Desarrollo web en entorno servidor

UT1.1 – Arquitectura y programación web

1 – Aplicaciones web – Modelos de programación web – CGI – Lenguajes de script – Frameworks

IES Clara del Rey – Madrid

Aplicaciones web

Una aplicación web es un programa informático que utiliza un navegador web para interactuar con el usuario y para realizar tareas en Internet.

Se ejecutan a través de un servidor web que responde a las solicitudes realizadas por el usuario.

Siguen el modelo "cliente-servidor", en el que el navegador del usuario (el cliente) realiza peticiones al servidor web.

Normalmente se pueden ejecutar en cualquier dispositivo que disponga de un navegador, incluidos teléfonos móviles, portátiles, tablets y ordenadores.

Pueden estar limitadas en lo que se refiere al acceso a los recursos del dispositivo del usuario (el móvil, portátil, etc.)

Funcionamiento de las aplicaciones web

El funcionamiento general de una aplicación web es el siguiente:

- El cliente (navegador) envía una solicitud al servidor web en Internet utilizando el navegador web o la interfaz de usuario de la aplicación.
- En el servidor web se ejecuta la solicitud y se genera un resultado. La tarea solicitada puede ser muy variada:
  - Leer datos de una BD y devolverlos, o guardar información.
  - Generar informes. Enviar email o mensajes instantáneos.
  - Etc.
- El servidor web devuelve los datos o resultado del proceso al cliente.
- El navegador web muestra los datos al usuario.

Modelos de programación web

Hay tres modelos básicos de programación en web:

- Páginas web estáticas
- Aplicaciones web dinámicas programadas en cliente
- Aplicaciones web dinámicas programadas en servidor

Estos modelos no suelen estar completamente aislados.

Actualmente lo más habitual es que una aplicación web esté formada por una combinación de las tecnologías de los tres modelos.

Páginas web estáticas

Tecnologías utilizadas:

- HTML
- CSS

Pueden utilizar alguna otra, como XML (o dialectos como SVG o MathML).

No generan contenido de forma dinámica. Las páginas existen como ficheros. Si queremos generar más páginas, tenemos que escribir distintos ficheros HTML.

En muchas ocasiones pueden funcionar en local, con solo almacenar los ficheros necesarios (html, css, imágenes, etc.) en una carpeta.

Si queremos publicarlas en Internet es necesario un servidor web donde alojar los contenidos, como por ejemplo Apache, nginx, o IIS (Windows).

Aplicaciones web dinámicas en cliente

Tecnologías utilizadas:

- HTML
- CSS
- JavaScript / TypeScript

Pueden generar contenido dinámico, modificando la página.

Utilizan el DOM (Document Object Model) para cambiar la página. Por ejemplo, añadir filas en una tabla o elementos en una lista, o modificar estilos para que cambie el aspecto.

También pueden funcionar en local, simplemente almacenando los ficheros necesarios (html, css, imágenes, etc.) en local. Para publicarlas en Internet también hace falta un servidor web.

Aplicaciones web dinámicas en servidor

Tecnologías utilizadas:

- HTML
- CSS
- Un lenguaje de programación en servidor, como Java, C#, Node.js (JavaScript), Python, PHP, etc.

Generan contenido dinámico, generando páginas HTML diferentes en función de la URL o parámetros que recibe.

Siempre necesitan de un servidor para poder publicarlas. En este caso suele usarse un servidor de aplicaciones, que está un escalón por encima de los servidores web (veremos más sobre esto en despliegue).

Aplicaciones web dinámicas en servidor

Permiten generar distintas versiones de la misma página en función de múltiples factores, como, por ejemplo:

- La identidad del usuario, su situación o su perfil en el sistema
- La zona geográfica
- La fecha u hora

Para generar las páginas dinámicas, el servidor ejecuta un programa o parte de un programa que:

- Obtiene los datos necesarios de la petición del usuario
- Busca lo que necesite de bases de datos, de ficheros, etc.
- Genera una página web que devuelve al usuario.

Aplicaciones web estáticas vs dinámicas

¿Qué enfoque es mejor?

En general, la generación de páginas dinámicas es más potente y flexible que las páginas estáticas, pero también puede tener sus inconvenientes:

- No es necesario saber programar para desarrollar una página web estática. Sólo es necesario HTML y CSS.
- Requieren más recursos en el servidor, porque consumen CPU y memoria para generar las respuestas dinámicas.
- Los recursos necesarios son más complejos de administrar que para las páginas estáticas, lo que implica más coste en personal especializado.

Aplicaciones web actuales

Hoy día, una gran mayoría de las aplicaciones web son una mezcla de tecnología que combina:

- Elementos de páginas web estáticas, como HTML o CSS.
- Un lenguaje (TypeScript / JavaScript) para la programación en cliente.
- Un desarrollo con lenguaje de servidor para la programación en servidor, que puede hacer varias cosas:
  - Generar páginas web completas como respuesta a una petición.
  - Generar fragmentos de páginas web para actualizar, usando JavaScript, parte de la página.
  - Generar sólo datos (no HTML) que el navegador, usando JavaScript, utiliza para generar nuevo contenido en la página.

Aplicaciones web – Algunas ventajas

- Accesibles desde cualquier lugar con conexión a Internet.
- Multiplataforma: cualquier sistema con navegador puede ejecutarlas.
- Instalación y distribución simples: no es necesario instalar la aplicación en cada equipo. Los cambios se implementan en el servidor.
- Actualización permanente: los usuarios siempre acceden a la versión más actualizada de la aplicación.
- Menos recursos: los equipos cliente pueden ser muy ligeros. El servidor es el que debe soportar la mayor carga.
- Escalabilidad: en general, son fácilmente escalables.
- Más seguras: tanto en almacenamiento de datos como en exposición al hackeo. El recurso que proteger es fundamentalmente el servidor.

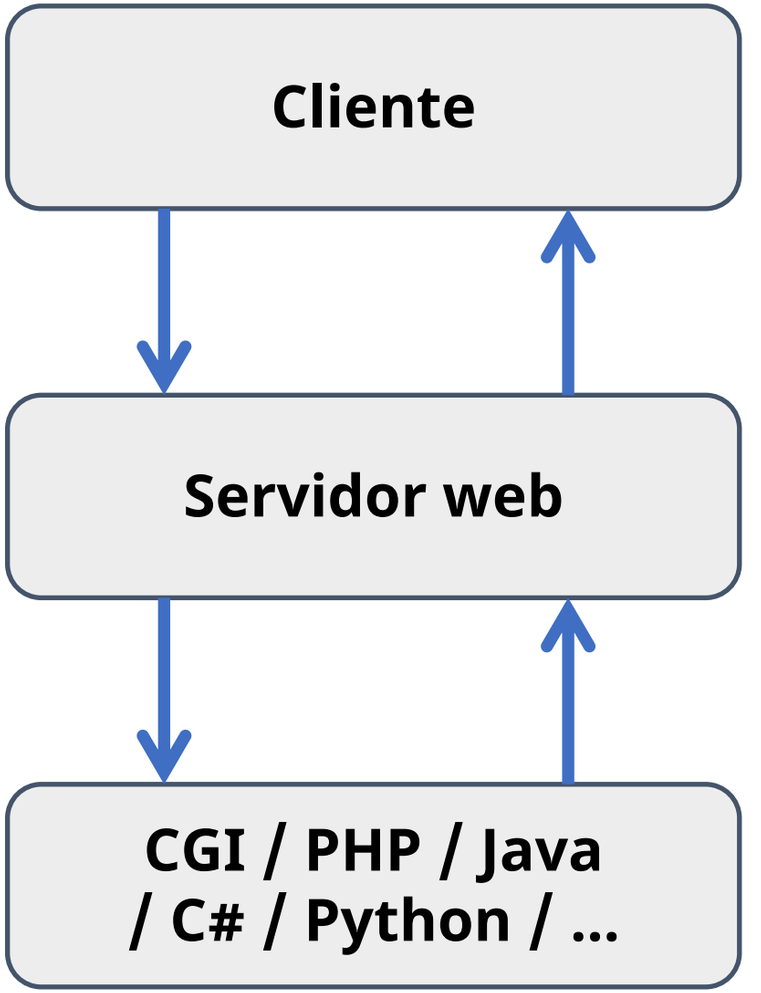
Aplicaciones web – Algunos inconvenientes

- Dependencia de Internet: sin conexión a Internet no funcionan.
- Dependencia del servidor: si el servidor falla, la aplicación falla.
- Rendimiento limitado: para ciertas tareas, el rendimiento de las aplicaciones web es menor que el de las aplicaciones de escritorio.
- Problemas de latencia: al funcionar con mensajes que viajan entre el cliente y el servidor, pueden afectarles las condiciones de la red.
- Funcionalidad limitada: cuando tenemos versión de escritorio y versión web de una aplicación, la versión web es más limitada (generalmente).
- Consumo de datos: al usar Internet puede implicar un gasto adicional en datos, en conexiones que no sean de tarifa plana.
- Expuestas a ataques DoS o DDoS, XSS, inyección SQL, etc.

Aplicaciones web – Funcionamiento

Un posible funcionamiento, a rasgos muy generales, de una página web dinámica en una aplicación web es:

1. El cliente lanza la petición al servidor
2. El servidor recibe la petición, la analiza, y decide cómo debe procesarla, qué componente debe responder.
3. El servidor pasa la petición al componente adecuado.
4. El componente procesa la petición, crea la página, y la devuelve al servidor.
5. El servidor devuelve la página al cliente.



¿Quién genera la página web dinámica?

En esencia, la parte de “servidor web” de una aplicación es un componente bastante “tonto”.

Se limita a recibir las peticiones y, en función de alguna característica de la petición, la redirige a otro componente, que es el que realmente se encarga de procesar esta petición.

Este componente, tras procesar la petición, genera una respuesta, y la devuelve al servidor web para que este a su vez la devuelva al cliente.

Estos componentes se pueden agrupar en tres grandes grupos:

- CGI – Common Gateway Interface
- Lenguajes de script (lenguajes de guiones)
- Frameworks de desarrollo web

CGI

Tecnología que permite al servidor interactuar con programas externos para generar el código de forma dinámica. El proceso básico es:

1. El navegador envía una petición HTTP al servidor.
2. El servidor **lanza un nuevo proceso** para ejecutar el programa CGI, pasando la información de la petición (formulario, parámetros, ...).
3. El programa CGI procesa la petición, genera una respuesta, como una página HTML o un archivo, y la suele escribir en su salida estándar.
4. El servidor recoge la respuesta generada por el programa CGI, y la transmite al navegador del cliente.

CGI no se usa demasiado en la actualidad, pero hay caso en los que se puede (PHP con Apache) y en otros es obligatorio (PHP con nginx, que no tiene soporte para módulos con este lenguaje).

CGI – Ventajas

- CGI es una especificación, no está vinculado o a un servidor específico. En la práctica se puede ejecutar casi cualquier programa en cualquier servidor, si se configura adecuadamente.
- Es un método relativamente sencillo de entender y de implementar.
- Como los CGI son independientes del servidor web, se pueden escribir en cualquier lenguaje, lo que da libertad a los desarrolladores.
- CGI permite mantener el servidor web y la lógica de la aplicación completamente separados, lo que facilita su mantenimiento.

CGI – Inconvenientes

- Bajo rendimiento. Implica crear un nuevo proceso para cada petición.
- Poco escalable. El modelo de "un proceso por solicitud", CGI no es adecuado para aplicaciones con un gran volumen de tráfico.
- CGI funciona sin estado, cada solicitud es independiente de las demás. Aplicaciones con sesión o estado necesitan mecanismos adicionales.
- Menor seguridad. CGI ejecuta programas en el servidor, una mala práctica que puede permitir la ejecución de comandos maliciosos.
- Mayor latencia. Con un proceso por solicitud, aumenta el tiempo de respuesta comparado con otros modelos. Una versión más moderna llamada FastCGI alivia algo este problema reutilizando procesos.

Lenguajes de Script en servidor

Estos lenguajes se ejecutan integrados en el servidor web (no programas externos) para generar el código dinámico. El proceso básico es:

1. El navegador envía una petición HTTP al servidor.
2. El servidor pasa la petición al procesador del lenguaje. Este se integra en el servidor como un "plug-in" independiente, pero dentro del mismo proceso (módulos).
3. El lenguaje de script genera el HTML necesario. El lenguaje se suele embeber dentro de un HTML base, con tags específicos.
4. El servidor recoge la respuesta generada por el script, y la transmite al navegador del cliente.

El lenguaje de script más utilizado en el mundo, con diferencia, es PHP. Vamos a centrarnos en él para analizar ventajas e inconvenientes.

Lenguajes de script (PHP) – Ventajas

- Hay una inmensa comunidad de desarrolladores, y mucha información y documentación.
- Desarrollo simple y rápido. Realizar una página PHP dinámica básica es cuestión de minutos. Además, al ser interpretado, permite probar rápidamente los cambios.
- Mantenimiento. En general el código es simple y fácil de mantener.
- Multiplataforma. Puede ejecutarse en casi cualquier sistema operativo.
- Bajo coste. Tanto el lenguaje como las herramientas necesarias para desarrollo y explotación pueden ser todas open source y gratuitas.
- Frente a programas CGI, tiene mejor rendimiento, porque se ejecuta en el proceso del servidor (salvo en nginx o IIS que es FastCGI).

Lenguajes de script (PHP) – Inconvenientes

- Menor rendimiento que lenguajes compilados. El código interpretado es más lento que el compilado (C#, Java).
- Usados como scripting "puro" (código PHP dentro de páginas HTML) no son los más aptos para desarrollos de gran envergadura. Este tipo de proyectos requieren una muy buena organización y estructuración del proyecto de desarrollo. Para estos casos, si se quiere seguir usando PHP como lenguaje, hay una serie de Frameworks que ayudan a estructurar el proyecto. Los dos más famosos son:
  - Symfony.
  - Laravel. Este framework usa varios componentes de Symfony.

Frameworks de desarrollo web

Un framework de desarrollo web es un conjunto de:

- Estructura estándar de aplicación, y directrices estandarización.
- Esqueletos de aplicación predefinidos.
- Bibliotecas de clases.
- Herramientas o entornos de trabajo.

Que permiten a los desarrolladores crear aplicaciones de forma más rápida, eficiente y consistente, en comparación con otras tecnologías como CGI o lenguajes de script "puros".

Frameworks de desarrollo web

Suelen seguir un patrón de arquitectura que organiza el código de una manera consistente y eficiente.

El patrón más habitual es MVC (Model-View-Controller):

- Model (modelo): Datos de la aplicación.
- View (vista): Se encarga de mostrar datos al usuario.
- Controller (controlador): Gestiona la interacción entre el usuario, la vista y el modelo.

Es normal, en cuanto la aplicación se complica un mínimo, que haya una cuarta "capa" denominada "de negocio" o "de servicios", que ayuda al controlador a gestionar la lógica de la aplicación.

Frameworks de desarrollo web

Además, los frameworks suelen incluir componentes muy útiles como:

- Routing: mapeo de las URLs a controladores específicos.
- ORM (Object-Relational Mapper): permite interactuar con bases de datos relacionales de manera más sencilla.
- Middleware: capas de procesamiento que manejan las solicitudes HTTP, la seguridad y la autenticación.
- Herramientas CLI (command line interface): permite crear elementos como controladores o modelos sin escribir código, y realizar pruebas o configurar procesos desde la línea de comandos o la terminal.

Frameworks de desarrollo – Ventajas

- Eficiencia y velocidad: tiene bibliotecas preconstruidas que permiten desarrollar aplicaciones más rápido.
- Buenas prácticas: al ser bastante dogmáticos, obligan a seguir patrones como MVC, generando un código más mantenible.
- Mantenimiento: los estándares hacen el código más mantenible.
- Seguridad: suelen tener activadas por defecto medidas para prevenir los ataques más habituales (inyecciones SQL, XSS, CSRF).
- Reutilización de código: al seguir estructura estándar, es más fácil reutilizar código en otros proyectos.
- Actualizaciones y comunidad: Muchos frameworks son mantenidos por grandes comunidades. Actualizaciones y mejoras constantes.

Frameworks de desarrollo – Inconvenientes

- Curva de aprendizaje: pueden ser más complejos de aprender y dominar que un simple lenguaje de script.
- Sobrecarga: pueden tener demasiadas funcionalidades, lo que quizá no los haga adecuados para desarrollos muy pequeños.
- Limitaciones: pueden imponer restricciones en lo que se refiere a estructura y organización de la aplicación.
- Dependencia: usar un framework tiene muchas ventajas, pero una vez utilizado, la aplicación tiene una fuerte dependencia de este, y de las posibles actualizaciones y cambios.

Frameworks de desarrollo – Ejemplos

PHP (aunque PHP se use mucho como script, existen frameworks):

- Laravel
- Symfony

JavaScript en servidor (Node.js)

- Express.js
- NextJS (añade a React la creación de páginas en servidor)

Python

- Django
- Flask

Frameworks de desarrollo – Ejemplos

Ruby

- Ruby on rails
- Sinatra

Java

- Spring (más usada su versión "simple" Spring Boot)
- JSF (Java Server Faces)

C# (.Net)

- ASP.Net Core
- Blazor

¿Qué usaremos durante el curso?

En este curso vamos a trabajar con dos lenguajes:

- PHP
  - Lo usaremos como lenguaje de script.
  - Se trata de una pequeña introducción para que tomemos contacto con este tipo de lenguajes.
  - No usaremos frameworks ni patrones (MVC, MVVM)
- Java
  - Utilizaremos Spring Boot, con patrón MVC, y principios SOLID
  - Utilizaremos técnicas avanzadas de acceso a datos (ORM)
  - Además de sitios web, haremos desarrollo servicios REST / JSON.