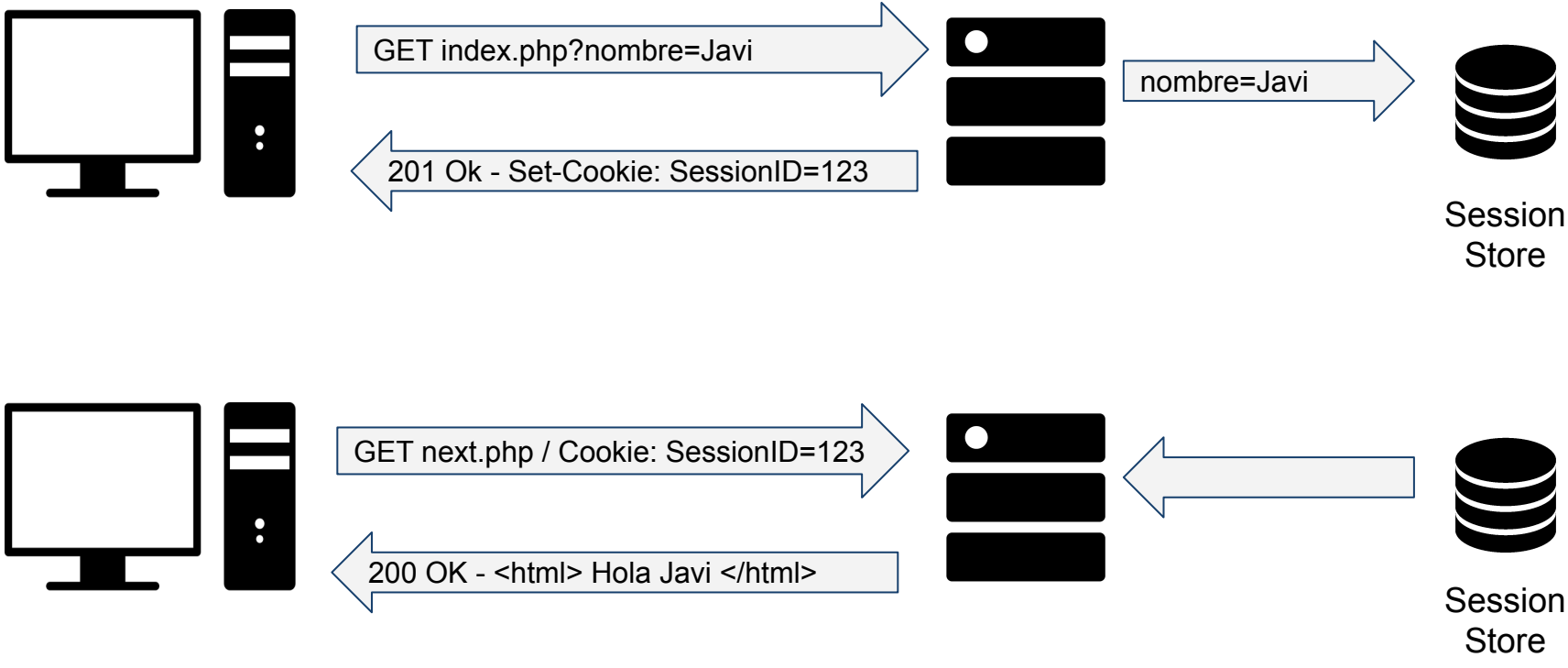
Por manejo de sesiones (cont.)

Utilizado para guardar información (estados) a través de los *requests* que un usuario hace durante su visita a un sitio web o aplicación.

La información guardada en una **sesión** puede llamarse en cualquier momento mientras la sesión esté abierta.

- La sesión **vive del lado del servidor**
- Es una pequeña porción de información que se guarda en el server
- Dura **mientras el usuario está conectado** al server
- Más seguro y confiable

Cómo funciona?



Por manejo de sesiones (cont.)

- Antes de abrir una sesión, el usuario deberá **autenticarse** (user + pass)
  - Pueden existir usuarios anónimos, que no lo requieren, pero debería ver qué permisos tiene
- Debo destruir la sesión al hacer logout
  - En caso que el usuario nunca hizo “logout”, deberíamos hacer que la sesión se cierre después de un tiempo

Manejo de Sesión en PHP

Con la sesión recuerdo el usuario para poder saber si estaba logueado.

<?php

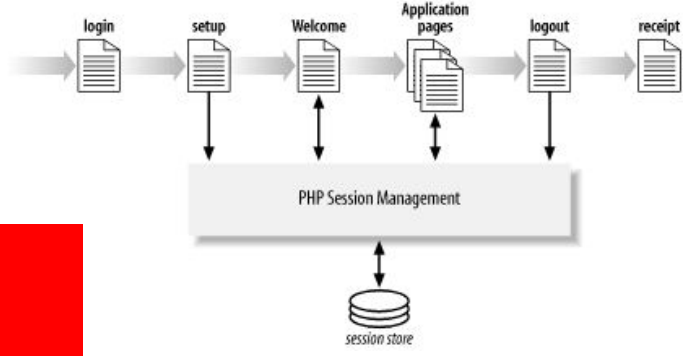
```
session_start();
// sesión iniciada
```

- Crea una sesión en el servidor, si ya existe trae la existente.
- Debe **llamarse siempre** antes de acceder/almacenar algún dato.

Por manejo de sesiones

- En el servidor se lleva cuenta de datos del usuario
  - EN PHP, se usa la variable

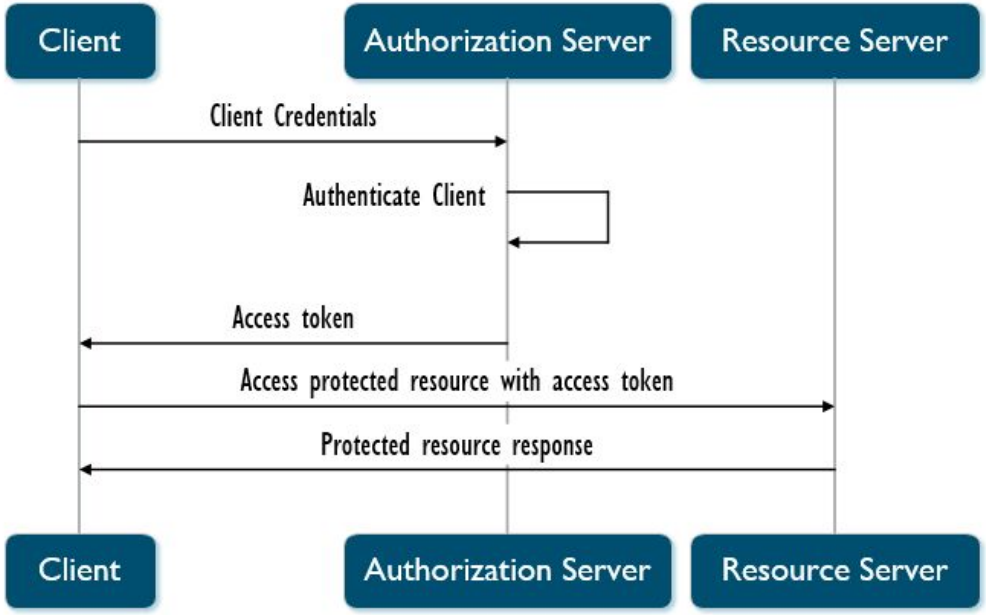
    ```
    $_SESSION["name"] = $username;
    ```
  - Al autenticar
  - `$_SESSION['logged'] = true;`
- Puedo permitir acciones, mientra tenga una “Sesión activa”
- ```
  if (isset($_SESSION['logged']) && $_SESSION['logged'] == true)
  ```
- ```
  { ..... DO EVERYTHING .....}
  ```



DEMO 2

Por API

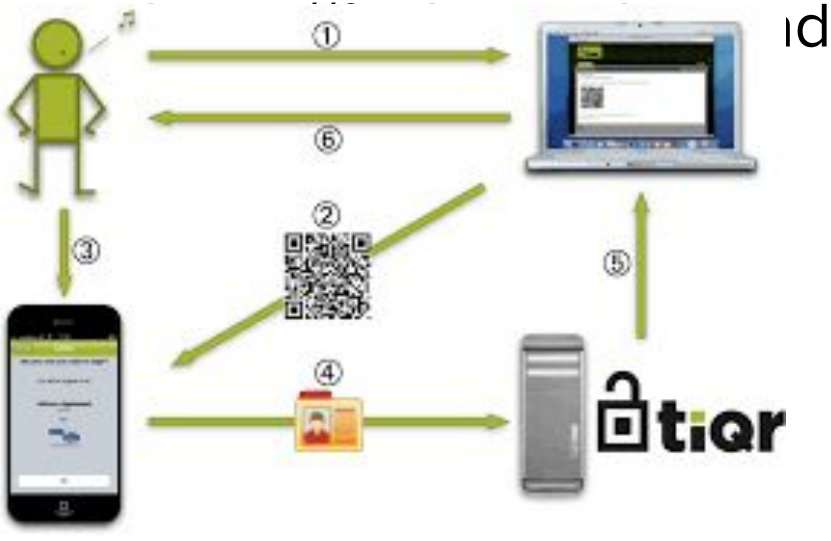
- Se realiza a través de “Tokens”
- Por cada conexión el usuario, pide un “token” y lo utiliza en cada petición
  - No hay una sesión en el servidor



Por API (cont.)

- El servidor si tiene que implementar un “manejador” de tokens
  - Puede ser local en memoria -> similar a una sesión
  - En la BBDD (puede ser distribuido)
- La API debe modificarse para soportar el acceso por token
- El cliente tiene que también manejar el token y
  -

Plataformas de autenticación



Top 10 de Amenazas - OWASP

OWASP Top 10 – 2010 (Previo)	OWASP Top 10 – 2013 (Nuevo)
A1 – Inyección	A1 – Inyección
A3 – Pérdida de Autenticación y Gestión de Sesiones	A2 – Pérdida de Autenticación y Gestión de Sesiones
A2 – Secuencia de Comandos en Sitios Cruzados (XSS)	A3 – Secuencia de Comandos en Sitios Cruzados (XSS)
A4 – Referencia Directa Insegura a Objetos	A4 – Referencia Directa Insegura a Objetos
A6 – Defectuosa Configuración de Seguridad	A5 – Configuración de Seguridad Incorrecta
A7 – Almacenamiento Criptográfico Inseguro – Fusionada A9→	A6 – Exposición de Datos Sensibles
A8 – Falla de Restricción de Acceso a URL – Ampliada en →	A7 – Ausencia de Control de Acceso a las Funciones
A5 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)
<dentro de A6: – Defectuosa Configuración de Seguridad>	A9 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Redirecciones y reenvíos no validados	A10 – Redirecciones y reenvíos no validados
A9 – Protección Insuficiente en la Capa de Transporte	Fusionada con 2010-A7 en la nueva 2013-A6

Top 10 de Amenazas - OWASP

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

A1 - Inyección SQL

Las fallas de inyección ocurren cuando datos no confiables son **enviados** a un intérprete como parte de un comando o consulta.

Instrucción SQL vulnerable:

```
String query = "SELECT * FROM account WHERE user = '" + userName + "';"
```

Ingreso: Pepe