

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

La primera página debería ser estática ya que no necesita obtener datos de ninguna fuente externa. La segunda debería ser dinámica ya que va a ser la encargada de enviar datos a una base de datos para almacenarlo o comprobar si ya existe ese correo electrónico dado de alta. Por último, la página final ha de ser de tipo dinámica ya que ha de obtener los datos almacenados en la base de datos para mostrarlos por pantalla.

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 tecleado lo haríamos en la máquina cliente mediante JavaScript donde haríamos uso de una expresión regular con los caracteres permitidos: que comience por letra o número, que contenga un solo símbolo de arroba, que finalice en un punto y dos (.es, .it, .uk, ...), tres (.com, .net, .org, ...) o cuatro (.info, .mobi, ...) caracteres, etc. Es decir, que usaríamos algo al estilo:

```
function emailValido(texto){
    if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3,4})+$/ .test(texto)) {
        alert("La dirección de email "+texto+" es correcta.");
    } else {
        alert("La dirección de email es incorrecta.");
    }
}
```

Podríamos utilizar php en el servidor, pero cargaríamos a éste con un proceso más cuando la única utilidad que queremos obtener es la validez de una dirección email. 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. Algo al estilo:

```
$sql="select email from tabla where email=$email";
$resultado=mysql_query($sql,$conexion);
if(mysql_num_rows($resultado)!=0){
    echo "Ya existe un usuario con ese email";
} else {
    echo "Podemos grabar los datos";
}
```

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 característica positiva el que se pueda modificar su código sin necesidad de compiladores, pero en contra tenemos

que la interpretación se realiza cada vez que queramos ejecutar el fichero, y nunca lo tenemos traducido a código máquina de manera definitiva, por lo que su proceso se ralentiza, aunque esto es inapreciable para un pequeño/mediano volumen de datos.

Si la aplicación fuese más compleja podríamos decantarnos por usar una arquitectura Java EE con un servidor de aplicaciones JOnAS que es de código abierto, multisistema y que posee un contenedor web incrustado Tomcat, igualmente de código abierto. Usaríamos páginas JSP que sería un código compilado a código intermedio, con lo que ganaríamos en rapidez con respecto a la utilización de php en la anterior arquitectura mencionada.

Si nos decantamos por una arquitectura propietaria, podemos utilizar ASP.Net con su servidor web IIS de Microsoft que integra soporte para manejar peticiones de páginas dinámicas ASP y ASP.NET siendo este último un lenguaje compilado a código nativo, aumentando la velocidad de proceso de las aplicaciones, pero encareciendo demasiado el producto final. Utilizaríamos el servidor de bases de datos SQL Server de Microsoft, igualmente propietario.

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:

- Tamaño del proyecto:** Que en este caso es bastante pequeño
- Lenguajes que conozcamos:** PHP, algo de Java y algo de ASP, por lo que el volumen del proyecto no nos 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.
- Código abierto o propietario:** Al igual que en los anteriores puntos, el volumen reducido del proyecto aconseja la utilización de código abierto y libre, que nos permita eludir la inversión en software.
- Programaré sólo o con un equipo:** El nivel del proyecto me permite abordarlo en solitario.
- Tengo un servidor web o decido utilizar el que crea necesario:** Poseo un servidor web de arquitectura AMP donde alojar todo el proyecto.
- 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 en su versión 2.4 donde se almacenarán todos los ficheros que compongan el sitio completo
- ✓ Un **servidor de bases de datos** como el MySQL 5.5 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 5.4.7 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.
- ✓ Aunque no es estrictamente necesario, también podríamos instalar un **panel de administración para la base de datos** como el phpMyAdmin 3.5 con lo que conseguiremos administrar las distintas bases de datos sin necesidad de programación, creando nuevas bases de datos, alterando las existentes o borrando las innecesarias.

Para poder probar el proyecto podemos instalar en nuestro equipo un servidor web de tipo AMP haciendo uso de un paquete de tipo XAMPP que integra todos los componentes anteriormente mencionados, pero hemos de tener en cuenta que al tratarse de un paquete estándar la seguridad es

muy reducida, por lo que tenemos que ajustar dicha seguridad modificando varios de sus parámetros como los datos de acceso, contraseñas, usuarios, etc.

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**.

Como indico en el punto anterior, este tipo de paquete adolece de una estricta seguridad, por lo que deberemos reforzarla, aunque también es cierto que incluye una herramienta especial para proteger fácilmente las partes más importantes.

8. Si utilizas el lenguaje PHP para programar la aplicación, ¿cuál será el tipo de datos que 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.

Por ejemplo:

```
<?
    if (!$HTTP_POST_VARS){
?>
<form action="envia_formulario.php" method=post>
    Nombre: <input type=text name="nombre" size=50 />
    <br />
    Email: <input type=text name=email size=25 />
    <br />
    <input type=submit value="Enviar" />
</form>
<?
    }else{
        //Estoy recibiendo el formulario, compongo el cuerpo
        $cuerpo = "Formulario enviado\n";
        $cuerpo .= "Nombre: " . $HTTP_POST_VARS["nombre"] . "\n";
        $cuerpo .= "Email: " . $HTTP_POST_VARS["email"] . "\n";
        mail("admin@tudominio.com","Formulario recibido",$cuerpo);
        echo " Gracias por rellenar el formulario. Se ha enviado correctamente.";
    }
?>
```