

TAREA RA8

PREGUNTAS

1. *¿Qué diferencias, en términos de peticiones HTTP, existen entre la ejecución de código de lenguajes de servidor y la ejecución de código de lenguajes de cliente?*

La principal diferencias en términos de peticiones HTTP , es que en lenguaje servidor como PHP ejecutan el código servidor cuando recibe una solicitud HTTP generando así contenido dinámico que se enviá de vuelta al cliente como una respuesta HTML.

En lenguaje cliente como JavaScript se ejecutan en el navegador del cliente después de que el contenido HTML ha sido recibido permitiendo la manipulación del DOM sin necesidad de realizar nuevas peticiones al servidor

2. *Teniendo en cuenta lo explicado en el punto anterior, ¿qué desventajas tienen los lenguajes que se ejecutan en el servidor?*

Las principales desventajas son que incluyen una mayor latencia debido a la comunicación continua del servidor y por ello una carga mas alta al servidor. Ademas también sufre dependencia del servidor ya que si el servidor falla las funcionalidades se verán afectadas

3. *¿Los servidores web (Apache u otros) están configurados por defecto para ejecutar lenguajes del servidor (PHP u otros)? ¿Qué hemos hecho en clase para que Apache pueda ejecutar PHP?*

Los servidores web como apache no siempre estan configurados por defecto para poder ejecutar lenguajes del servidor como PHP

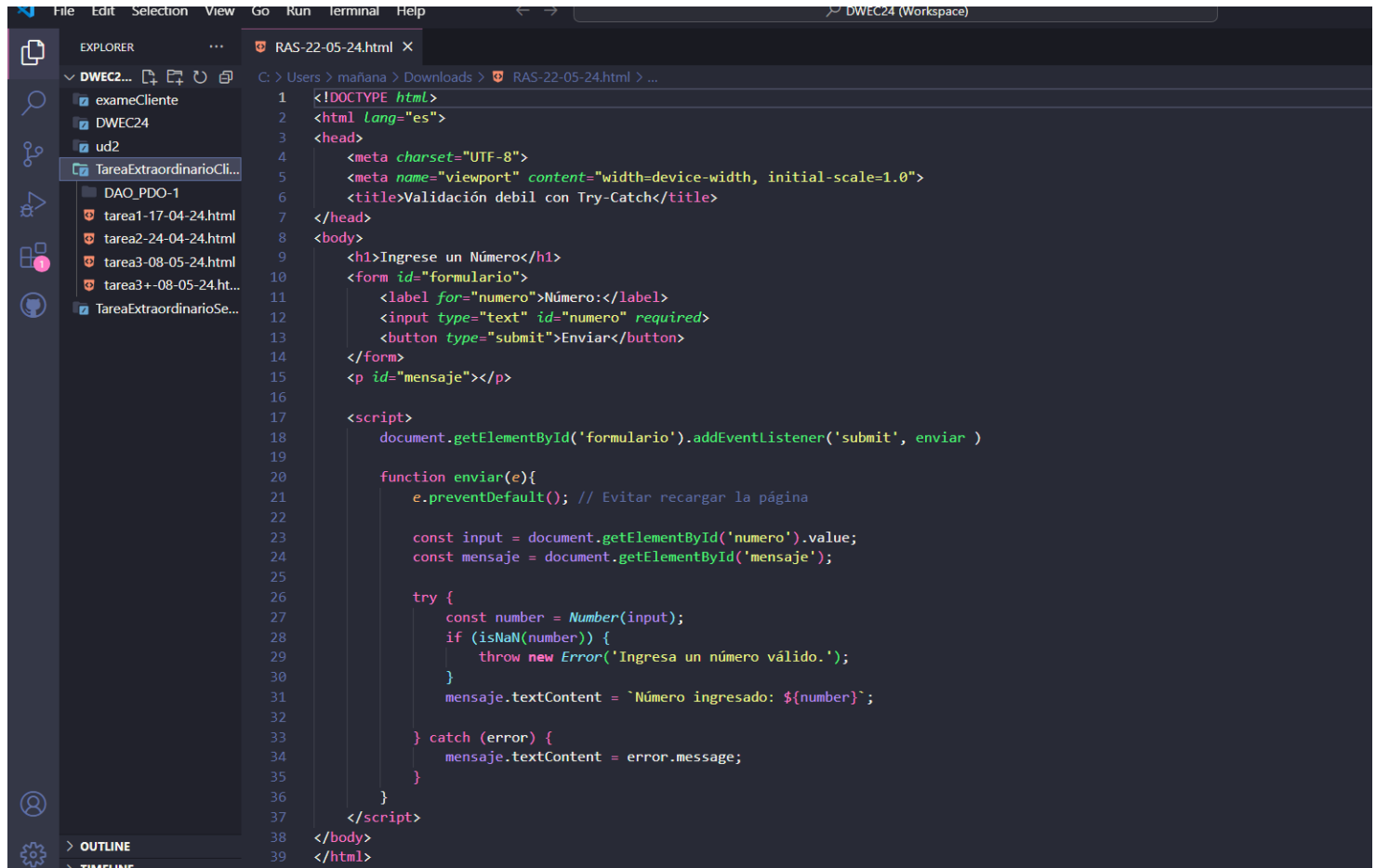
En clase lo que se hizo fue activar el modulo “mod_php” y utilizar XAMPP

4. *¿Es necesario configurar el servidor web para ejecutar lenguajes del lado del cliente? ¿Por qué?*

En el lado cliente no es necesario configurar el servidor web ya que para ejecutar lenguajes del lado cliente como JavaScript se utiliza el navegador del cliente

5. Cuando desarrollamos una aplicación web con formularios de entrada de datos, pon un ejemplo de validación que se podría hacer en el cliente y otro que necesariamente exija una conexión al servidor.

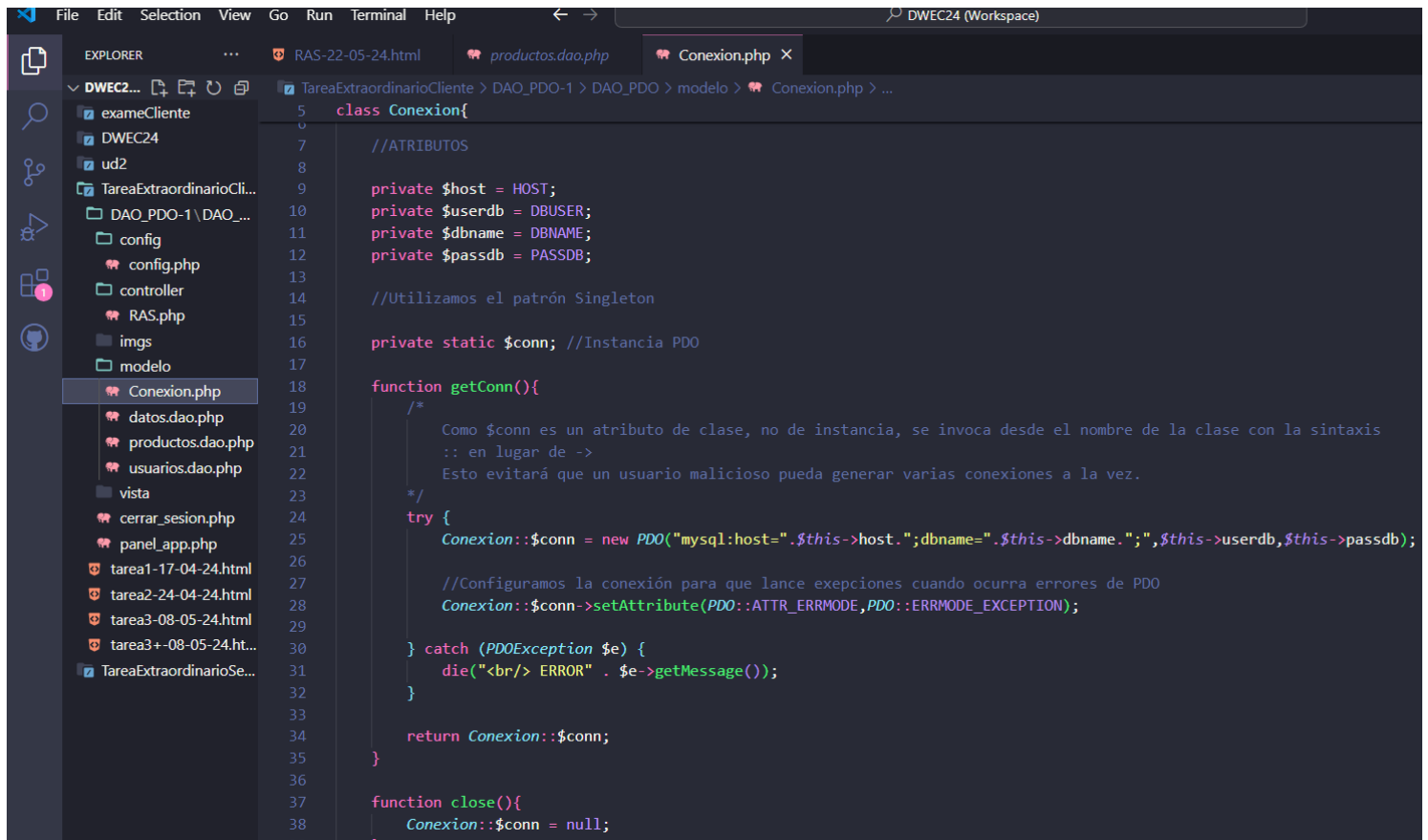
Para la validación en el lado cliente se puede utilizar JavaScript



```
1 <!DOCTYPE html>
2 <html Lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Validación debil con Try-Catch</title>
7 </head>
8 <body>
9   <h1>Ingresa un Número</h1>
10  <form id="formulario">
11    <label for="numero">Número:</label>
12    <input type="text" id="numero" required>
13    <button type="submit">Enviar</button>
14  </form>
15  <p id="mensaje"></p>
16
17  <script>
18    document.getElementById('formulario').addEventListener('submit', enviar )
19
20    function enviar(e){
21      e.preventDefault(); // Evitar recargar la página
22
23      const input = document.getElementById('numero').value;
24      const mensaje = document.getElementById('mensaje');
25
26      try {
27        const number = Number(input);
28        if (isNaN(number)) {
29          throw new Error('Ingresa un número válido.');
```

Este código se valida un form para validar que sea un numero valido el que se ingresa

Para la validación en el lado servidor se puede utilizar PHP donde utilizo un ejemplo de una app ya creada



The image shows a code editor with a dark theme. On the left is the Explorer sidebar showing a project structure with folders like 'config', 'controller', 'img', and 'modelo'. The 'Conexion.php' file is selected in the 'modelo' folder. The main editor area displays the code for the 'Conexion' class. The code includes private attributes for host, user, database name, and password, a Singleton pattern implementation with a static connection variable, a 'getConn()' method that creates a PDO connection and sets error handling, and a 'close()' method that nullifies the connection.

```
5 class Conexion{
6
7     //ATRIBUTOS
8
9     private $host = HOST;
10    private $userdb = DBUSER;
11    private $dbname = DBNAME;
12    private $passdb = PASSDB;
13
14    //Utilizamos el patrón Singleton
15
16    private static $conn; //Instancia PDO
17
18    function getConn(){
19        /*
20         Como $conn es un atributo de clase, no de instancia, se invoca desde el nombre de la clase con la sintaxis
21         :: en lugar de ->
22         Esto evitará que un usuario malicioso pueda generar varias conexiones a la vez.
23        */
24        try {
25            Conexion::$conn = new PDO("mysql:host=".$this->host.";dbname=".$this->dbname.";",$this->userdb,$this->passdb);
26
27            //Configuramos la conexión para que lance excepciones cuando ocurra errores de PDO
28            Conexion::$conn->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
29
30        } catch (PDOException $e) {
31            die("<br/> ERROR" . $e->getMessage());
32        }
33
34        return Conexion::$conn;
35    }
36
37    function close(){
38        Conexion::$conn = null;
39    }
40}
```

En este fragmento de código se realiza una conexión a la base de datos