

TAREA OPCIONAL

Antes de ponernos a hablar sobre la tarea opcional, hemos realizado unos cambios significativos respecto a la entrega anterior:

Evitar inyecciones SQL

Hemos modificado *Login.php* de manera que no se pueda realizar infecciones SQL. Para ello, hemos usado la función de PHP *mysqli_real_escape_string*:

```
$emailseguro=mysqli_real_escape_string($link, strip_tags($_POST["email"]));
$passsegura=mysqli_real_escape_string($link, strip_tags($_POST["pass"]));
$passCryp = crypt($passsegura,'$5$rounds=5000$ArenItur$');

$sql = "SELECT * FROM usuario WHERE usuario.email='" . $emailseguro . "' AND usuario.pass='" . $passCryp . "'";
```

Evitar la inserción de scripts dentro de preguntas o en el registro

Usando la función *strip_tags* hemos evitado que se puedan introducir scripts en la base de datos:

En *SignUp.php*:

```
//Comprobamos siptpygialides.
$nuevapass = crypt(strip_tags($_POST["pass"]), '$5$rounds=5000$ArenItur$');

$sql = "INSERT INTO usuario(tipo,email,nombre,pass,imagen) values('" . strip_tags($_POST["tipo"]) . "', '" . strip_tags($_POST["em"] . "';
if (!mysqli_query($link, $sql))
```

En *AddQuestionWithImageAjax.php*:

```
//Insercion en XML
$xml = simplexml_load_file('../xml/Questions.xml');
$pregunta = $xml->addChild('assessmentItem');
$pregunta->addAttribute('subject', strip_tags($_POST['tema']));
$pregunta->addAttribute('author', strip_tags($_POST['email']));
$itembody = $pregunta->addChild('itemBody');
$itembody->addChild('p', $_POST['enunc']);
$correctResponse = $pregunta->addChild('correctResponse');
$response = $correctResponse->addChild('response', strip_tags($_POST['resco']));
$incorrectResponses = $pregunta->addChild('incorrectResponses');
$incorrectResponses->addChild('response', strip_tags($_POST['resin1']));
$incorrectResponses->addChild('response', strip_tags($_POST['resin2']));
$incorrectResponses->addChild('response', strip_tags($_POST['resin3']));
//Insercion indentada:
$xmlContent = formatXml($xml);
$nuevoxml = new SimpleXMLElement($xmlContent);
$nuevoxml->asXML('../xml/Questions.xml');
echo '<p style="color:green; font-size:20px; font-weight: bold;">Pregunta insertada correctamente en el archivo xml</p>';

//insercion en DB
$link = new mysqli($server, $user, $pass, $basededatos);
$sql = "INSERT INTO preguntasconimagen(email,enunc,resco,resin1,resin2,resin3,difi,tema,img) values('" . strip_tags($_POST["ema"] . "';
if (!mysqli_query($link, $sql))
```

Uso de alertas personalizadas

Por cuestiones estéticas hemos decidido cambiar los alerts normales que se lanzan cuando usas "alert()" en JavaScript por alertas de la librería **SweetAlert2**.

- Ejemplo de código si un usuario **no debería de estar ahí**:

En *HandlingAccounts.php*:

```
<?php if(!isset($_SESSION['tipo']) || ($_SESSION['tipo']!="admin")){
    echo "<script>
        Swal.fire({
            icon: 'error',
            title: 'Vaya, parece que no deberías estar aquí...',
            allowOutsideClick: false,
            showDenyButton: false,
            showCancelButton: false,
            confirmButtonText: 'De acuerdo',
            denyButtonText: 'No',
        }).then((result) => {
            if (result.isConfirmed) {
                window.location.href = 'Layout.php';
            }
        });
    </script>";
```

Así se ve:



- Ejemplo de **mensaje de confirmación**:
En *alterarUsuarios.js*:

```
Swal.fire({  
  title: 'Estas seguro de querer cambiar el estado de este usuario?',  
  showDenyButton: true,  
  showCancelButton: false,  
  confirmButtonText: `Si`,  
  denyButtonText: `No`,  
}).then((result) => {  
  if (result.isConfirmed) {  
    window.location.href = "ChangeUserState.php?email="+email;  
  } else if (result.isDenied) {  
  }  
})
```

Así se vería:

**¿Estás seguro de querer
cambiar el estado de este
usuario?**

Si

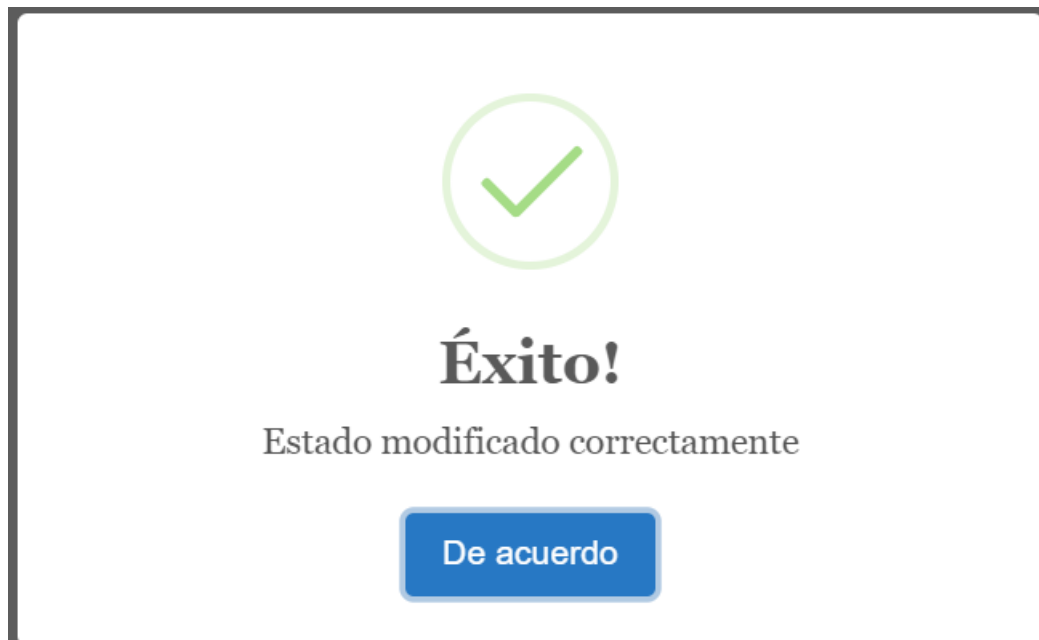
No

- **Mensaje de éxito** (cambio de estado/borrado):

Código en *ChangeUserState.php*:

```
echo "<script> Swal.fire({
  icon: 'success',
  title: 'Éxito!',
  text: 'Estado modificado correctamente',
  confirmButtonText: `De acuerdo`,
  allowOutsideClick: false,
}).then((result) => {
  if (result.isConfirmed) {
    window.location.href = 'HandlingAccounts.php';
  })</script>";
```

Así se vería:



Vamos ahora con la tarea opcional.

Realización de la tarea opcional

Para esta tarea opcional, hemos elegido encriptar las contraseñas usando **SHA 256**. Para eso el **salt** que le hemos añadido a la función *crypt()* comienza por “\$5\$rounds=5000\$” a la que le hemos añadido nuestra “clave” personalizada que hemos elegido que sea “ArenItur”, haciendo referencia las 4 primeras letras de nuestro primer apellido:



```
//Contraseña criptografiada:
$nuevapass = crypt(strip_tags($_POST["pass"]), '$5$rounds=5000$ArenItur$');
```

Hemos registrado nuevos usuarios y borrado los viejos, vamos a ver su contraseña a la BD:


tipo	email	nombre	pass	imagen	estado
admin	admin@ehu.es	Admin Admin	\$5\$rounds=5000\$ArenItur\$xc1RL5zlwVjstscHtuA3.t1Nw7...	../images/Soy_Admin.jpg	Activo
estu	garenales001@ikasle.ehu.eus	Guillermo Arenales	\$5\$rounds=5000\$ArenItur\$OXXto/OddBH3Sc.chFvxBf4k7C...	../images/	Bloqueado
estu	miturria003@ikasle.ehu.eus	Mikel Iturria	\$5\$rounds=5000\$ArenItur\$8IxATRnODb4wAwfSpWIPVOuP.c...	../images/mikelperfil.jpg	Activo
profe	vadillo@ehu.es	Jose Angel Vadillo	\$5\$rounds=5000\$ArenItur\$OXXto/OddBH3Sc.chFvxBf4k7C...	../images/	Activo

Efectivamente, se puede apreciar que la contraseña esta encriptada.

NOTA: A la hora de mostrar los usuarios en *HandlingUsers.php* pensamos en mostrar las contraseñas desenscriptadas pero resulta que no se puede:

- 3  `crypt` does not encrypt passwords (so there is no way to decrypt them). Instead it [hashes](#) a given password, producing a string that is impossible to reverse to the original password (because the hash function loses information in the process). The most practical way to attack `crypt` and recover passwords from their hashes is probably some sort of [dictionary attack](#). 

Con lo que en el panel del administrador aparecen las contraseñas encriptadas y solo el propio usuario podrá saber la contraseña:

Todos los usuarios							
Correo	Contraseña	Tipo	Imagen	Estado	Bloqueo	Borrar	
garenales001@ikasle.ehu.eus	\$5\$rounds=5000\$ArenItur\$0XKto/OddBH3Se.chFvxBf4k7C4gWY0iUrBHMf5GZUC	Estudiante	Sin imagen	Bloqueado	Cambiar Estado	Borrar usuario	
miturria003@ikasle.ehu.eus	\$5\$rounds=5000\$ArenItur\$8IxATRNOdb4wAwfSpWIPVouP.ceyiMEQndT6hxsUBg1	Estudiante		Activo	Cambiar Estado	Borrar usuario	
vadillo@ehu.es	\$5\$rounds=5000\$ArenItur\$0XKto/OddBH3Se.chFvxBf4k7C4gWY0iUrBHMf5GZUC	Profesor	Sin imagen	Activo	Cambiar Estado	Borrar usuario	

Peticiones del formulario de entrega:

Función utilizada

Se ha suado la función **crypt** con un salt **SHA 256** tal y como se ha explicado anteriormente.

Cambios en SignUp.php:

Creamos la nueva contraseña usando la función:

```
//Contraseña criptografiada:
$nuevapass = crypt(strip_tags($_POST["pass"]), '$5$rounds=5000$ArenItur$');
```

Cambios en Login.php:

Ahora en la BD estará guardada la contraseña cifrada, con lo que tendremos que cifrar lo que introduzca el usuario en el campo contraseña con el mismo *salt* que hemos usado en *SignUp.php* y hacer la llamada SQL con la contraseña cifrada:

```
$emailseguro=mysqli_real_escape_string($link, strip_tags($_POST["email"]));
$passseguro=mysqli_real_escape_string($link, strip_tags($_POST["pass"]));
$passCryp = crypt($passseguro, '$5$rounds=5000$ArenItur$');

$sql = "SELECT * FROM usuario WHERE usuario.email='".$emailseguro."' AND usuario.pass='".$passCryp.'";
```

Ejemplo de contraseña original y contraseña cifrada:

Como ejemplo de contraseña cifrada sirve el usuario administrador cuya contraseña es **admin000**:

Contraseña inicial: admin000

Contraseña después de cifrar:

\$5\$rounds=5000\$ArenItur\$xc1RL5zIwVjstscHtuA3.t1Nw7jPfqXaktTnAFCmo.6