

SGSSI:
Aspectos teóricos y prácticos del
almacenamiento de contraseñas en bases
de datos

Mikel Iturria

Noviembre-Diciembre 2021

Contents

1	Introducción	4
2	¿Por qué encriptar las claves guardadas?	4
3	Recomendación de manejo de <i>Logins</i>	4
3.1	Ventajas y desventajas del login con aplicaciones de terceros . . .	5
3.2	Ejemplos de webs con inicio de sesión con aplicaciones de terceros	6
4	Aspectos teóricos de la encriptación de contraseñas	8
4.1	Tipos de encriptación de claves	8
4.1.1	Problemas de encriptar las contraseñas usando solamente la encriptación básica	8
4.1.2	Cifrado de contraseñas usando “salting”	9
4.2	Cifrado de contraseñas usando <i>MySQL</i>	10
4.3	Cifrado de contraseñas usando <i>PHP</i>	11
5	Puesta en práctica del encriptado de contraseñas en inicios de sesión	12
5.1	Preparación de la práctica	12
5.2	Práctica encriptando la contraseña en <i>MySQL</i>	13
5.3	Práctica encriptando la contraseña en <i>PHP</i> con sal predefinida .	15
5.4	Práctica encriptando la contraseña en <i>PHP</i> con sal aleatoria . .	17
6	Conclusiones	20
7	Referencias	20

List of Figures

1	El <i>Login</i> de Google devuelve el email del usuario	5
2	<i>Login</i> mixto, de forma tradicional mas <i>Facebook</i> y <i>Google</i>	6
3	<i>Login</i> mixto, de forma tradicional que además de incluir <i>Facebook</i> y <i>Google</i> incluye también <i>Twitter</i> y <i>VK</i>	7
4	<i>Login</i> únicamente social mediante <i>Facebook</i> , <i>Google</i> o <i>VK</i>	7
5	<i>Login</i> únicamente tradicional	8
6	Variable establecida correctamente	10
7	Resultado de la funcion sobre 'sgssi2021'	11
8	Subrayado en rojo, dónde clicar para crear la BD.	12
9	Subrayado en rojo, el nombre a poner para crear la BD.	12
10	Se ha creado la tabla en la BD.	13
11	Subrayado en rojo, la función PASSWORD.	13
12	Subrayado en rojo, la función PASSWORD.	14
13	Valores introducidos para la prueba de registro	14
14	Datos en la base de datos	14

15	Valores introducidos para la prueba de Log in	15
16	Resultado de hacer Log in	15
17	Líneas de código donde se cifra la contraseña.	16
18	Valores introducidos para la prueba de registro	16
19	Datos en la base de datos	16
20	Valores introducidos para la prueba de Log in	17
21	Resultado de hacer Log in	17
22	Líneas de código donde se cifra la contraseña.	18
23	Valores introducidos para la prueba de registro	18
24	Datos en la base de datos	18
25	Llamada a la base de datos	19
26	Uso de la función password_verify()	19
27	Valores introducidos para la prueba de Log in	19
28	Resultado de hacer Log in	20
29	La contraseña es distinta a la que aparece en la figura 24	20

1 Introducción

Esta práctica opcional para la asignatura de SGSSI se centrará en la importancia de cómo guardar las contraseñas de inicio de sesión de una página web en la base de datos de la manera mas segura posible. El documento tendrá dos partes. La primera parte será una parte teórica donde se mencionarán los aspectos teóricos del almacenamiento de las contraseñas (la recomendación de las entidades expertas, las diferentes opciones de guardarlos...) y en la segunda parte, se realizará una pequeña práctica en la que se mostrará cómo guardar las contraseñas cifradas usando distintas herramientas en una base de datos *MySQL*.

2 ¿Por qué encriptar las claves guardadas?

Para empezar, nosotros como desarrolladores web tenemos la **obligación moral y ética** de que las contraseñas que nos *confían* nuestros usuarios, que de acuerdo con numerosos estudios [1] [2] a menudo suelen ser las mismas contraseñas que tienen en otros sitios web, nunca se vean comprometidas. Es nuestra responsabilidad asegurarnos que un ataque a nuestra base de datos no ponga en riesgo las contraseñas y cuentas que tengan nuestros usuarios en otros sitios web.

Si por algún caso los valores éticos y morales no te hacen creer que es necesario el cifrado de contraseñas, has de saber que tienes la **obligación legal** de cifrar las contraseñas y es que, incluso antes de la entrada en vigor de la RGPD¹ en 2018, la LOPD² ya imponía la obligación de cifrar la información que contenga datos de carácter personal. En concreto, el artículo 93.3 del Reglamento que desarrolla la LOPD especifica que las "*contraseñas deben almacenarse de forma ininteligible*" [3] y la mejor forma para eso es cifrando las contraseñas. Cabe recordar que las empresas que incumplan el RGPD en la vulneración de datos personales podrían ser sancionados con hasta 20 millones de euros o el 4% de la facturación [4].

3 Recomendación de manejo de *Logins*

La recomendación actualmente acerca del manejo de *logins* para una página web es "deslocalizar" el sistema de inicios de sesión a empresas de terceros como podrían ser *Google* o *Facebook* entre otros. Tomar esta decisión tiene numerosas ventajas pero también cuenta con alguna desventaja que detallaré a continuación.

¹Reglamento Europeo de Protección de Datos

²La Ley Orgánica de Protección de Datos

3.1 Ventajas y desventajas del login con aplicaciones de terceros

Las **ventajas** del uso de aplicaciones de inicio de sesión de terceros son notables:

- La primera ventaja es la **comodidad**. Para el usuario es mucho mas cómodo usar el inicio de sesión con su cuenta de *Google* que tener que estar memorizando usuarios y contraseñas de páginas web distintas.
- Otra ventaja es la **confianza** que produce al usuario. El usuario confiará más en una página web en la que pueda realizar un login social que en una en la que no pueda realizarla, ya que al poder iniciar sesión con aplicaciones de terceros no tendrá que introducir una contraseña y en consecuencia no tendrá el miedo a que su contraseña se vea comprometida.
- Aunque, sin duda, la **la mayor ventaja** para nosotros como desarrolladores web es que **evitamos guardar material sensible** cumpliendo así una de las premisas de la privacidad y la protección de datos que es *"guardar solo los datos sensibles que sean imprescindible y evitar guardar datos que no se necesiten"*. Con el inicio de sesión con webs de terceros nos prevenimos un problema que pudiera suceder en el futuro.

Pese a tener numerosas ventajas, también tiene alguna **desventaja** como por ejemplo que generas una **dependencia muy fuerte** para con un servicio de terceros. Eso genera inequívocamente un **riesgo** que es, ¿Qué pasaría si el servicio de terceros fallase? Además, actualmente los servicios de inicio de sesión de *Google* y de *Facebook* son gratuitos pero, ¿Qué sucedería si en futuro dichos servicios fueran de pago y no quisieras pagarlos? Son sin duda aspectos que hay que tener en cuenta. Una **gestión del riesgo** podría ser guardar el email del usuario que proporcionan los logins sociales. Así, si en un futuro no puedes seguir usando el login de terceros, al tener el correo del usuario bastará con enviarle una contraseña al mail y el usuario ya dispondrá de login.

```
function onSignIn(googleUser) {  
  // Useful data for your client-side scripts:  
  var profile = googleUser.getBasicProfile();  
  console.log("ID: " + profile.getId()); // Don't send this direc  
  console.log('Full Name: ' + profile.getName());  
  console.log('Given Name: ' + profile.getGivenName());  
  console.log('Family Name: ' + profile.getFamilyName());  
  console.log("Image URL: " + profile.getImageUrl());  
  console.log("Email: " + profile.getEmail());  
  
  // The ID token you need to pass to your backend:  
  var id_token = googleUser.getAuthResponse().id_token;  
  console.log("ID Token: " + id_token);  
}
```

Figure 1: El *Login* de Google devuelve el email del usuario

3.2 Ejemplos de webs con inicio de sesión con aplicaciones de terceros

Numerosas páginas web usan hoy en día distintos tipos de login sociales. La mayoría usan un sistema mixto de login tradicional (usuario + contraseña) con la posibilidad de login social pero hay algunas que únicamente tienen login social. Vamos con algunos ejemplos.

- Login de *000 WebHost*³

The image shows a login interface for 000 WebHost. It features a traditional login form with an 'Email' field containing 'you@example.com' and a 'Password' field with the placeholder 'Enter 6 characters or more'. A 'Forgot password?' link is located next to the password field. Below the password field is a prominent red 'LOG IN' button. Underneath the login form, there is a horizontal line with the word 'OR' in the center. Below this line are three social login buttons: a purple button with the Hostinger logo and the text 'UPGRADE TO HOSTINGER', a dark blue button with the Facebook logo and the text 'LOG IN WITH FACEBOOK', and a blue button with the Google logo and the text 'LOG IN WITH GOOGLE'.

Figure 2: *Login* mixto, de forma tradicional mas *Facebook* y *Google*

³<https://www.000webhost.com/cpanel-login>

- Login de *AliExpress* ⁴



The image shows the AliExpress login page. At the top is the AliExpress logo in red. Below it are two links: 'Regístrate' and 'Inicia sesión', with 'Inicia sesión' being the active one. There are two input fields: the first is for 'dirección de correo electrónico o ID de miemb...' and the second is for 'Contraseña' with an eye icon. Below the password field is a link 'Olvidaste tu contraseña? Con número de teléfono'. A large orange button labeled 'Iniciar sesión' is in the center. At the bottom, there's a section 'Acceso rápido con' followed by icons for Facebook, Google, Twitter, and VK. A 'Mostrar todo' link is below the icons.

Figure 3: *Login* mixto, de forma tradicional que además de incluir *Facebook* y *Google* incluye también *Twitter* y *VK*

- Login de *RivalRegions*⁵ con **únicamente** login social.

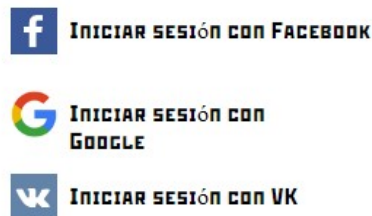


Figure 4: *Login* únicamente social mediante *Facebook*, *Google* o *VK*

Evidentemente, este tipo de *Logins* sociales **NO** se recomienda **para uso interno de empresas o intranets**. En ese caso es recomendable usar *logins*

⁴https://login.aliexpress.com/?return_url=https%3A%2F%2Fwww.aliexpress.com%2Faccount%2Findex.html

⁵<https://rivalregions.com/>

tradicionales para tener una mayor control de los usuarios. Un ejemplo sería la *Federación Guipuzcoana de Fútbol*⁶.

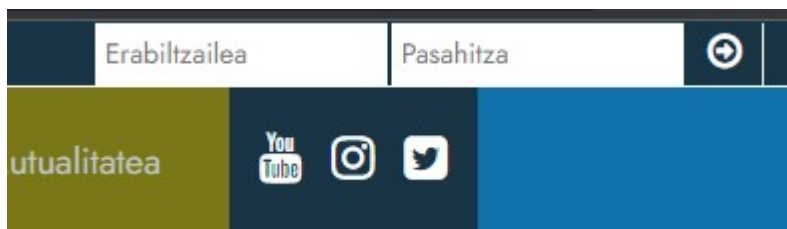


Figure 5: *Login* únicamente tradicional

4 Aspectos teóricos de la encriptación de contraseñas

4.1 Tipos de encriptación de claves

4.1.1 Problemas de encriptar las contraseñas usando solamente la encriptación básica

Tal y como hemos visto en el Apartado 2, hoy en día es imprescindible tanto a nivel ético como a nivel legal almacenar las contraseñas criptografiadas pero, ¿Es suficiente con usar la encriptación básica? ¿Es completamente segura una contraseña por el mero hecho de estar encriptada? La respuesta es **no**.

Aunque a priori nos pueda parecer que una contraseña encriptada es indescifrable porque computacionalmente es muy costoso desencriptar una contraseña, la realidad es que la contraseña sigue siendo **vulnerable** aunque, por supuesto, en mucha menor medida que si no encriptáramos las contraseñas. La cuestión es que si únicamente aplicamos un encriptado básico, la contraseña puede ser descubierta usando **la fuerza bruta**. Basta con encontrar una cadena de caracteres que aplicada la misma función hash que se le aplicó a la contraseña encriptada que se ha logrado desde la base de datos, se obtenga el mismo resultado que dicha contraseña filtrada. Pongamos un ejemplo práctico.

Supongamos que un "hacker" logra obtener de la base de datos que el usuario "Pepe" tiene como contraseña "f17ed833132a190c1907db8bbdf19e39" y sabe que esa contraseña ha sido encriptada usando MD5. Si bien el malhechor no sabe cual es la contraseña de Pepe, sabe que cuál es el resultado que se obtiene al aplicar MD5 a su contraseña, solo necesita realizar pruebas.

⁶<https://gipuzkoafutbola.eus/>

Así que el "hacker" empieza a realizar distintas pruebas de cadenas de caracteres que opina que son buenas candidatas para contraseña⁷:

Cadena de intento	Resultado al aplicar MD5	¿Coincide con CLA?
Prueba	c893bad68927b457dbed39460e6afd62	NO
1234	81dc9bdb52d04dc20036dbd8313ed055	NO
botella	f17ed833132a190c1907db8bbdf19e39	SÍ

Como podemos observar, la cadena "botella" ha logrado el mismo resultado que se había obtenido de la base de datos comprometida con lo que el malhechor puede asegurar de manera irrefutable que la contraseña de Pepe es "botella". Ahora bien, obtener las contraseñas por fuerza bruta no es tan sencillo como parece, en este caso el "hacker" ha conseguido encontrar la contraseña al tercer intento pero lo mas normal sería que le llevara cientos de miles o incluso millones de intentos obtenerla, con el gasto computacional que ello supone. Con lo cual, pese a que las contraseñas criptografiadas sean vulnerables, siguen siendo mucho mas seguras que las no encriptadas.

Para añadir mas seguridad a la encriptación de contraseñas, se puede usar un método llamado *salting* que hace que sea imposible la reidentificación de la contraseña, evitando así que la contraseña se vea comprometida por el uso de fuerza bruta.

4.1.2 Cifrado de contraseñas usando "salting"

El uso de *salting* es una práctica común hoy en día. Incluso la propia AEPD menciona el uso de salting en el documento *Introducción al hash como técnica de seudonimización de datos personales*[5].

El *salting*, al que se suele referir como "uso de sal", es una técnica para dificultar la reidentificación de una contraseña. Para ello, usa un valor al que se le denomina como "sal" y dicho valor se añade al comienzo o al final⁸ de la contraseña original y después de añadirlo, se aplica la función hash. De esta manera, aunque la contraseña cifrada almacenada en la base de datos se viera comprometida, sería imposible saber cual es la contraseña original si no se conoce que sal se ha utilizado.

Podría seguir hablando mucho mas sobre el uso de la sal pero recomiendo leer el apartado 6 de la referencia [5] puesto que me gustaría centrarme más en los aspectos prácticos de encriptado de contraseñas. En concreto, para el encriptado de contraseñas en *MySQL* y en *PHP*.

⁷Usando CLA como "f17ed833132a190c1907db8bbdf19e39", la contraseña obtenida en la base de datos

⁸La posición en la que se añade la sal no es relevante siempre y cuando en todas las contraseñas se añada la sal en la misma posición

4.2 Cifrado de contraseñas usando *MySQL*

El sistema de gestión de bases de datos *MySQL*, que será la herramienta que usaremos como base de datos para la práctica ya que es una herramienta gratuita, tiene funciones nativas muy interesantes para encriptar las contraseñas[6] [7] en la propia base de datos. La función que voy a explicar y que probaremos en la práctica será la función `PASSWORD()` a la que dado un texto plano, realiza el hash de dicho texto. La longitud del resultado y de la función hash usada depende la versión de *MySQL*⁹.

Pese a que la longitud de la contraseña generada depende de la versión, si nuestra versión está actualizada podemos elegir el algoritmo que queremos usar eligiendo el valor para la variable `old_passwords` haciendo únicamente `SET old_passwords=?` y sustituyendo el signo de interrogación por el código que le corresponde. Los códigos son los siguientes:

Código	Encriptación	Bytes
0	Hash nativo MySQL post v4.1	41 bytes
1	Hash "viejo" previo a la versión 4.1	16 bytes
2	Sha 256 con salt*	256 bytes

*El valor `old_passwords=2` usa una sal aleatoria.

Vamos a hacer una prueba para la contraseña "sgssi2021", para ello usaremos la función hash que obtiene un resultado de 41 bytes con lo que ejecutamos `SET old_passwords=0`.

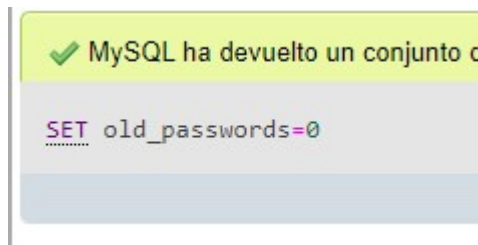


Figure 6: Variable establecida correctamente

Una vez ya hemos decidido que hash queremos que utilice, ejecutamos una sentencia de prueba para obtener el hash de la contraseña "sgssi2021". Para ello usaremos `SELECT PASSWORD('sgssi2021');` obteniendo el siguiente resultado.

⁹La versiones anteriores a la 4.1 usan un hash de 16 bytes y las versiones posteriores una de 41 bytes.



Figure 7: Resultado de la funcion sobre 'sgssi2021'

Como vemos funciona de manera correcta. Volveremos a usar esta herramienta en el apartado 5.2 en la que realizaremos una práctica real usando el encriptado de contraseñas de *MySQL* en un entorno de inicios de sesión.

4.3 Cifrado de contraseñas usando *PHP*

Al igual que *MySQL*, *PHP* también tiene funciones nativas para cifrar contraseñas. Una de las mas conocidas y usadas es `password_hash()` [8]. La función tiene dos parámetros obligatorios y un tercer parámetro opcional. El primero, será la contraseña a cifrar y en el segundo tenemos que decidir el algoritmo de cifrado que usaremos. Para ello, tenemos dos opciones, `PASSWORD_DEFAULT` que usa a día de hoy usa el algoritmo *bcrypt* (el predeterminado a partir de *PHP* 5.5.0), aunque la función está diseñada para actualizarse a medida que salgan algoritmos mas robustos, y la opción de `PASSWORD_BCRYPT` que usa el algoritmo *CRYPT_BLOWFISH* para crear el hash utilizando el identificador "\$2y\$".

Hay que tener en cuenta, y esto es importante, que la función `password_hash()` aplica de manera automática una **sal aleatoria** cada vez que se ejecuta la función. Al ser la sal aleatoria, el resultado de `password_hash()` es distinto cada vez que se ejecuta con lo que para comprobar la contraseña cuando el usuario haga login tendremos que usar la función `password_verify()` [9].

No haría falta usar la función anterior si el usuario especifica que quiere usar una sal en concreto¹⁰ mediante el tercer parámetro opcional ya que la función `password_hash()` devolvería siempre el mismo resultado. Como hemos especificado en el apartado 4.1.2, usando nuestra sal aunque la base de datos se viera comprometida, si el código no se ve comprometido y no averiguan nuestra sal, la contraseñas cifradas con este método estarían seguras. Usando la sal aleatoria la contraseña no se vería comprometida ni aunque se comprometiera el código.

Probaremos estas funciones en el apartado 5.3, en el que encriptaremos la contraseña y comprobaremos el login con `password_hash()`, ya que usaremos una sal predefinida y en el apartado 5.4 en el que cifraremos la contraseña con `password_hash()` con sal aleatoria y comprobaremos con `password_verify()`.

¹⁰Dicha sal deberá de tener una longitud mínima de 22 caracteres

5 Puesta en práctica del encriptado de contraseñas en inicios de sesión

5.1 Preparación de la práctica

La realización de esta práctica será muy sencilla y no serán necesarias muchas herramientas. Lo necesario para realizar la práctica es la aplicación [Xampp](#) y las dos carpetas CSS y PHP creadas por mí, que he dejado accesible en el siguiente [link de Github](#).

Una vez tengamos Xampp instalado, abriremos la carpeta htdocs, que se encontrará en la ruta [C:/xampp/htdocs](#) y colocaremos en esa carpeta las dos carpetas que hemos descargado de Github. El siguiente paso será crear la base de datos en Xampp. Para ello, abrimos la aplicación y clicamos en "start" tanto en la fila de Apache como en la fila de MySQL. A continuación, clicamos en "admin" en la fila de MySQL y veremos como nos abre una ventana en el navegador.

Si la ventana pide que te logees, bastará con darle a entrar. Si eso no funciona, el usuario por defecto es "root" y la contraseña "" (sin contraseña). Una vez iniciado sesión se debería ver el logo de *PHPMysqlAdmin*. Ahora, crearemos la base de datos clicando en "Nueva" en la columna de debajo del logo:

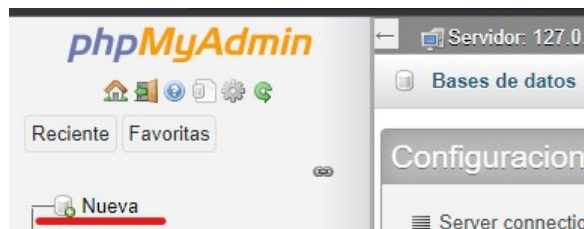


Figure 8: Subrayado en rojo, dónde clicar para crear la BD.

Llamaremos a la base de datos "sgssi", para coincidir con la conexión a la base de datos que se hace desde el código descargado en Github.

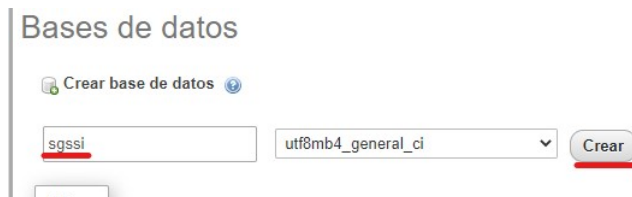


Figure 9: Subrayado en rojo, el nombre a poner para crear la BD.

Una vez creada la base de datos, procedemos a crear la tabla. Para ello, con la base de datos seleccionada, clicaremos donde pone "SQL" y escribiremos lo siguiente:

```
CREATE TABLE 'usuarios' (  
  'name' varchar(20) NOT NULL,  
  'login' varchar(100) NOT NULL,  
  'password' varchar(120) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
ALTER TABLE 'usuarios' ADD PRIMARY KEY ('login');
```

Haremos click en "Continuar" o presionaremos Ctrl+Enter y veremos como se nos ha creado una nueva tabla en nuestra base de datos:

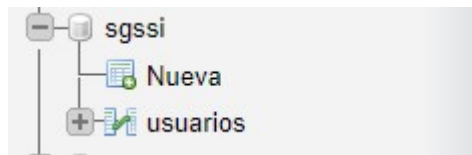


Figure 10: Se ha creado la tabla en la BD.

Una vez creada la tabla, estamos listos para empezar. Primero probaremos el almacenamiento de contraseñas con el cifrado nativo de MySQL.

5.2 Práctica encriptando la contraseña en *MySQL*

Borramos el "/" de la línea 38 y el "/" de la 78. Antes de empezar, miremos el código. Hay dos *ifs* que diferencian el login del registro, ya que ambas peticiones se manejan en el mismo fichero. En lo que nos tenemos que fijar en concreto es en la creación de consultas SQL.

Para el registro, en la línea 71 podemos apreciar que a la hora de hacer el INSERT, el valor de la contraseña esta precedido por la función descrita en el apartado 4.2, la función PASSWORD():

```
#Notese que usamos la funcion PASSWORD a la hora de hacer la consulta SQL  
$sql = "INSERT INTO usuarios VALUES('".$name."','".$login."','".$PASSWORD('".$contrasena."'))";  
mysql_query($link,$sql);  
echo '<p style="color:green; font-size:20px; font-weight: bold;">Registro correcto</p>';
```

Figure 11: Subrayado en rojo, la función PASSWORD.

Lo mismo ocurre con el login, al hacer el SELECT en la línea 52, el valor de la contraseña esta precedido también por la función PASSWORD():

```
#Notese que usamos la funcion PASSWORD a la hora de hacer la consulta SQL
$sql = "SELECT * FROM usuarios WHERE login='".$login.'" AND password=PASSWORD('".$contrasena."')";
$usuario = mysqli_query($link,$sql);
```

Figure 12: Subrayado en rojo, la función PASSWORD.

Una vez visto dónde usamos la función de MySQL, procedemos a realizar la práctica de manera visual. Lo primero será realizar el registro de un usuario. Para ello, clicamos en "Sign Up" y introducimos los datos. En mi caso serán los que se pueden apreciar en la figura 13. Hacemos click en registrar.

The image shows a registration form with a light blue border. At the top, there are two tabs: "SIGN IN" and "SIGN UP", with "SIGN UP" being the active tab. Below the tabs are three input fields: the first contains "Mikel", the second contains "mikel@sgssi.eus", and the third contains "iturria". At the bottom of the form is a blue button labeled "REGISTER".

Figure 13: Valores introducidos para la prueba de registro

Una vez registrado, comprobamos como se ha guardado la contraseña en la base de datos. Para ello vamos a <http://localhost/phpmyadmin/sql.php?db=sgssi&table=usuarios>. Como podemos observar en la siguiente imagen la contraseña se ha guardado de manera correcta.

name	login	password
r Mikel	mikel@sgssi.eus	*B9A0C5F0F906B978CA811FE9D4A190291D4727CF

Figure 14: Datos en la base de datos

Ahora, procedemos a intentar hacer login:

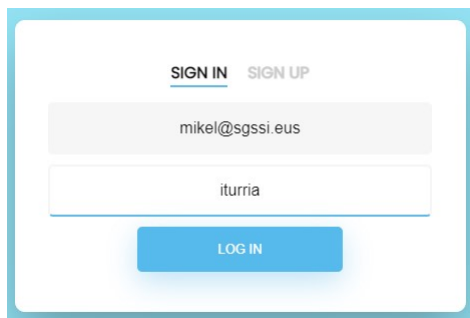
A login form with a light blue border. At the top, there are two links: "SIGN IN" (underlined) and "SIGN UP". Below these are two input fields. The first field contains the email "mikel@sgssi.eus" and the second field contains the password "iturria". At the bottom of the form is a blue button with the text "LOG IN" in white.

Figure 15: Valores introducidos para la prueba de Log in

Obtenemos el siguiente resultado. Sabemos que el resultado es correcto porque aparece el nombre que está guardado en la base de datos, en este caso, Mikel.



Figure 16: Resultado de hacer Log in

Hemos podido observar cómo funciona el manejo de logins usando la criptografía de MySQL. Ahora, para poder continuar con la práctica, vamos a dejar todo como nos lo hemos encontrado. Volvemos a colocar el "/" de la línea 38 y el "/" de la 78 en LogIn.php y además, ejecutamos la siguiente instrucción SQL en la base de datos: `DELETE FROM usuarios WHERE 1=1` para borrar todos los usuarios que hayamos creado.

5.3 Práctica encriptando la contraseña en *PHP* con sal predefinida

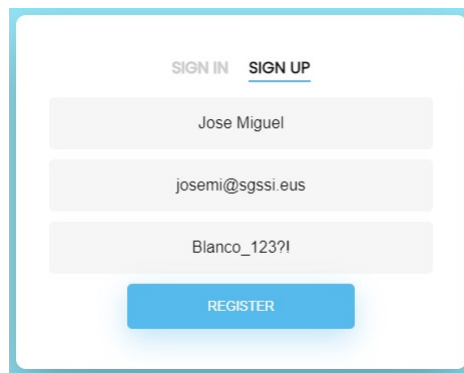
Borramos el "/" de la línea 80 y el "/" de la 132. Antes de empezar, miramos el código. Al igual que el anterior hay dos *ifs* que diferencian el login del registro, pero en esta ocasión a diferencia del apartado 5.2 tenemos mas líneas de código. Para el registro, el código añadido es de la línea 118 a la línea 122 (es el mismo código que se usa para el login):

```
#Elegimos la sal
$options = [
    'salt' => 'mikel iturria para sgssi en 2021'
];
$contra_cifrada = password_hash($contrasena,PASSWORD_DEFAULT,$options);
```

Figure 17: Líneas de código donde se cifra la contraseña.

Como podemos observar en la figura 17, he elegido que la sal sea "mikel iturria para sgssi en 2021" aunque la cadena de caracteres puede ser modificada siempre y cuándo tenga un mínimo de 22 caracteres. Vemos también como se realiza la llamada a la función de *PHP* introduciendo la contraseña enviada vía POST por el usuario, el algoritmo, que en este caso he elegido el algoritmo por defecto y el array "options", donde hemos introducido la sal. Procedemos ahora a probarlo.

Al igual que en el anterior apartado, lo primero será realizar el registro de un usuario. Para ello, clicamos en "Sign Up" y introducimos los datos. Como ejemplo usaremos a Josemi, que al ser el profesor de seguridad, introducirá una contraseña muy robusta. Hacemos click en registrar.



The image shows a registration form with a light blue border. At the top, there are two links: "SIGN IN" and "SIGN UP", with "SIGN UP" being underlined. Below the links are three input fields: the first contains "Jose Miguel", the second contains "josemi@sgssi.eus", and the third contains "Blanco_123?!". At the bottom of the form is a blue button with the text "REGISTER".

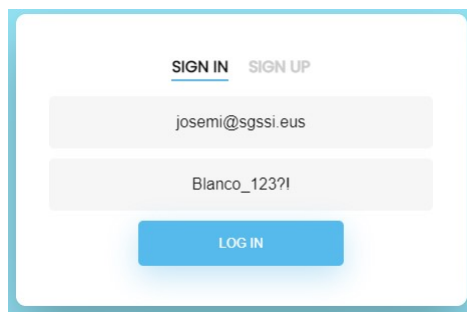
Figure 18: Valores introducidos para la prueba de registro

Una vez registrado, comprobamos como se ha guardado la contraseña en la base de datos. Para ello vamos a <http://localhost/phpmyadmin/sql.php?db=sgssi&table=usuarios>. Como podemos observar en la siguiente imagen la contraseña se ha guardado de manera correcta.

name	login	password
r Jose Miguel	josemi@sgssi.eus	\$2y\$10\$bWlrZWwgaXR1cnJpYSBwYOqPqapTczvH4lprJoUp46/...

Figure 19: Datos en la base de datos

Podemos observar que la contraseña guardada ahora es muy distinta a la que se ha guardado usando MySQL. Ahora, procedemos a intentar hacer login:



A screenshot of a web application's login interface. At the top, there are two links: "SIGN IN" (underlined) and "SIGN UP". Below these are two input fields. The first field contains the email address "josemi@sgssi.eus". The second field contains the password "Blanco_123?!". Below the password field is a blue button labeled "LOG IN". The entire form is enclosed in a light blue border.

Figure 20: Valores introducidos para la prueba de Log in

Obtenemos el siguiente resultado. Sabemos que el resultado es correcto porque aparece el nombre que está guardado en la base de datos, en este caso, Jose Miguel.



Figure 21: Resultado de hacer Log in

Hemos podido observar cómo funciona el manejo de logins usando la criptografía de PHP usando sal fija. Ahora, al igual que el anterior ejercicio vamos a dejar todo como nos lo hemos encontrado. Volvemos a colocar el "/" de la línea 80 y el "*" de la 132 en LogIn.php y además, ejecutamos la siguiente instrucción en la base de datos: `DELETE FROM usuarios WHERE 1=1` para borrar todos los usuarios que hayamos creado.

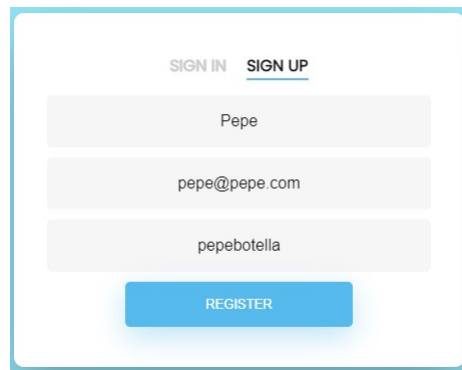
5.4 Práctica encriptando la contraseña en *PHP* con sal aleatoria

Borramos el "/" de la línea 134 y el "*" de la 181. Antes de empezar, miramos el código. Al igual que en los anteriores apartados hay dos *ifs* que diferencian el login del registro. El registro, se realiza de manera casi idéntica al apartado 5.3 solo que en esta ocasión la función `password_hash()` solo tiene dos parámetros, ya que no estamos añadiendo la sal de manera predefinida.

```
#Usamos sal aleatoria
$contra_cifrada = password_hash($contrasena,PASSWORD_DEFAULT);
```

Figure 22: Líneas de código donde se cifra la contraseña.

Al igual que en los anteriores apartados, lo primero será realizar el registro de un usuario. Para ello, clicamos en "Sign Up" y introducimos los datos. Como ejemplo usaremos a "Pepe", con contraseña "pepebotella".



The image shows a registration form with a light blue border. At the top, there are two links: "SIGN IN" and "SIGN UP", with "SIGN UP" being underlined. Below the links are three input fields. The first field contains the text "Pepe". The second field contains the text "pepe@pepe.com". The third field contains the text "pepebotella". Below these fields is a blue button with the text "REGISTER" in white capital letters.

Figure 23: Valores introducidos para la prueba de registro

Una vez registrado, comprobamos como se ha guardado la contraseña en la base de datos. Para ello vamos a <http://localhost/phpmyadmin/sql.php?db=sgssi&table=usuarios>. Como podemos observar en la siguiente imagen la contraseña se ha guardado de manera correcta.¹¹

▼	name	login	password
Borrar	Pepe	pepe@pepe.com	\$2y\$10\$NyQ.Xs6J605fGQAzy8GefuxNr46ZRUwNBmu/nHfq58T...

Figure 24: Datos en la base de datos

Para realizar el login hay una diferencia respecto al anterior apartado ya que ahora tenemos que usar `password_verify()`. Para ello, al hacer la llamada a la base de datos solo mandaremos como parámetro el correo electrónico y recogeremos la contraseña del resultado.

¹¹La contraseña de la foto no coincidirá con la que te ha salido a ti porque se usa una sal aleatoria y en consecuencia cada vez que se usa la función la contraseña será distinta.

```
#Notamos que ahora no pedimos la contraseña a la hora de hacer la consulta SQL
$sql = "SELECT * FROM usuarios WHERE login='" . $login . "'";
```

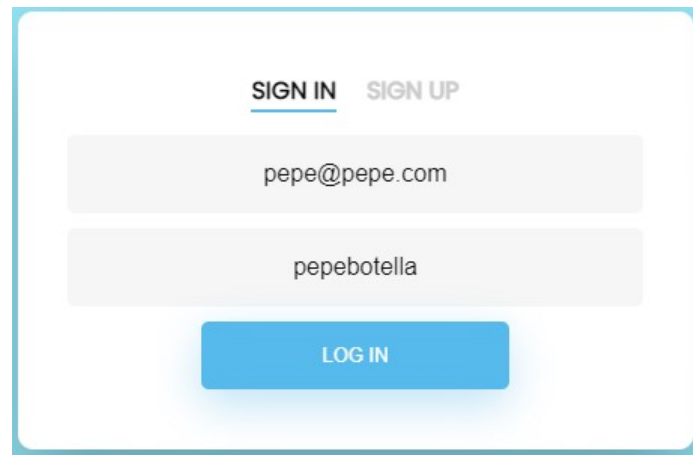
Figure 25: Llamada a la base de datos

Una vez tenemos la contraseña, la comprobamos usando `password_verify()` a la que le pasaremos la contraseña introducida por el usuario como primer parámetro y la contraseña cifrada obtenida de la base de datos como segundo parámetro:

```
#Guardamos la contraseña cifrada que obtenemos de la base de datos
$contrasenaDB = $row["password"];
#Será la función password_verify la que compruebe si la contraseña es correcta.
if(password_verify($contrasena,$contrasenaDB)){
    echo '<p style="color:green; font-size:20px; font-weight: bold;">Login correcto, '.$row["name"].'</p>';
}else{
    echo '<p style="color:red; font-size:20px; font-weight: bold;">LOGIN INCORRECTO</p>';
}
```

Figure 26: Uso de la función `password_verify()`

Una vez visto el código, procedemos a intentar hacer login:



The image shows a login interface with a light blue border. At the top, there are two links: 'SIGN IN' (underlined) and 'SIGN UP'. Below these are two input fields. The first field contains the email 'pepe@pepe.com' and the second field contains the password 'pepebotella'. Below the password field is a blue button with the text 'LOG IN' in white capital letters.

Figure 27: Valores introducidos para la prueba de Log in

Obtenemos el siguiente resultado. Sabemos que el resultado es correcto porque aparece el nombre que está guardado en la base de datos, en este caso, Pepe.



Figure 28: Resultado de hacer Log in

Hemos podido observar cómo funciona el manejo de logins usando la criptografía de PHP usando sal aleatoria. Ahora, al igual que el anterior ejercicio vamos a dejar todo como nos lo hemos encontrado. Volvemos a colocar el "/" de la línea 134 y el "*" de la 181 en LogIn.php y además, ejecutamos la siguiente instrucción en la base de datos: `DELETE FROM usuarios WHERE 1=1` para borrar todos los usuarios que hayamos creado.

Para ver que la sal es realmente aleatoria, podemos realizar el registro del apartado 5.4 de nuevo y comprobaremos que la contraseña guardada en la base de datos es distinta que a la que habíamos realizado anteriormente, pese a haber usado la misma función para la misma contraseña.

	name	login	password
ar	Pepe	pepe@pepe.com	\$2y\$10\$VO6zYQAWY6b2vu5KJtEldujk8VnrD1nQHZEgKx9.4TP...

Figure 29: La contraseña es distinta a la que aparece en la figura 24

6 Conclusiones

En conclusión, mi recomendación es usar el método realizado en la práctica 5.4 usando una sal aleatoria ya que cuenta con todas las ventajas de usar una sal predefinida pero con una ventaja mas y es que, aunque sea difícil, una contraseña cifrada con una sal predefinida puede ser reidentificada si la base de datos y el código se ven comprometidos. Algo que no sucede usando una sal aleatoria.

7 Referencias

References

- [1] *Password Management Strategies for Online Accounts*.
Shirley Gaw, Edward W. Felten
Department of Computer Science.
Princeton University.
<https://dl.acm.org/doi/pdf/10.1145/1143120.1143127>

- [2] *Single Password, Multiple Accounts*
Mohamed G. Gouda, Alex X. Liu, Lok M. Leung, Mohamed A. Alam
Department of Computer Science.
The University of Texas at Austin,.
[http://www.cse.msu.edu/~alexliu/publications/Password/
password.pdf](http://www.cse.msu.edu/~alexliu/publications/Password/password.pdf)
- [3] *Cómo y cuándo cifrar documentos para cumplir la legislación de protección de datos*
[https://www.proteccion-de-datos-madrid.com/index.php/
94-como-y-cuando-cifrar-documentos-para-cumplir-la-lopd/](https://www.proteccion-de-datos-madrid.com/index.php/94-como-y-cuando-cifrar-documentos-para-cumplir-la-lopd/)
- [4] *Diferencias entre el rgpd y el lpd.*
Apartado: *Sanciones*
<https://clickdatos.es/cuales-son-las-diferencias-entre-el-rgpd-y-la-lopd/>
- [5] *AEPD: Introducción al hash como técnica de seudonimización de datos personales*
Capítulo VI: *Anexión de una cabecera al mensaje o sal con reutilización.*
[https://www.aepd.es/sites/default/files/2020-05/
estudio-hash-anonimidad.pdf](https://www.aepd.es/sites/default/files/2020-05/estudio-hash-anonimidad.pdf)
- [6] *Password hashing in MySQL*
<https://dev.mysql.com/doc/refman/5.6/en/password-hashing.html>
- [7] *Encryption functions: The PASSWORD() function*
[https://dev.mysql.com/doc/refman/5.6/en/encryption-functions.
html#function_password](https://dev.mysql.com/doc/refman/5.6/en/encryption-functions.html#function_password)
- [8] *Function: password_hash()*
<https://www.php.net/manual/es/function.password-hash.php>
- [9] *Function: password_verify()*
<https://www.php.net/manual/es/function.password-verify.php>