IIC2413

AYUDANTÍA 12

WEB + PHP + SQL

¿QUÉ VEREMOS?

1.Concepto de servidor de web y conexión desde un browser (URL)

2. Concepto de URL a un programa PHP desde el browser

3. Conexión desde un programa PHP a la base de datos de postgresql

4. Cómo se ejecutan los queries de SQL dentro de PHP

¿Qué es un servidor web?

Es un programa que recibe **peticiones** HTTP (generalmente a través del **puerto** 80 para HTTP o **443** para **HTTPS**) y devuelve respuestas, que pueden ser archivos HTML, imágenes, o resultados de programas ejecutados.

Conexión desde un browser

- Escribes una URL en el navegador (ej: https://www.ejemplo.com/pagina.php)
- El navegador contacta al servidor web en el puerto 443 (HTTPS)
- El servidor procesa la petición y devuelve una respuesta
- El navegador muestra el resultado

Métodos HTTP

1. **GET**
    - Envía los datos en la URL.
    - Se usa típicamente para leer información, no modificarla.

Ejemplo de URL con GET:

https://ejemplo.com/buscar.php?**query=libros&orden=precio**

Lo que aparece después del ? se llama cadena de consulta (query string).Cada parámetro tiene un nombre y un valor:

- query = libros
- orden = precio

En PHP, se puede acceder a esos valores con**:**

```
$_GET['query']    // devuelve "libros"
$_GET['orden']    // devuelve "precio"
```

Métodos HTTP

2. POST

- Envía los datos ocultos en el cuerpo de la solicitud HTTP (no aparecen en la URL).
- Se usa para enviar información sensible o modificar datos, como contraseñas, formularios de registro, etc.

Ejemplo:

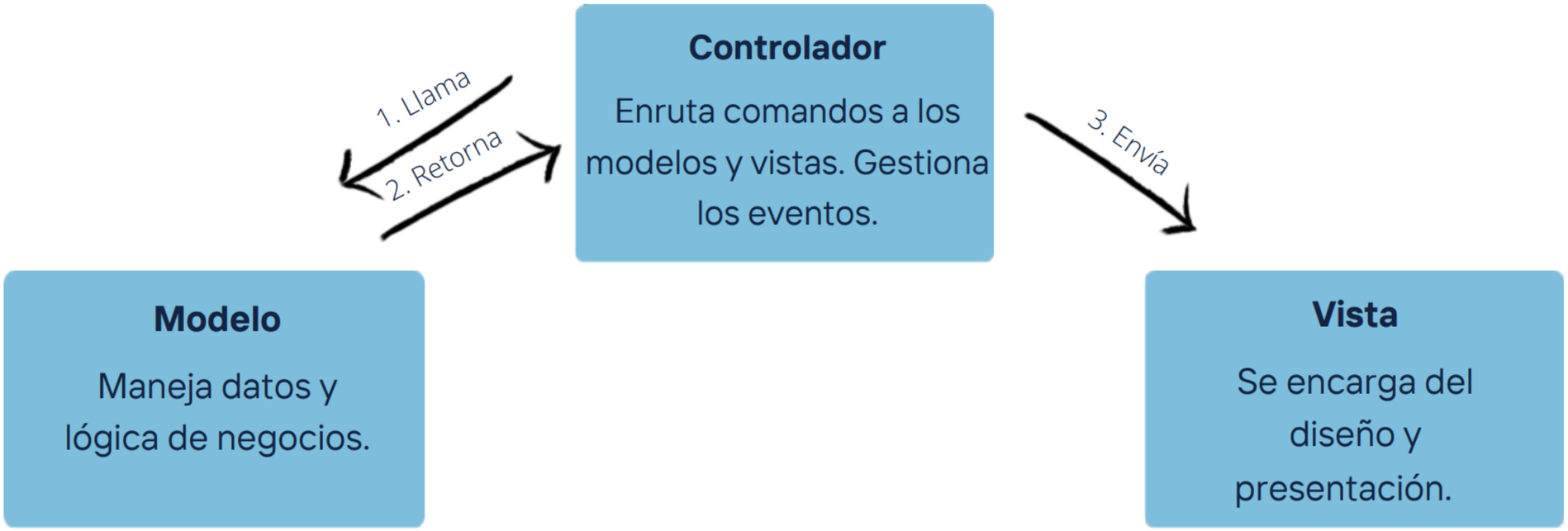
```
<form action="login.php" method="POST">
  <input type="text" name="usuario">
  <input type="password" name="clave">
  <button type="submit">Iniciar sesión</button>
</form>
```

En PHP, se puede acceder a esos valores con:

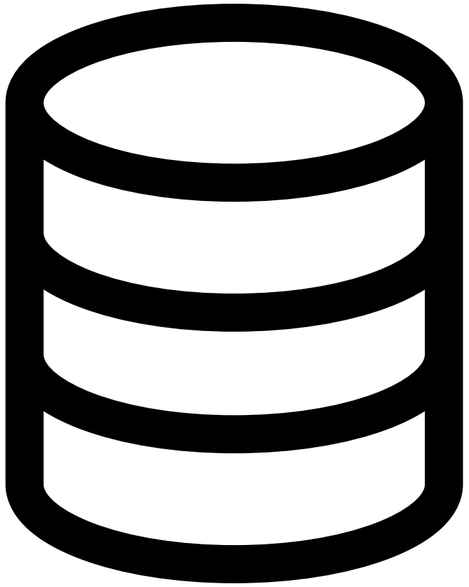
$_POST['usuario']
$_POST['clave']

Modelo Vista Controlador

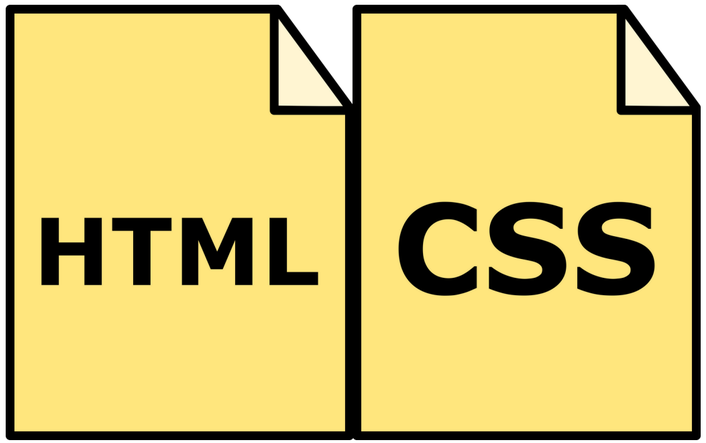
MVC (Modelo-Vista-Controlador) es un patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Tiene 3 partes:



MVC en la Entrega 3



Modelo
Tablas, esquema relacional (BDD y consultas sql)



Vista
Página web, en este caso HTML + CSS



Controlador
Lenguaje de programación con soporte para bases de datos (PHP)

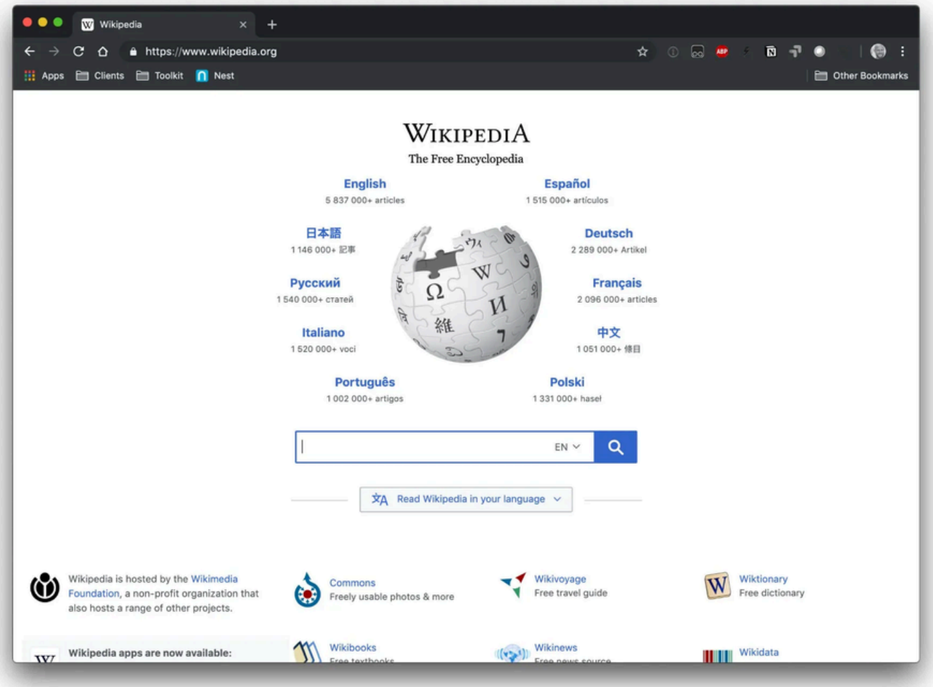
HTML

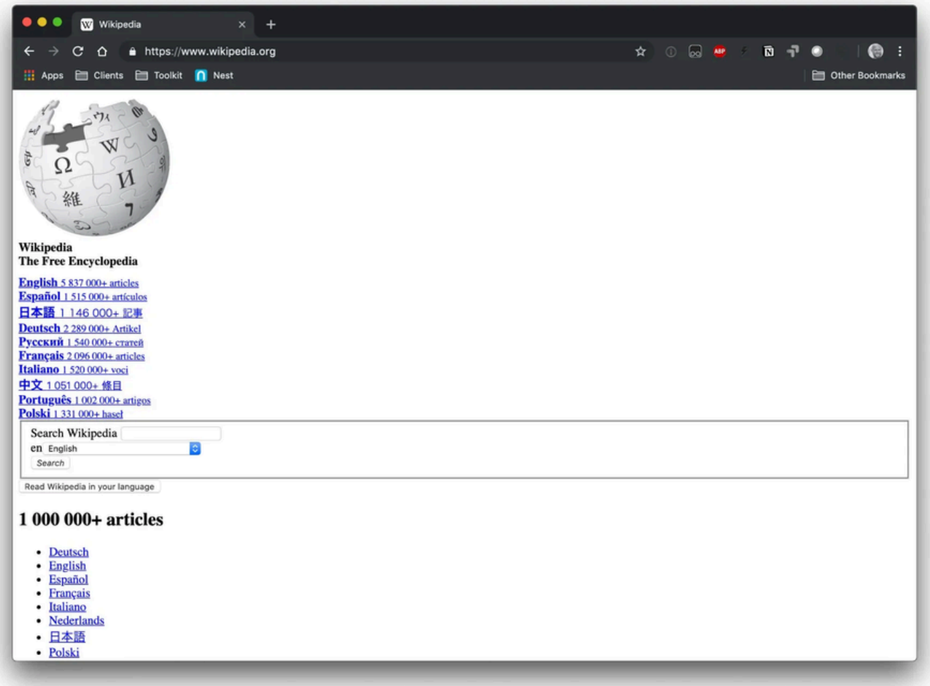
HTML (HyperText Markup Language) es el **lenguaje de marcado** utilizado para crear la **estructura básica de una página web**.
No es un lenguaje de programación (como PHP o JavaScript), sino un lenguaje que organiza el contenido en la web usando **etiquetas**.

Las etiquetas ocupan el siguiente syntax:
<tag> contenido **</tag>**

HTML solo entrega la estructura básica. CSS le da cierto estilo a la página.

HTML + CSS





Ejemplo de página web con y sin CSS.

Etiquetas HTML archivo index.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Inicio de sesión - Booked.com</title>
    <link rel="stylesheet" href="../css/style.css">
</head>
```

Etiquetas HTML archivo index.php

- **<!DOCTYPE html>** :Le dice al navegador que este documento está escrito en HTML5, la versión actual del lenguaje.
- **<html** lang="es"**> ... </html>:** Es el elemento raíz del documento HTML. Todo lo que se muestra en la página está dentro de esta etiqueta.
- **<head> ... </head>:** Contiene metadatos (información sobre la página) que no se muestran directamente al usuario.
- **<meta charset="UTF-8">**: Define la codificación de caracteres, en este caso UTF-8, que permite mostrar correctamente acentos, ñ y caracteres especiales.
- **<title>**Inicio de sesión - Booked.com**</title>**: Define el título de la pestaña del navegador.
- **<link** rel="stylesheet" href="../css/style.css"**>**: Conecta un archivo externo de CSS (hojas de estilo) para aplicar diseño. href es la ruta al archivo

Etiquetas HTML archivo index.php

```
<body>
    <div class="container">
        <h1>Bienvenido a Booked.com</h1>
        <form action="validar_login.php" method="POST" class="formulario">
            <label for="usuario">Usuario:</label>
            <input type="text" id="usuario" name="usuario" required>

            <label for="contrasena">Contraseña:</label>
            <input type="password" id="contrasena" name="contrasena" required>

            <button type="submit">Iniciar sesión</button>
        </form>

        <p>¿No tienes cuenta? <a href="registro.php">Regístrate aquí</a></p>

        <?php if ($error): ?>
            <p class="error"><?= htmlspecialchars($error) ?></p>
        <?php endif; ?>
    </div>
</body>
```

Etiquetas HTML archivo index.php

- **<body> ... </body>:** Contiene todo el contenido visible de la página web
- **<div** class="container"**>** ... ** **: Crea una sección o contenedor. Es una etiqueta genérica para agrupar contenido. class="container" permite aplicar estilos desde CSS
- **<h1>**Bienvenido a Booked.com**</h1>**: Muestra un título principal de la página. Los títulos van desde <h1> (más importante) hasta <h6> (menos importante).
- **<form** action="validar_login.php" method="POST" class="formulario"> ..**</form>**

Crea un formulario para que el usuario pueda ingresar datos:

  - action="validar_login.php": archivo al que se enviarán los datos cuando se envíe el formulario.
  - method="POST": los datos se envían de forma oculta (no se ven en la URL).
  - class="formulario": clase CSS para aplicar estilos al formulario.

Etiquetas HTML archivo index.php

- **<label** for="usuario">Usuario:**</label>**: Etiqueta para el campo de entrada (input). for="usuario" conecta el label con el input que tiene id="usuario"

- **<input** type="text" id="usuario" name="usuario" required**>**: Campo donde el usuario escribe su nombre de usuario. type="text": campo de texto, id y name: identificadores que se usan en el HTML y para enviar datos al servidor, required: el campo es obligatorio.

- **<input** type="password" id="contrasena" name="contrasena" required**>**: Igual al anterior, pero el tipo es "password", por lo que el texto se oculta al escribirlo.

- **<button** type="submit"**>**Iniciar sesión**</button>**: Botón para enviar el formulario.

Ejecutar un programa PHP desde el browser

Para que un archivo PHP se ejecute al acceder desde el navegador el archivo debe estar en el directorio del servidor web. En su caso se ingresará a la aplicación desde la siguiente dirección:

https://stonebraker.ing.uc.cl/<usuariouc.e4>/E4

Esta ruta ejecutará o mostrará el contenido del archivo:

~/Sites/ruta/al/archivo.php

Ahora veamos un código de ejemplo

Ejercicio

1. Crear un archivo test.php

2. Escribir el siguiente código en ese archivo:

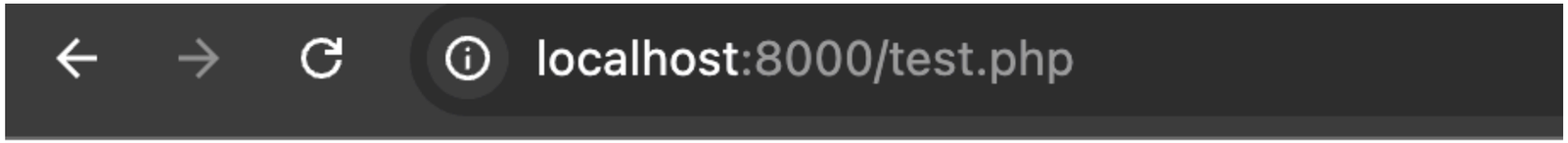
<?php echo "Hola, mundo desde PHP!";?>

3. Escribir en consola php -S localhost:8000

4. Escribir la siguiente URL en su navegador de preferencia

http://localhost:8000/test.php

Ejemplo



Hola mundo!

Instrucciones Básicas PHP

- **echo:** imprimir texto
- **$:** declaración de variable
- **array(var1, var2, ..., varn):** lista de objetos
- **//:** comentarios
- **foreach:** iterador
- **include:** importar otro módulos

Conexión desde PHP a PostgreSQL

```php
<?php
function conectarBD() {
    $host = 'localhost'; // Cambiar al servidor bdd1.ing.puc.cl si se quiere usar el servidor remoto
    $dbname = 'bdd'; // Nombre de usuario  Nombre bdd
    $usuario = 'username'; // Nombre de usuario
    $clave = 'contraseña'; // Número de alumno

    try {
        $db = new PDO("pgsql:host=$host;dbname=$dbname", $usuario, $clave);
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $db;
    } catch (PDOException $e) {
        echo "Error de conexión: " . $e->getMessage();
        exit();
    }
}
?>
```

Conexión desde PHP a PostgreSQL

1. Variables de conexión:

```php
$host = 'localhost'; // Cambiar al servidor bdd1.ing.puc.cl si se quiere usar el servidor remoto
$dbname = 'bdd'; // Nombre de usuario
$usuario = 'username'; // Nombre de usuario
$clave = 'contraseña'; // Número de alumno
```

2. Bloque try-catch para manejo de errores

```php
try {
    // Intento de conexion
} catch (PDOException $e) {
    // Manejo de errores
}
```

Conexión desde PHP a PostgreSQL

3. Conexión a la base de datos

```php
$db = new PDO("pgsql:host=$host;dbname=$dbname", $usuario, $clave);
```

- Se crea una nueva instancia de **PDO** (PHP Data Objects), que es una interfaz para acceder a bases de datos en PHP.

- El DSN (Data Source Name) especifica:
  - pgsql: El driver de PostgreSQL.
  - host: El servidor donde está la base de datos.
  - dbname: El nombre de la base de datos.

Conexión desde PHP a PostgreSQL

4. Configuración de atributos

```
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

- Establece el modo de error para que PDO lance excepciones cuando ocurran errores (PDO::ERRMODE_EXCEPTION).
- Esto permite un mejor manejo de errores mediante try-catch.

5. Retorno de la conexión

```
return $db;
```

- Si la conexión es exitosa, la función devuelve el objeto $db que representa la conexión.

Conexión desde PHP a PostgreSQL

6. Manejo de errores:

```
} catch (PDOException $e) {
    echo "Error de conexión: " . $e->getMessage();
    exit();
}
```

- Si ocurre un error, se captura la excepción (PDOException).
- Se muestra un mensaje de error que incluye la descripción del problema ($e->getMessage()).
- Se termina la ejecución del script con exit().

Conexión desde PHP a PostgreSQL

```php
<?php
function conectarBD() {
    $host = 'localhost'; // Cambiar al servidor bdd1.ing.puc.cl si se quiere usar el servidor remoto
    $dbname = 'bdd'; // Nombre de usuario
    $usuario = 'username'; // Nombre de usuario
    $clave = 'contraseña'; // Número de alumno

    try {
        $db = new PDO("pgsql:host=$host;dbname=$dbname", $usuario, $clave);
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $db;
    } catch (PDOException $e) {
        echo "Error de conexión: " . $e->getMessage();
        exit();
    }
}
?>
```

Ejecutar queries SQL en PHP

```php
// Consulta con parámetros (segura contra inyección SQL)
$query = "SELECT * FROM usuarios WHERE id = :id";
$stmt = $conn->prepare($query);
$stmt->bindParam(':id', $id_usuario);
$stmt->execute();

// Obtener resultados
$resultados = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Ejecutar queries SQL en PHP

1. **Preparación de la consulta SQL**

```php
$query = "SELECT * FROM usuarios WHERE id = :id";
```

Define una consulta SQL que selecciona todos los campos de la tabla usuarios donde el id coincide con un valor.

2. Preparar la sentencia

```php
$stmt = $conn->prepare($query);
```

- $stmt: Es un objeto de tipo PDOStatement que representa la consulta preparada.
- La consulta se compila en el servidor de base de datos antes de que se inserten los valores.

Ejecutar queries SQL en PHP

3. Vincular parámetro

```php
$stmt->bindParam(':id', $id_usuario);
```

Asocia el marcador :id con la variable $id_usuario

4. Ejecutar la consulta

```php
$stmt->execute();
```

Ejecuta la consulta preparada con los parámetros vinculados. La consulta se envía al servidor de base de datos con el valor real de $id_usuario.

Ejecutar queries SQL en PHP

5. Obtener resultados

```php
$resultados = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

- Recupera todas las filas del resultado de la consulta.
- PDO::FETCH_ASSOC:
  - Es una constante que indica que los resultados deben ser devueltos como un array asociativo
  - Las claves del array serán los nombres de las columnas

Hagamos un ejercicio con una BDD real!