**CICLO: DESARROLLO APLICACIONES WEB-DISTANCIA**
**MÓDULO: DESARROLLO DE APLICACIONES WEB EN ENTORNO SERVIDOR**
**TAREA: DWES-01**
**ALUMNO: DAVID MEDINA GARCIA**

1. **¿Qué tipo de páginas, estáticas o dinámicas, utilizarás para programar cada una de las páginas que componen tu aplicación? ¿Por qué?**

   -**Página de presentación**: Esta página será estática ya que solo necesita mostrar información sobre la aplicación y no requiere interacción del usuario más allá de proporcionar un enlace a la siguiente página.

   - **Página de introducción de datos**: Esta página será dinámica ya que necesita interactuar con el usuario para recoger su nombre y dirección de correo electrónico. Los datos ingresados por el usuario deben ser enviados al servidor para su procesamiento o almacenamiento.

   - **Página de visualización de datos**: Esta página será dinámica ya que necesita interactuar con el servidor para recuperar y mostrar la lista actualizada de nombres y direcciones de correo electrónico.

2. **Si en la página de introducción de datos quieres comprobar, antes de enviar los datos, que el correo electrónico introducido cumple unas ciertas normas (por ejemplo, que tiene una @), ¿qué tecnología/lenguaje utilizarás?**

   Para comprobar el formato de correo lo haríamos en la máquina cliente mediante **JavaScript** donde haríamos uso de una expresión regular con los caracteres permitidos: que comience por caracteres que no sean espacios en blanco, que contenga un solo símbolo de arroba, seguido ve de varios caracteres que no sean espacios en blanco, un punto y finaliza con caracteres que no sean espacios en blanco.
   Es decir, usaríamos la siguiente función:

```
function validarCorreo() {
    var correo = document.getElementById("correo").value;
    var expresionRegular = /\S+@\S+\.\S+/;

    if (!expresionRegular.test(correo)) {
       alert("Por favor, introduce un correo electrónico válido.");
       return false;
    }

    return true;

}
```

   Al hacerlo en el lado cliente hemos de advertir en algún lugar de la página que debe tener activo JavaScript, ya que de lo contrario no lo validaría y podría dar lugar al almacenamiento de una dirección errónea.
   Por ello la mayoría de las ocasiones nos encontramos con páginas que tras la solicitud de grabación de sus datos pide confirmación enviando un email a la dirección indicada para activar la orden de almacenamiento de los datos, y de esta forma, si se introduce una dirección errónea no podrá darse de alta.

3. **Si en esa misma página, ahora quieres comprobar que el correo electrónico introducido no se haya introducido anteriormente y ya figure en la lista, ¿qué tecnología/lenguaje utilizarás?**

   Para comprobar si el email está dado de alta se han de comprobar los datos con los almacenados en la **BD**, por lo que mediante un lenguaje de entorno servidor, como puede ser **php**, realizamos una conexión a la base de datos, por ejemplo **MySQL**, y comprobamos si el dato introducido ya se haya almacenado.

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Crear conexión
$conn = new mysqli($servername, $username, $password, $dbname);

// Verificar conexión
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$email = $_POST['email'];

$sql = "SELECT * FROM users WHERE email='$email'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  echo "Este correo electrónico ya ha sido registrado.";
} else {
  $sql = "INSERT INTO users (email) VALUES ('$email')";
  if ($conn->query($sql) === TRUE) {
    echo "¡Registro exitoso!";
  } else {
    echo "Error: " . $sql . "<br>" . $conn->error;
  }
}

$conn->close();
?>
```

4. **¿Qué arquitecturas puedes usar en el servidor para ejecutar la aplicación? ¿Cómo es el o los lenguajes que se usa en cada una de esas arquitecturas: de guiones, compilado a código nativo o compilado a código intermedio?**

   Como se trata de una aplicación pequeña que no precisa de requisitos muy específicos y que no exige demasiada rapidez, me decido a utilizar una arquitectura AMP (servidor web Apache, servidor de base de datos MySQL y lenguaje de programación PHP), que además todos sus componentes son de código libre, por lo que no necesitaremos realizar inversión alguna en la compra de dicha arquitectura.
   El PHP, al ser un código de guiones, se interpretará cada vez que se ejecute, teniendo como ventajas que no es necesario traducir el código fuente original para ser ejecutado, lo que aumenta su portabilidad y también se puede modificar su código sin necesidad de compiladores. Por el contrario tenemos que el proceso de interpretación ofrece un peor rendimiento que las otras alternativas., aunque esto es inapreciable para un pequeño/mediano volumen de datos.

5. **¿Qué parámetros debes tener en cuenta para decidirte por usar una arquitectura u otra?**
   Para decidirnos por una arquitectura u otra debemos pensar en:
   1. **Tamaño del proyecto**: En este caso es un proyecto pequeño.
   2. **Lenguajes que conozcamos**: PHP, Java y JavaScript, además de HTML y CSS. Por el volumen del proyecto no se aconseja invertir dinero en el aprendizaje de nuevos lenguajes o perfeccionamiento de los conocidos, sino que hemos de realizar la programación de forma rápida, precisa y con la justa inversión en renovación de conocimientos.
   3. **Código abierto o propietario**: Al ser un proyecto pequeño se aconseja la utilización de código abierto y libre, que nos permita evitar la inversión en software.
   4. **Programaré sólo o con un equipo**: El volumen del proyecto permite que lo haga en solitario.
   5. **Tengo un servidor web o gestor de base de datos o decido utilizar el que crea necesario**: Utilizaría un servidor web de arquitectura AMP donde alojar todo el proyecto.
   6. **Licencia que tendrá la aplicación**: De vuelta al minimalismo de la aplicación optaré por una licencia GPL de código abierto.

6. **Si te decides por utilizar una arquitectura AMP para la aplicación ¿qué componentes necesitas instalar en tu servidor para ejecutar la aplicación? Indica algún producto concreto para cada componente.**

   La arquitectura AMP estará compuesta por:

   - Un **servidor web** como por ejemplo Apache (versión 2.4.57) donde se almacenarán todos los ficheros que compongan el sitio completo
   - Un **gestor de bases de datos** como el MySQL (versión 8.1.0) que nos valdrá para almacenar de forma constante la información que haya de emitir o recibir el sitio
   - Un **lenguaje de programación** como el lenguaje de guiones PHP en su (versión 8.2.11) con el que comprobaremos en el lado servidor la validez de los datos introducidos, se envían los datos a la base de datos y se solicita la información a la misma.

   Para poder probar el proyecto podemos instalar en nuestro equipo un servidor web de tipo AMP haciendo uso de un paquete de tipo **XAMPP** (versión 8.2.4) que integra todos los componentes anteriormente mencionados.

7. **¿Qué necesitas instalar en tu ordenador para poder desarrollar la aplicación?**
   Para la realización del proyecto necesitaremos crear nuestro propio servidor instalando todos los programas indicados en el punto anterior, por lo que podemos optar por descargarnos e instalar el paquete XAMPP que es gratuito y nos permite añadir, a todo lo anterior, el **servidor FTP Filezilla**, el **servidor de aplicaciones Tomcat** y el **lenguaje Perl**.
   También debemos tener un **navegador web** (Chrome, Edge, Opera…) y un **Editor de código (IDE)** como Eclipse, Netbeans o VS Code.

8. **Si utilizas el lenguaje PHP para programar la aplicación, ¿cuál será el tipo de datos se utilizará para manipular cada una de las direcciones de correo?**

   Las direcciones de correo pueden estar compuestas por caracteres alfabéticos, numéricos y especiales, por lo que el tipo de datos será el de cadena (String), aunque en php no es necesario declararla.