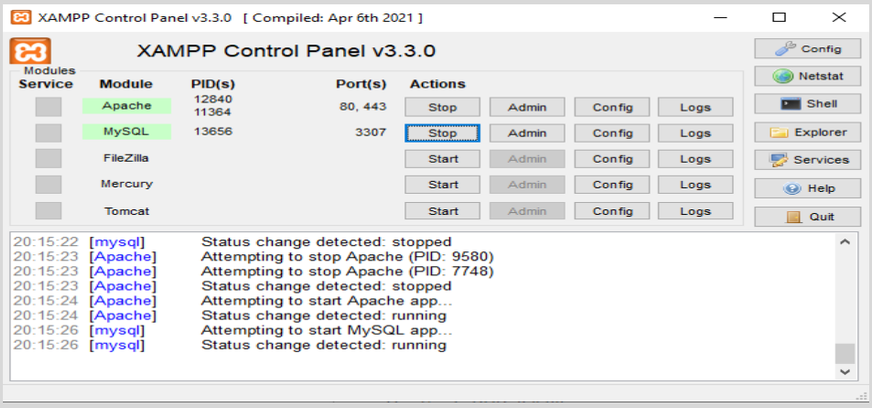
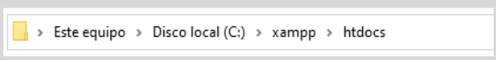
PASO A PASO CRUD

HTML-CSS-MYSQL-PHP

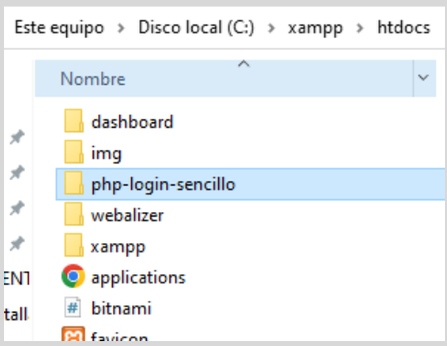
Paso1. Inicializar los servicios de Xampp para los módulos de Apache y Mysql, es decir damos clic en Start.



Paso2. Debemos crear la carpeta de nuestro proyecto en la siguiente ruta:



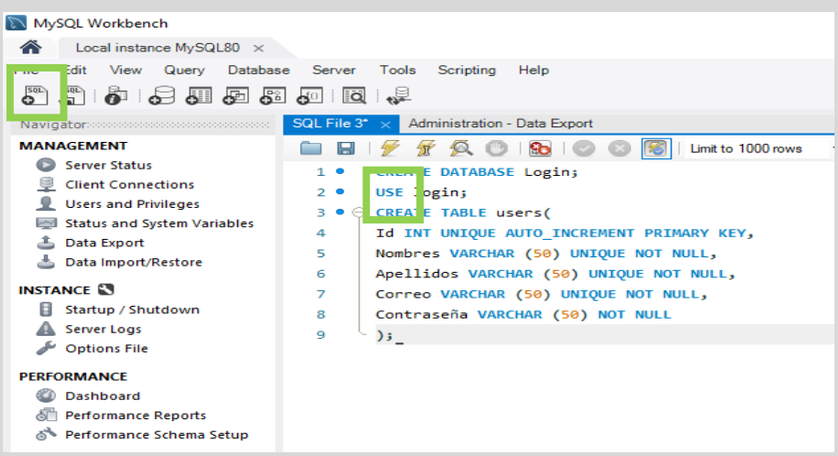
En mi caso cree la siguiente carpeta para alojar los archivos correspondientes:



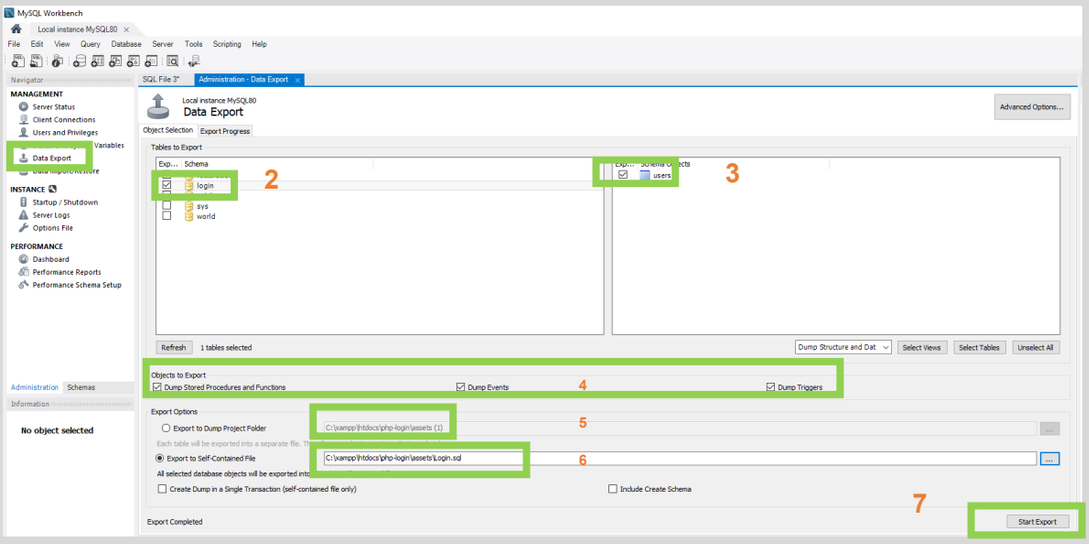
Paso3. Debemos crear nuestra base de datos, para esto tenemos dos opciones, crearla directamente phpMyAdmin o realizarla en Mysql.

Entonces, vamos a hacerlo de las dos maneras.

3.1. MySql Workbench



Realizamos el export, pero antes crearíamos una carpeta llamada “assets” dentro de la carpeta de nuestro proyecto para guardar el archivo del export.

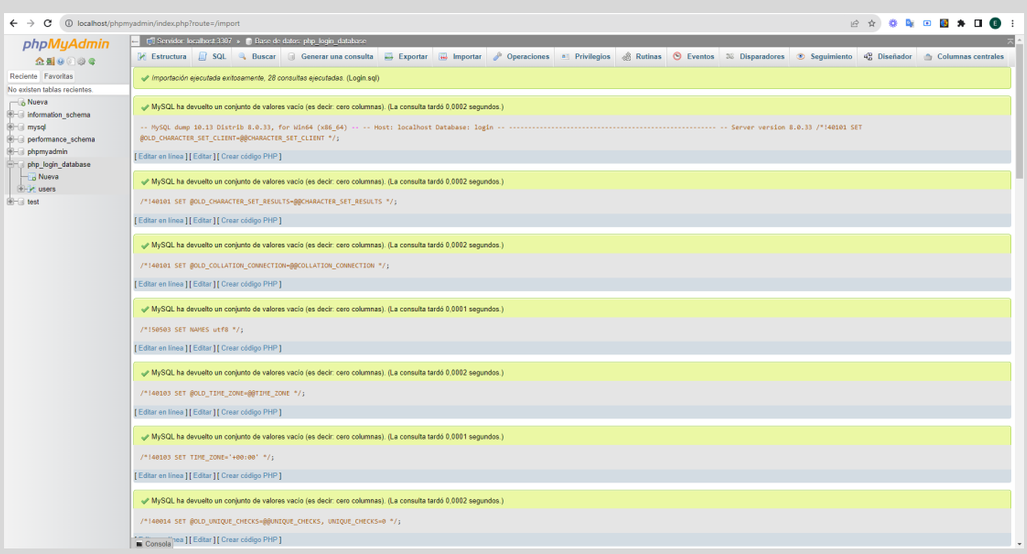


Ahora importamos a PhpMyAdmin, para trabajarla. Si tenemos los servicios de Xampp correctamente iniciados, desde el navegador ingresamos a: http://localhost/phpmyadmin/

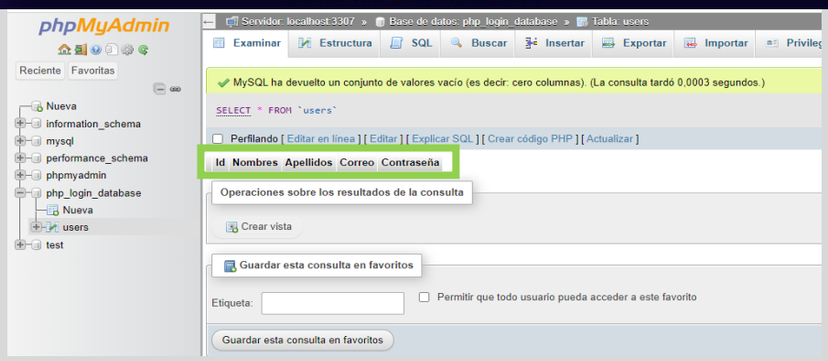
Debemos crear la base de datos para lograr realizar el import, para esto damos clic en “Nueva” en el panel izquierdo, le asignamos el nombre que usaremos en este caso será “php_login_database”, y en la lista desplegable usamos el utf8mb4_unicode_ci (es una codificación que permite almacenar y procesar texto en varios idiomas, incluyendo caracteres de 4 bytes como emojis, y realiza comparaciones de texto sin distinguir entre mayúsculas y minúsculas. Es una configuración comúnmente recomendada para bases de datos que necesitan manejar contenido multilingüe y diverso). Para que este importe sea exitoso también debemos asegurarnos que en el archivo sql este debidamente asignada la codificación así:

```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

En la parte superior ingresamos a “Importar”, seleccionamos el archivo .sql creado y damos clic en “Importar”



Ya tendríamos lista la base de datos para trabajar con la opción 1:

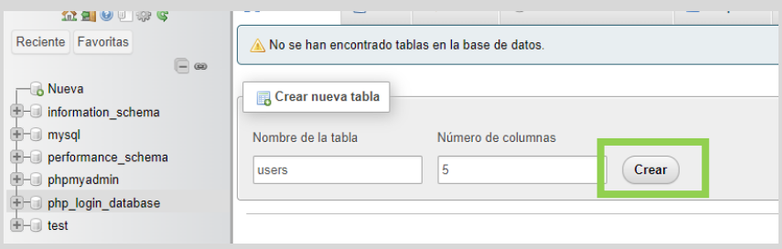


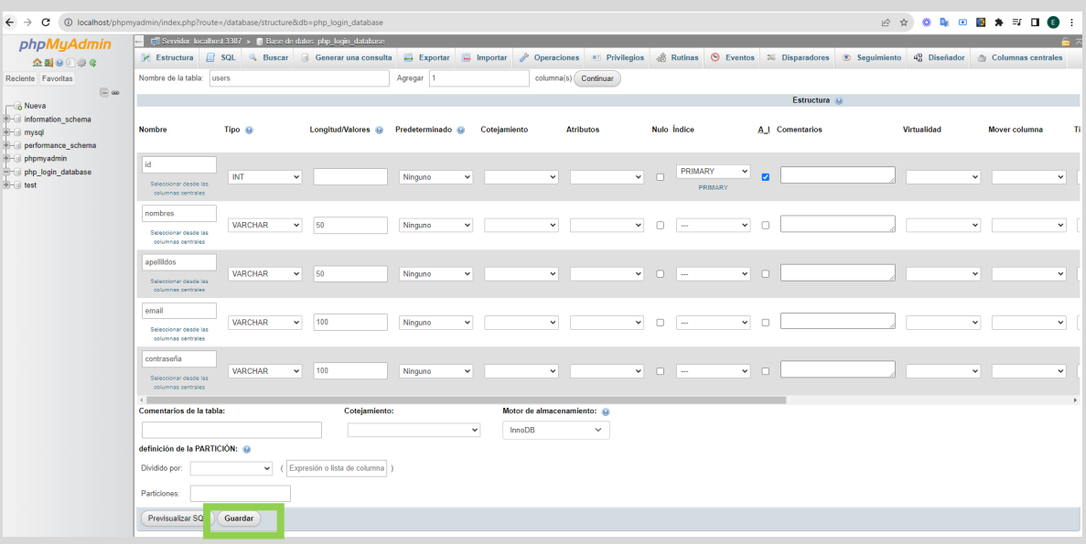
3.2. Crear la base desde PhpMyAdmin

Damos clic en “Nueva” en el panel izquierdo, le asignamos el nombre que usaremos en este caso será “php_login_database”, y en la lista desplegable usamos el utf8mb4_unicode_ci (es una codificación que permite almacenar y procesar texto en varios idiomas, incluyendo caracteres de 4 bytes como emojis, y realiza comparaciones de texto sin distinguir entre mayúsculas y minúsculas. Es una configuración comúnmente recomendada para bases de datos que necesitan manejar contenido multilingüe y diverso). Para que este importe sea exitoso también debemos asegurarnos que en el archivo sql esté debidamente asignada la codificación así:

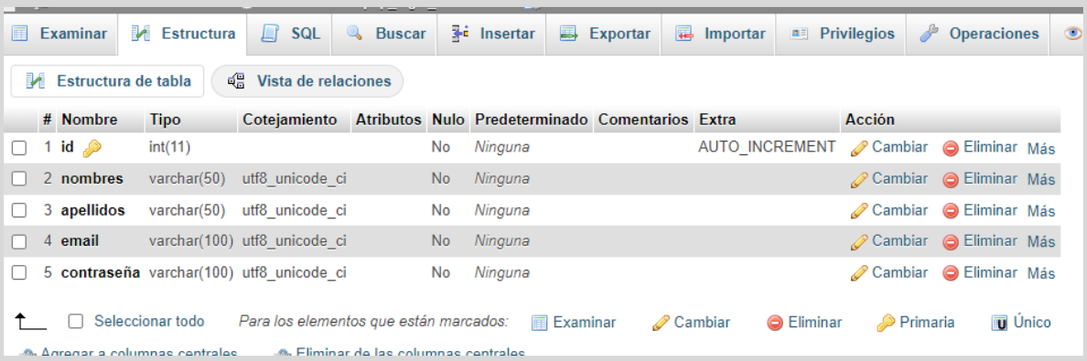
```
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Creamos la tabla con 5 columnas: Id, Nombre, Apellido, Correo y Contraseña

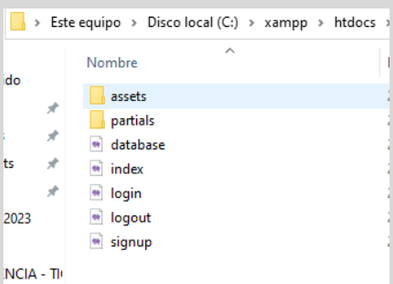




Así quedaría:



Paso4. Debemos crear 4 archivos ".php": database, index, login, logout, signup (Hagan uso del editor de código de su preferencia) y creen 2 carpetas llamadas: Assets (Activos – donde se guardará el archivo de estilos "Style.css" y el archivo con la base o estructura de nuestra base de datos "Login.sql") y Partials (parciales – aquí crearemos un archivo llamado "header.php" donde tendremos el encabezado).



Paso5. Arrancaremos por crear el archivo "database.php":

```php
<?php

$server = 'localhost:3307';
$username = 'root';
$password = '';
$database = 'php_login_database';

try {
  $conn = new PDO("mysql:host=$server;dbname=$database;", $username,
$password);
} catch (PDOException $e) {
  die('Connection Failed: ' . $e->getMessage());
}

?>
```

Este código en PHP se encarga de establecer una conexión a una base de datos MySQL utilizando la extensión PDO (PHP Data Objects).

Explicación del código

Las primeras líneas definen los datos de conexión necesarios para conectarse a la base de datos:

$server: Especifica la dirección del servidor de la base de datos y el número de puerto. En este caso, el servidor está en localhost (es decir, en la misma máquina donde se ejecuta el código) y utiliza el puerto 3307.

$username: Es el nombre de usuario utilizado para acceder a la base de datos. En este caso, se establece como 'root', que es un usuario comúnmente utilizado para fines de desarrollo.

$password: Es la contraseña asociada al usuario especificado. En este caso, la contraseña está en blanco, lo que puede ser común en entornos locales de desarrollo, pero en producción se debe utilizar una contraseña segura.

$database: Es el nombre de la base de datos a la que se desea conectar. En este caso, la base de datos se llama 'php_login_database'.

La sección try-catch es un bloque de código que permite manejar excepciones que puedan ocurrir durante la conexión a la base de datos. En este caso, se intenta crear un objeto PDO con los datos de conexión proporcionados:

new PDO(...): Crea un nuevo objeto PDO para interactuar con la base de datos.

"mysql:host=$server;dbname=$database;": Es una cadena de conexión que especifica el tipo de base de datos (mysql), la dirección del servidor ($server) y el nombre de la base de datos ($database) al que se conectará.

$username y $password: Se pasan como argumentos para autenticar la conexión a la base de datos.

Si la conexión es exitosa, no ocurre ningún problema y el bloque try se ejecuta sin errores. En caso de que haya un error durante la conexión, se captura la excepción (PDOException $e) y se ejecuta el bloque catch. El mensaje de error se imprime mediante die(), que finaliza la ejecución del programa mostrando el mensaje de error.

Paso6. Organizaremos nuestro header.php.

```
<header>
  <h2><a href ="index.php">Nuestro primer proyecto CADPH - Instructora
Eliana Yineth Lozano Triana</a></h2>
</header>
```

Este fragmento de código HTML representa un encabezado (<header>) de nuestra página web.

Explicación del código:

<header>: Es una etiqueta HTML que se utiliza para definir la sección de encabezado de una página web. Generalmente, se coloca en la parte superior de la página y contiene elementos como logotipos, títulos, enlaces de navegación u otra información importante.

<h2>: Es una etiqueta HTML de encabezado de nivel 2. Representa un título secundario en la jerarquía de encabezados. Los encabezados de nivel 2 son más pequeños y menos importantes que los encabezados de nivel 1 (<h1>) pero más importantes que los de nivel 3 (<h3>), y así sucesivamente.

...: Es una etiqueta de enlace (hipervínculo) en HTML. La parte importante aquí es el atributo href, que indica la URL o dirección a la que apunta el enlace. En este caso, el enlace apunta al archivo index.php.

El texto entre las etiquetas <a>: Es el contenido del enlace. Es lo que se muestra al usuario y representa el texto que el usuario puede hacer clic para acceder al recurso vinculado (en este caso, el archivo index.php).

Paso7. Continuaremos con el archivo “index.php”.

```php
<?php
  session_start();

  require 'database.php';

  if (isset($_SESSION['user_id'])) {
    $records = $conn->prepare('SELECT id, email, password FROM users WHERE
id = :id');
    $records->bindParam(':id', $_SESSION['user_id']);
    $records->execute();
    $results = $records->fetch(PDO::FETCH_ASSOC);

    $user = null;

    if (count($results) > 0) {
      $user = $results;
    }
  }
?>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Bienvenidos aprendices</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto"
rel="stylesheet">
    <link rel="stylesheet" href="assets/css/style.css">
  </head>
  <body>
    <?php require 'partials/header.php' ?>

    <?php if(!empty($user)): ?>
      <br> Bienvenido. <?= $user['email']; ?>
      <br>El login fue exitoso.
      <a href="logout.php"> Cierre de sesión </a>
```

```
    <?php else: ?>
      <h3>Si ya tienes cuenta, ingresa!! si no la tienes, te invito a
regitrarte</h3>

      <a href="login.php"> Ingreso</a> o
      <a href="signup.php">Registro</a>
    <?php endif; ?>
    <div style="display: flex; justify-content: center; align-items: center;
height: 70vh;">
      <img src="partials/img.png" alt="algo bonito">
    </div>
   </body>
</html>
```

Este código en PHP muestra una página web que verifica si el usuario ha iniciado sesión y, en función de eso, muestra diferentes contenidos.

Explicación del código:

session_start();: Esta línea inicia una sesión de PHP. Es necesario llamar a esta función antes de usar variables de sesión en el código.

require 'database.php';: Se utiliza para incluir el archivo 'database.php' que contiene la conexión a la base de datos. Esto permite acceder al objeto $conn, que se utiliza para realizar consultas a la base de datos.

if (isset($_SESSION['user_id'])) { ... }: Verifica si existe una variable de sesión llamada 'user_id'. Si esta variable existe, significa que el usuario ha iniciado sesión previamente.

Dentro del bloque if, se realiza una consulta a la base de datos para obtener información sobre el usuario que ha iniciado sesión:

Se prepara una consulta SQL para seleccionar el id, email y contraseña del usuario cuyo id coincida con el valor almacenado en $_SESSION['user_id'].

Se utiliza el método bindParam() para vincular el valor de $_SESSION['user_id'] al marcador :id en la consulta preparada.

La consulta se ejecuta utilizando el método execute() y los resultados se obtienen mediante fetch(PDO::FETCH_ASSOC), almacenándolos en la variable $results.

Se crea una variable $user inicializada como null, que se utilizará para almacenar la información del usuario si se encuentra en la base de datos.

Si se encuentran resultados en la consulta (es decir, el usuario existe en la base de datos), se asigna el array $results a la variable $user.

A continuación, se muestra el contenido de la página dependiendo del estado del inicio de sesión:

Si $user no está vacío (es decir, el usuario ha iniciado sesión), se muestra un mensaje de bienvenida junto con el correo electrónico del usuario y un enlace para cerrar sesión.

Si $user está vacío (es decir, el usuario no ha iniciado sesión), se muestra un mensaje invitando al usuario a iniciar sesión si ya tiene una cuenta o a registrarse si no la tiene.

Se muestra una imagen en la página utilizando la etiqueta <img>. La imagen se encuentra en el archivo 'partials/img.png' y se muestra centrada en la página.

Paso8. Construiremos lo correspondiente al registro de usuario: signup.php.

```php
<?php

require 'database.php';

$message = '';

if (!empty($_POST['email']) && !empty($_POST['password']) &&
!empty($_POST['nombres']) &&  !empty($_POST['apellidos'])) {
  $sql = "INSERT INTO users (nombres, apellidos, email, password) VALUES
(:nombres, :apellidos, :email, :password)";
  $stmt = $conn->prepare($sql);
  $stmt->bindParam(':nombres', $_POST['nombres']);
  $stmt->bindParam(':apellidos', $_POST['apellidos']);
  $stmt->bindParam(':email', $_POST['email']);
  $password = password_hash($_POST['password'], PASSWORD_BCRYPT);
  $stmt->bindParam(':password', $password);

  if ($stmt->execute()) {
    $message = 'El usuario fue creado exitosamente';
  } else {
    $message = 'Lo sentimos! pero hubo un problema al crear su cuenta';
  }
}
?>
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <title>SignUp</title>
  <link href="https://fonts.googleapis.com/css?family=Roboto"
rel="stylesheet">
  <link rel="stylesheet" href="assets/css/style.css">
</head>

<body>

  <?php require 'partials/header.php' ?>

  <?php if (!empty($message)): ?>
    <p>
      <?= $message ?>
    </p>
  <?php endif; ?>
```

```
  <h1>Formulario de registro</h1>


  <form action="signup.php" method="POST">
    <input name="nombres" type="text" placeholder="Ingrese su nombre">
    <input name="apellidos" type="text" placeholder="Ingrese su apellido">
    <input name="email" type="text" placeholder="Ingrese su correo">
    <input name="password" type="password" placeholder="Ingrese su
contraseña">
     <h3>De acuerdo con los Terminos y Condiciones</h3>
    <input type="submit" value="Guardar">
  </form>
  <a href="login.php">¿Ya tengo una cuenta?</a></span>

</body>

</html>
```

Este código en PHP muestra un formulario de registro que permite a los usuarios crear una cuenta en un sitio web. Vamos a explicar cómo funciona:

require 'database.php';: Esta línea incluye el archivo 'database.php', que contiene la conexión a la base de datos MySQL utilizando PDO. Esto permite acceder al objeto $conn, necesario para realizar consultas a la base de datos.

Se declara una variable $message inicializada como una cadena vacía. Esta variable se utilizará para almacenar mensajes que se mostrarán al usuario después de enviar el formulario.

Se comprueba si se ha enviado el formulario y si todos los campos ($_POST['email'], $_POST['password'], $_POST['nombres'] y $_POST['apellidos']) no están vacíos utilizando !empty().

Si todos los campos tienen datos (es decir, el formulario ha sido enviado con la información necesaria), se procede a insertar un nuevo registro en la tabla de la base de datos 'users'.

Se prepara una consulta SQL para realizar la inserción utilizando la sintaxis INSERT INTO. Se vinculan los valores de los campos del formulario utilizando marcadores de posición :nombres, :apellidos, :email y :password.

La contraseña se cifra usando la función password_hash() antes de ser almacenada en la base de datos. Esto es una práctica segura para almacenar contraseñas.

Se ejecuta la consulta utilizando el método execute() del objeto preparado $stmt. Si la ejecución es exitosa, se establece el mensaje de éxito en la variable $message, de lo contrario, se establece el mensaje de error.

A continuación, se muestra el contenido de la página web:

- Si la variable $message no está vacía, se muestra el mensaje (éxito o error) después de enviar el formulario.
- Se muestra un título "Formulario de registro" y un formulario con campos para ingresar nombres, apellidos, correo electrónico y contraseña.
- Un botón de envío "Guardar" que envía los datos ingresados en el formulario al mismo script 'signup.php' para procesar el registro.
- Un enlace para aquellos que ya tienen una cuenta y desean iniciar sesión.

Paso9. Posterior trabajaremos con el archivo de login.php.

```php
<?php

  session_start();

  if (isset($_SESSION['user_id'])) {
    header('Location: /php-login');
  }
  require 'database.php';

  if (!empty($_POST['email']) && !empty($_POST['password'])) {
    $records = $conn->prepare('SELECT id, email, password FROM users WHERE
email = :email');
    $records->bindParam(':email', $_POST['email']);
    $records->execute();
    $results = $records->fetch(PDO::FETCH_ASSOC);

    $message = '';

    if (count($results) > 0 && password_verify($_POST['password'],
$results['password'])) {
      $_SESSION['user_id'] = $results['id'];
      header('Location: /php-login-sencillo/index.php');
    } else {
      $message = 'Lo sentimos, la contraseña ingresada no coincide';
    }
  }

?>

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Login</title>
    <link href="https://fonts.googleapis.com/css?family=Roboto"
rel="stylesheet">
    <link rel="stylesheet" href="assets/css/style.css">
  </head>
  <body>
    <?php require 'partials/header.php' ?>

    <?php if(!empty($message)): ?>
      <p> <?= $message ?></p>
    <?php endif; ?>

    <h1>Ingresar</h1>
    <span>o <a href="signup.php">Registrarse</a></span>

    <form action="login.php" method="POST">
      <input name="email" type="text" placeholder="Ingrese su correo">
      <input name="password" type="password" placeholder="Ingrese su
contraseña">
```

```
      <input type="submit" value="Guardar">
    </form>
  </body>
</html>
```

Este código en PHP representa una página de inicio de sesión (login) para nuestro sitio web que utiliza sesiones para gestionar la autenticación de usuarios.

Explicación del código:

session_start();: Esta línea inicia una sesión de PHP, lo cual es necesario para poder utilizar variables de sesión. Las sesiones permiten mantener la información del usuario durante varias páginas y sesiones en el sitio.

if (isset($_SESSION['user_id'])) { ... }: Esta parte del código verifica si ya hay un usuario autenticado en la sesión. Si existe la variable de sesión $_SESSION['user_id'], significa que el usuario ya ha iniciado sesión previamente. En este caso, se redirige al usuario a la página /php-login utilizando header('Location: /php-login'). La redirección impide que un usuario autenticado pueda volver a la página de inicio de sesión mientras mantiene la sesión activa.

require 'database.php';: Se incluye el archivo 'database.php', que contiene la conexión a la base de datos MySQL utilizando PDO. Esto permitirá realizar consultas a la base de datos para verificar las credenciales del usuario.

A continuación, se comprueba si se ha enviado el formulario de inicio de sesión ($_POST['email'] y $_POST['password'] no están vacíos). Si el formulario ha sido enviado, se procede a verificar las credenciales ingresadas por el usuario.

Se realiza una consulta SQL para buscar un registro en la tabla 'users' que coincida con el correo electrónico (email) proporcionado en el formulario. La consulta se prepara con un marcador de posición :email, que luego se vincula al valor del campo $_POST['email'] utilizando bindParam().

Se ejecuta la consulta utilizando execute() y se almacenan los resultados en la variable $results como un array asociativo utilizando fetch(PDO::FETCH_ASSOC).

Se declara una variable $message inicializada como una cadena vacía. Esta variable se utilizará para almacenar mensajes de error si las credenciales ingresadas no son válidas.

Se verifica si se encontraron resultados en la consulta y si la contraseña proporcionada ($_POST['password']) coincide con la contraseña almacenada en la base de datos utilizando password_verify(). Si ambos criterios se cumplen, significa que las credenciales son válidas.

Si las credenciales son válidas, se establece la variable de sesión $_SESSION['user_id'] con el valor del campo 'id' del usuario obtenido de la base de datos. Luego, se redirige al usuario a la página /php-login-sencillo/index.php utilizando header('Location: /php-login-sencillo/index.php'). Esta redirección llevará al usuario a una página de inicio de sesión exitosa.

Si las credenciales no son válidas, se establece un mensaje de error en la variable $message.

A continuación, se muestra el contenido de la página web:

- Si la variable $message no está vacía, se muestra el mensaje de error después de enviar el formulario.
- Se muestra un título "Ingresar" y un formulario con campos para ingresar el correo electrónico y la contraseña.
- Un botón de envío "Guardar" que envía los datos ingresados en el formulario al mismo script 'login.php' para procesar el inicio de sesión.
- Un enlace para aquellos que no tienen una cuenta y desean registrarse.

Paso10. Acá como se construye el inicio de sesión, se debe destruir la misma, por lo que se trabajará en el archivo logout.php.

```php
<?php
  session_start();

  session_unset();

  session_destroy();

  header('Location: /php-login-sencillo/index.php');
?>
```

Este fragmento de código en PHP se utiliza para cerrar la sesión de un usuario en nuestro sitio web. Permíteme explicar cada parte:

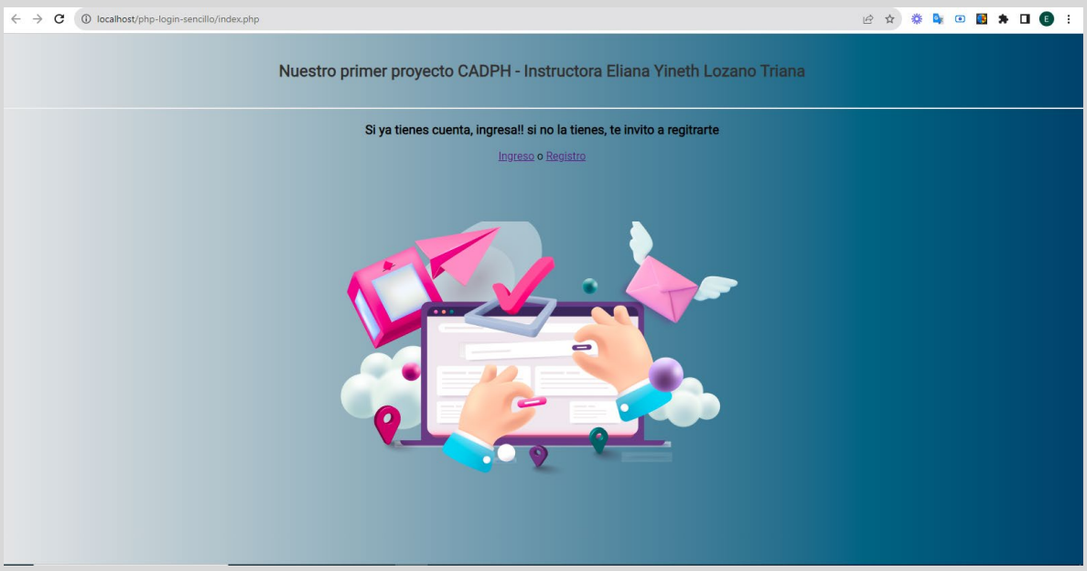
session_start();: Esta línea inicia una sesión de PHP, lo cual es necesario para poder utilizar variables de sesión. Las sesiones permiten mantener información del usuario durante varias páginas y sesiones en el sitio.

session_unset();: La función session_unset() se utiliza para eliminar todos los datos almacenados en la sesión. Esto significa que todas las variables de sesión y su contenido serán eliminadas, pero la sesión en sí seguirá activa.

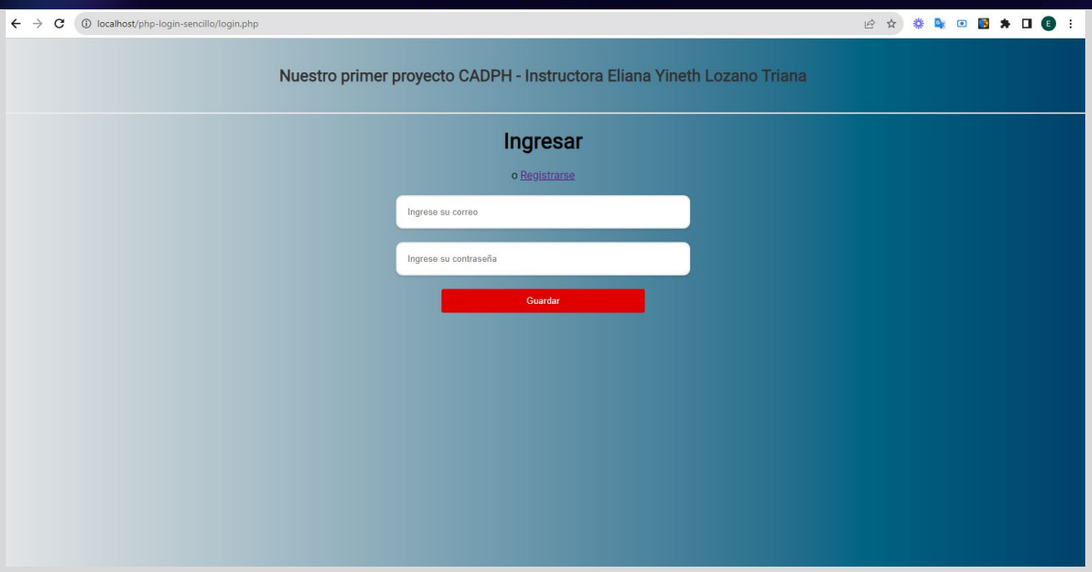
session_destroy();: La función session_destroy() se utiliza para destruir la sesión actual. Esto eliminará completamente todos los datos de la sesión, además de eliminar la cookie de sesión del lado del cliente. Después de ejecutar esta función, la sesión se considera terminada.

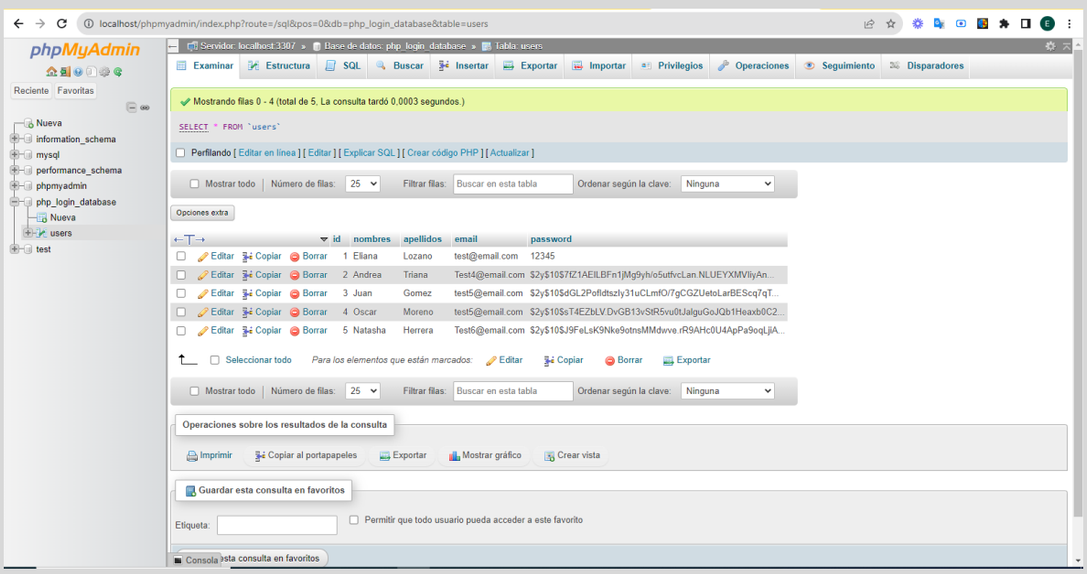
header('Location: /php-login-sencillo/index.php');: Esta línea redirige al usuario a la página 'index.php' ubicada en el directorio '/php-login-sencillo/'. La redirección ocurre después de haber destruido la sesión, lo que significa que el usuario ya no estará autenticado y se le llevará de vuelta a la página de inicio de sesión.

En resumen, nuestro sitio debería de quedar así:




Bases de datos
ELIANA YINETH
LOZANO TRIANA
eylozano@sena.edu.co


localhost/php-login-sencillo/login.php
Nuestro primer proyecto CADPH - Instructora Eliana Yineth Lozano Triana
Ingresar
o Registrarse
Ingrese su correo
Ingrese su contraseña
Guardar


localhost/phpmyadmin/index.php?route=/sql&pos=0&db=php_login_database&table=users
phpMyAdmin
Reciente
Favoritas
Nueva
information_schema
mysql
performance_schema
phpmyadmin
php_login_database
Nueva
users
test
Examinar
Estructura
SQL
Buscar
Insertar
Exportar
Importar
Privilegios
Operaciones
Seguimiento
Disparadores
Mostrando filas 0 - 4 (total de 5, La consulta tardó 0,0003 segundos.)
SELECT * FROM `users`
Perfilando [Editar en línea] [Editar] [Explicar SQL] [Crear código PHP] [Actualizar]
Mostrar todo
Número de filas: 25
Filtrar filas:
Buscar en esta tabla
Ordenar según la clave:
Ninguna
Opciones extra
id
nombres
apellidos
email
password
1 Eliana Lozano test@email.com 12345
2 Andrea Triana Test4@email.com $2y$10$7fZ1AEILBFn1jMg9yh/o5utfvcLan.NLUEYXMVliyAn...
3 Juan Gomez test5@email.com $2y$10$dGL2PofIdtszly31uCLmfO/7gCGZUetoLarBEScq7qT...
4 Oscar Moreno test6@email.com $2y$10$sT4EZbLV.DvGB13vSlR5vu0tJalguGoJQb1Heaxb0C2...
5 Natasha Herrera Test6@email.com $2y$10$J9FoLsK9Nke9otnsMMdwve.rR9AHc0U4ApPa9oqLjiA...
Seleccionar todo
Para los elementos que están marcados:
Editar
Copiar
Borrar
Exportar
Operaciones sobre los resultados de la consulta
Imprimir
Copiar al portapapeles
Exportar
Mostrar gráfico
Crear vista
Guardar esta consulta en favoritos
Etiqueta:
Permitir que todo usuario pueda acceder a este favorito
Consola