UNIVERSITAT DE BARCELONA

Facultad de Matemáticas e Informática
Grado en Ingeniería Informática

Computación Orientada a la Web

Entrega Sección 2-3

Alumno: William D. Sotto

Memoria de la Práctica - Sección 2-3

1 - Introducción

El objetivo de esta práctica es extender la funcionalidad del sistema de reservas de hoteles desarrollado en la Práctica 1, añadiendo la capa de servidor con PHP y la integración con una base de datos MySQL.

Los cambios se dividen en dos apartados:

- **Apartado 1:** Implementar el envío de información entre cliente y servidor mediante PHP, utilizando formularios y validaciones de datos.
- **Apartado 2:** Incorporar el uso de MySQL mediante PhpMyAdmin para gestionar clientes y ciudades, permitiendo la inserción y consulta de datos desde la base de datos.

Para ello, he seguido las recomendaciones de la clase de prácticas. Mi enfoque estará en la información del usuario y de los hoteles:
El signUp donde el Usuario envía datos por un formulario y la lista de hoteles del Backend.

2 - Apartado 1: Envío de información entre cliente y servidor

En este apartado, he implementado la comunicación entre el cliente y el servidor a través de un formulario de registro (`sign-up.php`), que envía datos al servidor (`server.php`) mediante el método `POST`.

2.1 Validación de datos

Para garantizar la seguridad y coherencia de los datos ingresados, aplico validaciones tanto en el **cliente** (HTML5) como en el **servidor** (PHP).

Validaciones en el cliente (HTML5)

Los campos del formulario incluyen restricciones con `pattern` y `required`:

```
<input type="text" name="username" pattern="[A-Za-z0-9]{3,20}" required title="Solo letras y números, entre 3 y 20 caracteres.">
<input type="email" name="email" required>
<input type="tel" name="phone" pattern="[0-9]{9,15}" required title="Solo números, entre 9 y 15 dígitos.">
```

Validaciones en el servidor (PHP)

Una vez que el usuario envía los datos, server.php los procesa y aplica validaciones adicionales:

```php
$username = $_POST['username'];
$password = $_POST['password'];
$email = $_POST['email'];
$phone = $_POST['phone'];

// Validar nombre
if (!preg_match('/^[A-Za-z0-9]{3,20}$/', $username)) {
  die("❌ Error: El nombre debe contener solo letras y números (3-20 caracteres). ");
}

// Validar email con filtro de PHP
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  die("❌ Error: Email no válido.");
}

// Validar teléfono
if (!preg_match('/^[0-9]{9,15}$/', $phone)) {
  die("❌ Error: El teléfono debe contener entre 9 y 15 dígitos numéricos.");
}
```

Estas validaciones aseguran que los datos sean correctos antes de ser almacenados o procesados.

Ejemplo de expresiones regulares alternativas que pude encontrar en Internet:

- **Nombre:** `'/^[A-Za-z\s]{3,30}$/'` (Permite espacios en nombres compuestos).
- **Teléfono internacional:** `'/^\+?[0-9]{9,15}$/'` (Opcionalmente permite el código de país con +).
- **Contraseña segura:** `'/^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/'` (Mínimo 8 caracteres, al menos una mayúscula, un número y un símbolo especial).

3 - Apartado 2: Integración con MySQL

3.1 Base de Datos y PhpMyAdmin

Para la gestión de los datos, he modificado la base de datos que nos daban. Ya que las imágenes no estaban actualizadas en la tabla `hoteles`.
Ahora al comienzo de mi código actualizo algunos hoteles con imágenes reales respectivas para cada hotel con la conexión.

Además, el archivo `start-backend.php` comprueba si las tablas existen y, si no, las crea automáticamente.

Un problema con el que tuve que encontrarme es que al hacer el Sign Up de los usuarios, la base de datos que nos daban, no permitía tener un autoIncremento del ID en la tabla de usuarios, así que alteré también la tabla de Users para poder seguir haciendo más pruebas con la página:

```
//Alterar la tabla `users`
$sql = "ALTER TABLE users MODIFY id int(10) UNSIGNED NOT NULL AUTO_INCREMENT";
$conn->exec($sql);
```

3.2 Inserción y Consulta de Datos

- Después de la validación por PHP que presenté anteriormente, el usuario puede añadir su cuenta a la base de datos SOLO si no existe un usuario existente. Para hacer esta comprobación, realizo un SELECT en mi tabla user con el email y el nombre del usuario, para ver si alguna coincide con algún usuario ya guardado con anterioridad:

```
// Verificar si el usuario ya existe
$stmt = $conn->prepare("SELECT id FROM users WHERE email = ? OR name = ?");
$stmt->execute([$email, $username]);
if ($stmt->rowCount() > 0) {
  return "El usuario ya existe en nuestra base de datos.";
}
// Insertar usuario
$sql = "INSERT INTO users (`name`, `email`, `password`, `descripcion`, `phone`)
    VALUES (?, ?, ?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->execute([$username, $email, $hashed_password, $desc, $phone]);

return true;}
```

(Todo eso está en `create_user.php`)

- Para mostrar la lista de ciudades disponibles en un desplegable:

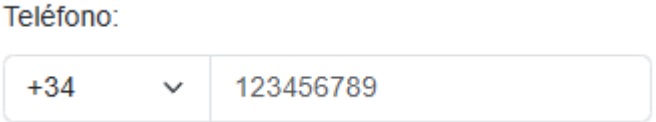
```php
require 'db.php';
// Variable de Conexión a Base de Datos
global $conn;

$result = $conn->query( statement: 'SELECT * FROM hoteles');

while ($row = $result->fetch( mode: PDO::FETCH_ASSOC)) {
    echo '<div class="col-12 col-md-6 col-lg-4 mb-4">';
    echo '  <div class="card">';
    echo '    <img src="' . htmlspecialchars($row['img']) . '" class="card-img-top" alt="' . htmlspecialchars($row['nombre']) . '">';
    echo '    <div class="card-body">';
    echo '      <h5 class="card-title">' . htmlspecialchars($row['nombre']) . '</h5>';
    echo '      <p><strong>Ubicación:</strong> ' . htmlspecialchars($row['ciudad']) . ', ' . htmlspecialchars($row['pais']) . '</p>';
    echo '      <p><strong>Zona:</strong> ' . htmlspecialchars($row['zona']) . '</p>';
    echo '      <p><strong>Piscina:</strong> ' . ($row['piscina'] ? '✅ Sí' : '❌ No') . '</p>';
    echo '    </div>';
    echo '  </div>';
    echo '</div>';
}
```

He utilizado una clase `db.php` el cual hace la conexión con el Backend, de esta forma evito repetir el código constantemente. Luego, selecciono todos los hoteles del backend y los muestro al usuario. Utilizo el ***htmlspecialchars*** como vimos en clases para evitar que hayan strings con valores como “h2”, “div” o demás que modifiquen mi vista.

- Además de esto también añadí al Backend un “country_codes” para poder añadir un desplegable para el usuario cuando añade su teléfono móvil. Así que en el sign up también hago un SELECT de estos valores.

```php
<label for="phone" class="form-label">Teléfono:</label>
<div class="input-group">
  <select class="form-select" name="country_code" required style="flex: 0.3;">
    <?php
    require_once '../db.php';
    global $conn;
    try {
      $stmt = $conn->query( statement: 'SELECT code FROM country_codes ORDER BY code ASC');
      while ($row = $stmt->fetch( mode: PDO::FETCH_ASSOC)) {
        $selected = ($row['code'] == '+34') ? 'selected' : '';
        echo '<option value="' . $row['code'] . '" ' . $selected . '>' . $row['code'] . '</option>';
      }
    } catch (PDOException $e) {
      echo '<option value="">❌ Error al cargar</option>';
    }
    ?>
  </select>
  <input type="tel" class="form-control" id="phone" name="phone" required
      pattern="[0-9]{9,15}" title="Solo números, entre 9 y 15 dígitos." placeholder="123456789">
</div>
```

Vista del Usuario:



4 - Conclusión

Esta práctica ha permitido llevar el sistema de reservas a un nivel superior:

Páginas PHP dínámicas en lugar de HTML estático.
Manejo de datos con MySQL en lugar de listas estáticas.
Validación de formularios con REGEXP, asegurando datos coherentes.

Esta práctica ha sido un gran avance sobre la versión anterior y me ha permitido entender la importancia de trabajar con PHP y bases de datos de manera segura y estructurada. Aún así, creo que podría haber evitado más repetición de código, pero aún no estoy del todo familiarizado con PHP. Se mejorará definitivamente.