Clase 11

Análisis de Sistemas

Materia:
Programación Web II

Docente contenidista: ROLDÁN, Hernán

Revisión: Coordinación

Contenido

Módulo MySQLi (Parte 2) 3

CRUD a una base de datos MySQL 3

Mejorando aún más la seguridad de la aplicación 15

- htmlspecialchars 15
- trim 16
- stripslashes 16

Bibliografía 22

Para ampliar la información 22

Clase 11

¡Te damos la bienvenida a la materia **Programación Web II**!

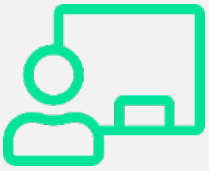
En esta clase vamos a ver los siguientes temas:

- CRUD a una base de datos MySQL.
- Seguridad de la conexión.
- Seguridad de la aplicación.

Módulo MySQLi (Parte 2)

CRUD a una base de datos MySQL

En la clase anterior, vimos el módulo MySQLi, una poderosa extensión de PHP que nos permite interactuar con bases de datos MySQL de manera eficiente y segura.
Aprendimos a establecer una conexión a la base de datos utilizando tanto el enfoque procedural como el orientado a objetos, destacando la importancia de manejar los errores durante la conexión para garantizar una conexión exitosa.
Exploramos los métodos más utilizados de MySQLi, como **mysqli_connect(), mysqli_query(), mysqli_fetch_assoc(),** entre otros, que nos permitieron ejecutar consultas SQL y obtener los resultados de manera estructurada.
También aprendimos a manejar errores y excepciones correctamente durante la ejecución de consultas para asegurar una mayor robustez en nuestras aplicaciones. Mediante ejemplos prácticos, logramos comprender cómo trabajar con MySQLi de forma efectiva, lo que nos permite desarrollar aplicaciones web que gestionen datos de manera segura y confiable en futuros proyectos.



Con estos conocimientos, estamos preparados para enfrentar nuevos desafíos en el manejo de bases de datos MySQL y seguir mejorando nuestras habilidades como desarrolladores PHP.

En esta oportunidad, aplicaremos todos los conocimientos adquiridos para realizar un CRUD (Crear, Leer, Actualizar y Eliminar) a la base de datos MySQL. Utilizaremos el módulo MySQLi en estilo procedural para establecer la conexión a la base de datos y llevar a cabo diferentes operaciones sobre la tabla "usuarios" que creamos previamente.
A través de este ejercicio, pondremos en práctica el uso de los métodos **mysqli_query()**, **mysqli_fetch_assoc()**, **mysqli_affected_rows()** y **mysqli_error()**, entre otros, para crear nuevos registros, leer los datos existentes, actualizar la información y eliminar registros de la base de datos de manera segura y eficiente.
También aprenderemos a manejar adecuadamente los errores y excepciones que puedan surgir durante estas operaciones.

Veamos este código detalladamente:

```php
<?php

  // Configuración de la conexión a la base de datos
  $servidor = "localhost";
  $usuario = "root";
  $clave = "";
  $basededatos = "crud_mysqli";

  // Establecer la conexión a la base de datos
  $mysqli = mysqli_connect($servidor, $usuario, $clave, $basededatos);

  // Comprobar si se produjo algún error durante la conexión
  if(!$mysqli){
    die("Error de conexión: " . mysqli_connect_error());
  }

  // Función para mostrar un mensaje de éxito o error
  function mostrarMensaje($mensaje, $tipo = 'success'){
    echo "<div style='padding: 10px; background-color: ";
    if($tipo === 'success'){
      echo 'green';
    }
    else{
      echo 'red';
    }
    echo "; color: white; text-align: center;'>$mensaje</div>";
  }

  // Leer todos los registros de la tabla "usuarios"
  function leerUsuarios($mysqli){
    $query = "SELECT * FROM usuarios";
    $result = mysqli_query($mysqli, $query);
    return $result;
  }

  // Insertar un nuevo registro en la tabla "usuarios"
  function insertarUsuario($mysqli, $nombre, $email, $edad){
    $query = "INSERT INTO usuarios (nombre, email, edad) VALUES
('$nombre', '$email', $edad)";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro insertado correctamente.");
    }
    else{
```

```
      mostrarMensaje("Error al insertar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Actualizar un registro existente en la tabla "usuarios"
  function actualizarUsuario($mysqli, $id, $nombre, $email, $edad) {
    $query = "UPDATE usuarios SET nombre = '$nombre', email = '$email',
edad = $edad WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro actualizado correctamente.");
    }
    else{
      mostrarMensaje("Error al actualizar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Eliminar un registro de la tabla "usuarios"
  function eliminarUsuario($mysqli, $id){
    $query = "DELETE FROM usuarios WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
        mostrarMensaje("Registro eliminado correctamente.");
    }
    else{
      mostrarMensaje("Error al eliminar el registro: " .
mysqli_error($mysqli), 'error');
    }
}

  // Procesar formularios
  if($_SERVER['REQUEST_METHOD'] === 'POST'){
    if(isset($_POST['insertar'])){
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      insertarUsuario($mysqli, $nombre, $email, $edad);
    }
    elseif(isset($_POST['actualizar'])){
      $id = $_POST['id'];
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      actualizarUsuario($mysqli, $id, $nombre, $email, $edad);
    }
    elseif (isset($_POST['eliminar'])){
```

```
      $id = $_POST['id'];
      eliminarUsuario($mysqli, $id);
    }
  }

  // Obtener todos los usuarios de la tabla "usuarios"
  $result = leerUsuarios($mysqli);

?>

<!DOCTYPE html>
<html>
<head>
    <title>CRUD MySQLi</title>
</head>
<body>
    <h1>CRUD MySQLi - Usuarios</h1>
    <h2>Insertar nuevo usuario:</h2>
    <form method="post">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" required>
        <label for="email">Email:</label>
        <input type="email" name="email" required>
        <label for="edad">Edad:</label>
        <input type="number" name="edad" required>
        <input type="submit" name="insertar" value="Insertar">
    </form>

    <h2>Actualizar usuario:</h2>
    <?php while($usuario = mysqli_fetch_assoc($result)): ?>
        <form method="post">
            <input type="hidden" name="id" value="<?php echo
$usuario['id']; ?>">
            <label for="nombre">Nombre:</label>
            <input type="text" name="nombre" value="<?php echo
$usuario['nombre']; ?>" required>
            <label for="email">Email:</label>
            <input type="email" name="email" value="<?php echo
$usuario['email']; ?>" required>
            <label for="edad">Edad:</label>
            <input type="number" name="edad" value="<?php echo
$usuario['edad']; ?>" required>
            <input type="submit" name="actualizar" value="Actualizar">
            <input type="submit" name="eliminar" value="Eliminar">
        </form>
    <?php endwhile; ?>

    <?php
```

```
    // Cerrar la conexión a la base de datos
    mysqli_close($mysqli);

  ?>

</body>
</html>
```

Ahora explicaremos paso a paso el código:

1. Se establece la conexión a la base de datos utilizando las credenciales proporcionadas. Si la conexión falla, se muestra un mensaje de error y se termina la ejecución del script.

2. Se define una función llamada **mostrarMensaje()** que muestra un mensaje en la parte superior de la página. Esta función recibe dos parámetros: el mensaje a mostrar y el tipo de mensaje (por defecto, 'success' para mensajes de éxito). Dependiendo del tipo, se establece el color de fondo del mensaje en verde para éxito o rojo para errores.

3. Se define una función llamada **leerUsuarios()** que realiza una consulta a la base de datos para obtener todos los registros de la tabla "usuarios". La función retorna el resultado de la consulta.

4. Se define una serie de funciones para realizar las operaciones del CRUD:

   - **insertarUsuario($mysqli, $nombre, $email, $edad):** Inserta un nuevo registro en la tabla "usuarios" con los datos proporcionados.

   - **actualizarUsuario($mysqli, $id, $nombre, $email, $edad):** Actualiza un registro existente en la tabla "usuarios" con los datos proporcionados.

   - **eliminarUsuario($mysqli, $id):** Elimina un registro de la tabla "usuarios" con el ID especificado.

5. Se procesan los formularios enviados mediante el método POST. Si se detecta un envío de formulario para insertar, actualizar o eliminar un usuario, se llaman a las funciones correspondientes para realizar las operaciones en la base de datos. Luego, se muestra un mensaje de éxito o error dependiendo del resultado de la operación.

6. Se define una estructura HTML que muestra dos secciones:

   - En la primera sección, hay un formulario para insertar un nuevo usuario en la tabla "usuarios". El formulario contiene campos para ingresar el nombre, email y edad del nuevo usuario, y un botón para enviar los datos para su inserción.

   - En la segunda sección, se utiliza un bucle while para recorrer los resultados obtenidos de la consulta a la base de datos mediante **leerUsuarios().** Para cada usuario, se muestra un formulario con campos para editar los datos del usuario (nombre, email y edad), y botones para enviar los datos para su actualización o eliminación.

7. El script finaliza cerrando la conexión a la base de datos mediante **mysqli_close($mysqli)**.

En resumen, este código implementa un CRUD completo utilizando el módulo MySQLi en estilo procedural para la tabla "usuarios". Permite insertar nuevos registros, actualizar datos existentes y eliminar registros de la base de datos.
Los mensajes de éxito o error se muestran en la parte superior de la página después de cada operación, lo que brinda una retroalimentación clara al usuario.
La estructura HTML muestra un formulario para insertar nuevos usuarios y una lista de usuarios existentes con opciones para editar o eliminar sus datos.

*Recordá que el módulo **mysqli** ofrece varios métodos para realizar diferentes operaciones, como insertar, actualizar, leer y eliminar datos.*
Podés encontrar más información sobre esta librería consultando la documentación oficial de PHP:
https://www.php.net/manual/en/book.mysqli.php

Agregando seguridad a la conexión

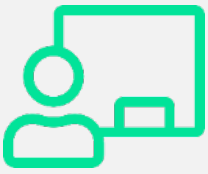
En la clase anterior, cuando tratamos cuáles eran algunos de los métodos mysqli más usados mencionamos el método **mysqli_real_escape_string,** sobre su finalidad y uso.
Con la idea de ampliar este tema tan importante y que es la seguridad, lo retomaremos a continuación con el fin de implementar este al código anterior.

A modo repaso, recordemos que el método **mysqli_real_escape_string** es una función de PHP que se utiliza para escapar caracteres especiales dentro de una cadena de texto antes de enviarla a una consulta SQL.
Su propósito principal es prevenir la inyección de SQL, que es una vulnerabilidad de seguridad común en aplicaciones web.

La inyección de SQL ocurre cuando un atacante inserta código SQL malicioso en una entrada de datos de la aplicación con el objetivo de manipular la consulta SQL y obtener acceso no autorizado a la base de datos o realizar otras operaciones dañinas.

Cuando utilizamos el método **mysqli_real_escape_string**, esta función reemplaza caracteres especiales que podrían afectar negativamente a una consulta SQL.

Por ejemplo, si la cadena de texto contiene comillas simples o dobles, estos caracteres son precedidos por una barra invertida \, lo que evita que la consulta SQL los interprete como delimitadores de cadenas y, en su lugar, los trate como parte de los datos.



Veamos a continuación la explicación y ejemplo práctico de un ataque mediante el uso de inyecciones SQL

Supongamos que tenemos un formulario donde los usuarios pueden ingresar su nombre de usuario y contraseña para iniciar sesión:

```php
<?php

  // Datos ingresados por el usuario
  $usuario = $_POST['usuario'];
  $clave = $_POST['clave'];

  // Consulta SQL
  $query = "SELECT * FROM usuarios WHERE usuario='$usuario' AND
contrasena='$clave'";

?>
```

En este caso, si un atacante ingresa en el campo de usuario el siguiente valor: ' OR '1'='1, la consulta SQL resultante se verá así:

```
SELECT * FROM usuarios WHERE usuario='' OR '1'='1' AND clave=''
```

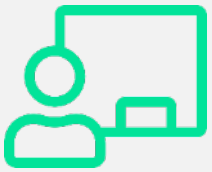
Esta consulta maliciosa siempre devolverá todos los registros de la tabla de usuarios porque la condición '1'='1' siempre es verdadera, o lo que en lógica proposicional se conoce como tautología.
De esta manera, el atacante podría obtener acceso no autorizado a la aplicación.

Para prevenir este tipo de ataque, debemos usar **mysqli_real_escape_string** para escapar los caracteres especiales en las variables antes de construir la consulta SQL:

```php
<?php

  // Escapar caracteres especiales antes de utilizarlos en la consulta
SQL
  $usuario = mysqli_real_escape_string($mysqli, $_POST['usuario']);
  $clave = mysqli_real_escape_string($mysqli, $_POST['clave']);

  // Consulta SQL
  $query = "SELECT * FROM usuarios WHERE usuario='$usuario' AND
clave='$clave'";

?>
```

Con **mysqli_real_escape_string**, el valor ' OR '1'='1 sería escapado y tratado como una cadena, lo que evitaría la inyección de SQL y haría que la consulta funcione de manera segura:

```
SELECT * FROM usuarios WHERE usuario='\' OR \'1\'=\'1' AND
contrasena=''
```



Veamos Código del CRUD refactorizado (actualizado) con la implementación del método **mysqli_real_scape_string**

```php
<?php

  // Configuración de la conexión a la base de datos
  $servidor = "localhost";
  $usuario = "root";
  $clave = "";
  $basededatos = "crud_mysqli";

  // Establecer la conexión a la base de datos
  $mysqli = mysqli_connect($servidor, $usuario, $clave, $basededatos);

  // Comprobar si se produjo algún error durante la conexión
  if(!$mysqli){
    die("Error de conexión: " . mysqli_connect_error());
  }

  // Función para mostrar un mensaje de éxito o error
  function mostrarMensaje($mensaje, $tipo = 'success'){
    echo "<div style='padding: 10px; background-color: ";
    if($tipo === 'success'){
      echo 'green';
    }
    else{
      echo 'red';
    }
    echo "; color: white; text-align: center;'>$mensaje</div>";
  }

  // Leer todos los registros de la tabla "usuarios"
  function leerUsuarios($mysqli){
    $query = "SELECT * FROM usuarios";
    $result = mysqli_query($mysqli, $query);
    return $result;
  }

  // Insertar un nuevo registro en la tabla "usuarios"
  function insertarUsuario($mysqli, $nombre, $email, $edad){
    $nombre = mysqli_real_escape_string($mysqli, $nombre);
    $email = mysqli_real_escape_string($mysqli, $email);
```

```
    $query = "INSERT INTO usuarios(nombre, email, edad)
VALUES('$nombre', '$email', $edad);";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro insertado correctamente.");
    }
    else{
      mostrarMensaje("Error al insertar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Actualizar un registro existente en la tabla "usuarios"
  function actualizarUsuario($mysqli, $id, $nombre, $email, $edad) {
    $nombre = mysqli_real_escape_string($mysqli, $nombre);
    $email = mysqli_real_escape_string($mysqli, $email);
    $query = "UPDATE usuarios SET nombre = '$nombre', email =
'$email', edad = $edad WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro actualizado correctamente.");
    }
    else{
      mostrarMensaje("Error al actualizar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Eliminar un registro de la tabla "usuarios"
  function eliminarUsuario($mysqli, $id){
    $query = "DELETE FROM usuarios WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
        mostrarMensaje("Registro eliminado correctamente.");
    }
    else{
      mostrarMensaje("Error al eliminar el registro: " .
mysqli_error($mysqli), 'error');
    }
}

  // Procesar formularios
  if($_SERVER['REQUEST_METHOD'] === 'POST'){
    if(isset($_POST['insertar'])){
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      insertarUsuario($mysqli, $nombre, $email, $edad);
```

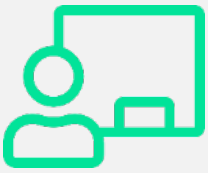
```
    }
    elseif(isset($_POST['actualizar'])){
      $id = $_POST['id'];
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      actualizarUsuario($mysqli, $id, $nombre, $email, $edad);
    }
    elseif(isset($_POST['eliminar'])){
      $id = $_POST['id'];
      eliminarUsuario($mysqli, $id);
    }
  }

  // Obtener todos los usuarios de la tabla "usuarios"
  $result = leerUsuarios($mysqli);

?>

<!DOCTYPE html>
<html>
<head>
    <title>CRUD MySQLi</title>
</head>
<body>
    <h1>CRUD MySQLi - Usuarios</h1>
    <h2>Insertar nuevo usuario:</h2>
    <form method="post">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" required>
        <label for="email">Email:</label>
        <input type="email" name="email" required>
        <label for="edad">Edad:</label>
        <input type="number" name="edad" required>
        <input type="submit" name="insertar" value="Insertar">
    </form>

    <h2>Actualizar usuario:</h2>
    <?php while($usuario = mysqli_fetch_assoc($result)): ?>
        <form method="post">
            <input type="hidden" name="id" value="<?php echo
$usuario['id']; ?>">
            <label for="nombre">Nombre:</label>
            <input type="text" name="nombre" value="<?php echo
$usuario['nombre']; ?>" required>
            <label for="email">Email:</label>
            <input type="email" name="email" value="<?php echo
$usuario['email']; ?>" required>
```

```
            <label for="edad">Edad:</label>
            <input type="number" name="edad" value="<?php echo
$usuario['edad']; ?>" required>
            <input type="submit" name="actualizar" value="Actualizar">
            <input type="submit" name="eliminar" value="Eliminar">
        </form>
    <?php endwhile; ?>

    <?php

      // Cerrar la conexión a la base de datos
      mysqli_close($mysqli);

    ?>

</body>
</html>
```

*En resumen, el método **mysqli_real_escape_string** es una herramienta esencial para proteger nuestras aplicaciones web contra ataques de inyección SQL al escapar correctamente los caracteres especiales en las entradas del usuario antes de utilizarlos en consultas SQL. Al hacerlo, podemos garantizar que nuestras aplicaciones sean más seguras y confiables.*

Mejorando aún más la seguridad de la aplicación

Por último, y para mejorar aún más la seguridad de nuestra aplicación, agregaremos al código anterior los métodos **htmlspecialchars**, **trim**, y **stripslashes** y que ya vimos en clases anteriores.



Recordá que estos métodos los usamos para la sanitización de datos, ya que proporcionan una capa adicional de seguridad y consistencia en las entradas de datos del usuario antes de utilizarlos en una aplicación web.
Cada uno de estos métodos tiene un propósito específico para abordar diferentes aspectos de la seguridad y la integridad de los datos.

Repasemos rápidamente lo que resuelve cada uno:

- htmlspecialchars

  Este método se utiliza para prevenir ataques XSS (Cross-Site Scripting). XSS es una vulnerabilidad común en aplicaciones web que permite a los atacantes insertar código malicioso, generalmente en forma de scripts, en páginas web vistas por otros usuarios.

  Al convertir caracteres especiales a entidades HTML, como <, >, &, ', y ", **htmlspecialchars** evita que los scripts maliciosos se ejecuten en el navegador del usuario.

  En lugar de interpretar los caracteres como código HTML o JavaScript, los navegadores los mostrarán como texto normal.

  Esto ayuda a proteger la aplicación y a los usuarios de posibles ataques XSS.

- ## trim

  Este método se utiliza para eliminar espacios en blanco al inicio y al final de una cadena de texto.

  A menudo, los usuarios pueden involuntariamente agregar espacios en blanco adicionales al completar formularios, lo que podría afectar negativamente la funcionalidad del sistema o causar problemas de validación.

  Al usar **trim**, nos aseguramos de que los datos ingresados por el usuario sean limpiados de cualquier espacio en blanco adicional antes de ser utilizados, lo que mejora la consistencia y evita problemas no deseados en la aplicación.

- ## stripslashes

  Este método se utiliza para eliminar las barras invertidas que podrían haber sido agregadas a las entradas de datos por la función magic_quotes_gpc de PHP (que ahora está obsoleta y se ha eliminado en PHP 5.4).

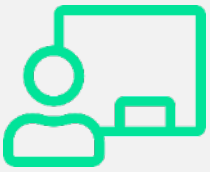
  La función magic_quotes_gpc solía escapar automáticamente las comillas simples y dobles en los datos enviados mediante formularios.

  Sin embargo, esto puede provocar problemas cuando se trabaja con bases de datos que requieren datos sin escapar.

  Al utilizar **stripslashes**, aseguramos que los datos ingresados por el usuario estén limpios de barras invertidas no deseadas y evitamos posibles errores en las consultas SQL.

Al combinar estos métodos en la función **limpiarDatosUsuario**, como se muestra en el ejemplo del CRUD anterior, podemos garantizar que los datos ingresados por los usuarios sean seguros, consistentes y libres de caracteres especiales que puedan afectar negativamente la aplicación o exponerla a vulnerabilidades.

Es importante aplicar estos métodos en todas las entradas de datos del usuario antes de utilizarlos en consultas SQL o mostrarlos en la página para mantener la seguridad y la integridad de la aplicación web.



Veamos ahora:
Código del CRUD refactorizado nuevamente con los métodos mysqli_real_scape_string, htmlspecialchars, trim y stripslashes trabajando en conjunto.

```php
<?php

  // Configuración de la conexión a la base de datos
  $servidor = "localhost";
  $usuario = "root";
  $clave = "";
  $basededatos = "crud_mysqli";

  // Establecer la conexión a la base de datos
  $mysqli = mysqli_connect($servidor, $usuario, $clave, $basededatos);

  // Comprobar si se produjo algún error durante la conexión
  if(!$mysqli){
    die("Error de conexión: " . mysqli_connect_error());
  }

  // Función para mostrar un mensaje de éxito o error
  function mostrarMensaje($mensaje, $tipo = 'success'){
    echo "<div style='padding: 10px; background-color: ";
    if($tipo === 'success'){
      echo 'green';
    }
    else{
```

```
      echo 'red';
    }
    echo "; color: white; text-align: center;'>$mensaje</div>";
  }

  // Leer todos los registros de la tabla "usuarios"
  function leerUsuarios($mysqli){
    $query = "SELECT * FROM usuarios";
    $result = mysqli_query($mysqli, $query);
    return $result;
  }

  // Insertar un nuevo registro en la tabla "usuarios"
  function insertarUsuario($mysqli, $nombre, $email, $edad){
    $nombre = limpiarDatosUsuario($nombre);
    $email = limpiarDatosUsuario($email);
    $nombre = mysqli_real_escape_string($mysqli, $nombre);
    $email = mysqli_real_escape_string($mysqli, $email);
    $query = "INSERT INTO usuarios(nombre, email, edad)
VALUES('$nombre', '$email', $edad);";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro insertado correctamente.");
    }
    else{
      mostrarMensaje("Error al insertar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Actualizar un registro existente en la tabla "usuarios"
  function actualizarUsuario($mysqli, $id, $nombre, $email, $edad) {
    $nombre = limpiarDatosUsuario($nombre);
    $email = limpiarDatosUsuario($email);
    $nombre = mysqli_real_escape_string($mysqli, $nombre);
    $email = mysqli_real_escape_string($mysqli, $email);
    $query = "UPDATE usuarios SET nombre = '$nombre', email =
'$email', edad = $edad WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
      mostrarMensaje("Registro actualizado correctamente.");
    }
    else{
      mostrarMensaje("Error al actualizar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Eliminar un registro de la tabla "usuarios"
  function eliminarUsuario($mysqli, $id){
    $query = "DELETE FROM usuarios WHERE id = $id";
    $result = mysqli_query($mysqli, $query);
    if($result){
        mostrarMensaje("Registro eliminado correctamente.");
    }
```

```
    else{
      mostrarMensaje("Error al eliminar el registro: " .
mysqli_error($mysqli), 'error');
    }
  }

  // Limpiar los datos del usuario
  function limpiarDatosUsuario($dato){
    $dato = trim($dato);
    $dato = htmlspecialchars(stripslashes($dato));
    return $dato;
  }

  // Procesar formularios
  if($_SERVER['REQUEST_METHOD'] === 'POST'){
    if(isset($_POST['insertar'])){
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      insertarUsuario($mysqli, $nombre, $email, $edad);
    }
    elseif(isset($_POST['actualizar'])){
      $id = $_POST['id'];
      $nombre = $_POST['nombre'];
      $email = $_POST['email'];
      $edad = $_POST['edad'];
      actualizarUsuario($mysqli, $id, $nombre, $email, $edad);
    }
    elseif(isset($_POST['eliminar'])){
      $id = $_POST['id'];
      eliminarUsuario($mysqli, $id);
    }
  }

  // Obtener todos los usuarios de la tabla "usuarios"
  $result = leerUsuarios($mysqli);

?>

<!DOCTYPE html>
<html>
<head>
    <title>CRUD MySQLi</title>
</head>
<body>
  <h1>CRUD MySQLi - Usuarios</h1>
  <h2>Insertar nuevo usuario:</h2>
  <form method="post">
    <label for="nombre">Nombre:</label>
    <input type="text" name="nombre" required>
    <label for="email">Email:</label>
    <input type="email" name="email" required>
    <label for="edad">Edad:</label>
    <input type="number" name="edad" required>
    <input type="submit" name="insertar" value="Insertar">
```

```
  </form>

  <h2>Actualizar usuario:</h2>
  <?php while($usuario = mysqli_fetch_assoc($result)): ?>
    <form method="post">
      <input type="hidden" name="id" value="<?php echo $usuario['id'];
?>">
      <label for="nombre">Nombre:</label>
      <input type="text" name="nombre" value="<?php echo
$usuario['nombre']; ?>" required>
      <label for="email">Email:</label>
      <input type="email" name="email" value="<?php echo
$usuario['email']; ?>" required>
      <label for="edad">Edad:</label>
      <input type="number" name="edad" value="<?php echo
$usuario['edad']; ?>" required>
      <input type="submit" name="actualizar" value="Actualizar">
      <input type="submit" name="eliminar" value="Eliminar">
    </form>
  <?php endwhile; ?>

  <?php

    // Cerrar la conexión a la base de datos
    mysqli_close($mysqli);

  ?>

</body>
</html>
```

Hemos llegado así al final de esta clase en la que vimos:

- CRUD a una base de datos MySQL.
- Agregando seguridad a la conexión.
- Mejorando aún más la seguridad de la aplicación.

Te esperamos en la **clase en vivo** de esta semana.
No olvides realizar el **desafío semanal**.

¡Hasta la próxima clase!

Bibliografía

Documentación Oficial de PHP

W3Schools: PHP Tutorial

Cabezas Granado, Luis. González Lozano, F. (2018) Desarrollo web con PHP y MySQL. Anaya Multimedia.

Heurtel, O. (2016) Desarrolle un sitio web dinámico e interactivo. Eni Ediciones.

Welling, Luke Thompson, L. (2017) Desarrollo Web con PHP y MySQL. Quinta Edición (2017, Editorial Anaya).

Para ampliar la información

PHP: mysqli_real_scape_string

PHP: The Right Way

Todo sobre PHP

PHP Programming Language Tutorial - Full Course

PHP Full Course for non-haters

CURSO de php desde cero

MDN: Introducción al lado Servidor

Guía de HTML