Introducción a la implantación de aplicaciones web

José Domingo Muñoz

IES Gonzalo Nazareno

Septiembre 2022

Introducción a las aplicaciones web

Página web us Aplicación web

- Una **Página Web** es un documento al que se puede acceder a través de un navegador web a través de internet o una intranet. Se suele decir que el contenido es estático (sólo cambia si el el desarrollador web cambia el contenido del documento).
- Una **Aplicación Web** nos ofrece páginas web que se generan de forma dinámica según unas circunstancias (por ejemplo si el usuario está logueado, según la información obtenida de una base de datos…). Nos ofrecen una determinada funcionalidad (blog, wiki, …). Son programas que desarrollamos utilizando lenguajes de programación web:
  - Se pueden ejecutar en el servidor: PHP, Python, Java, Ruby, Go, …
  - Se pueden ejecutar en el cliente: JavaScript, TypeScript,…

- **Específicas**: Son desarrolladas para gestionar un problema específico. En la actualidad en el desarrollo de aplicaciones web se utilizan **frameworks**: symfony (PHP), django (python), flask(python), RoR (Ruby on Rails),…
- **CMS**: Sistema de gestión de contenido. Son aplicaciones desarrolladas por un tercero que nos ofrecen una funcionalidad general. Ejemplos: blogs, portales, wiki, comercio electrónico, educación a distancia, gestión de proyectos,… Las más conocidas: wordpress, joomla, mediawiki, prestashop, moodle, redmine, …

Protocolo HTTP / HTTPS

Para acceder a las Páginas Web o Aplicaciones Web necesitamos usar el protocolo HTTP/HTTPS:

- Un **cliente** (por ejemplo, **un navegador web**) realiza una **petición HTTP** solicitando un recurso a un **servidor web**.
- El **servidor web** recibe la petición y genera una **respuesta HTTP** devolviendo si es posible el recurso.

Nota: Se profundizará en el protocolo HTTP en el módulo de Servicios de Red e Internet.

Acceso a páginas web estáticas

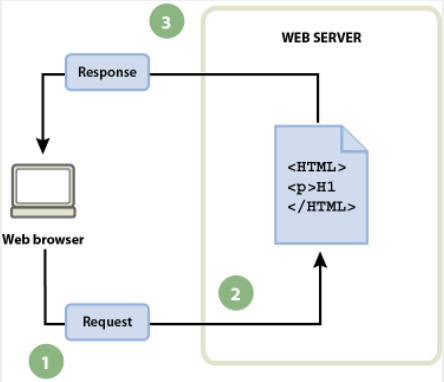

Figura 1: Procesamiento de una página web estática

1. El cliente realiza una **petición HTTP**.
2. El servidor web recibe la petición y busca el recurso que se ha solicitado.
3. El servidor web devuelve una **respuesta HTTP**.

Acceso a aplicación web dinámicas

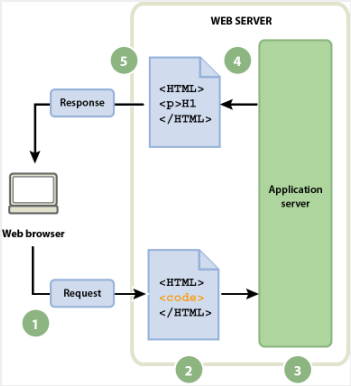

Figura 2: Procesamiento de una aplicación web dinámica

1. El cliente realiza una **petición HTTP**.
2. Si el recurso solicitado es un programa escrito en algún lenguaje de programación web pasamos al paso 3. Si se solicita un recurso estático (hoja de estilo, pdf, imagen…) se devolverá como en el caso anterior.
3. El **servidor de aplicaciones** es el programa responsable de ejecutar el programa.
4. La salida de dicho programa suele ser una página web (por eso la llamamos **dinámica**).
5. El servidor web devuelve una **respuesta HTTP** con la página generada.

Acceso a aplicación web dinámicas con acceso a base de datos

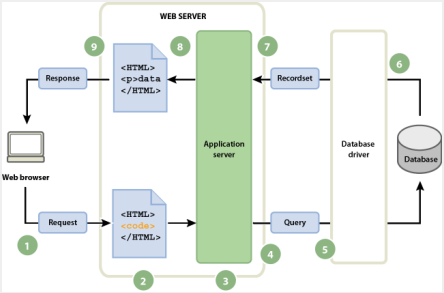

Figura 3: Procesamiento de una aplicación web dinámica con acceso a una BD

- Los tres primeros pasos iguales…

4. El programa puede realizar una consulta a una base de datos.
5. La consulta la recibe el gestor de base de datos.
6. La base de datos devuelve el resultado de la consulta.
7. Que utiliza el programa para construir la página dinámica.

- Los dos últimos pasos iguales…

Ventajas y desventajas de las páginas web estática

- **No** ofrecen ninguna funcionalidad (o una funcionalidad limitada, solo la que puede ofrecer javascript porque se ejecuta en el cliente).
- El desarrollo es sencillo: escritas en HTML y el diseño usando hojas de estilo.
- El mantenimiento puede ser complicado, por ejemplo cuando queremos cambiar la estructura de muchas páginas de nuestro sitio web.
- El desarrollo y mantenimiento se puede hace más sencillo, usando **Generadores de Páginas Web Estáticas**. Son programas que a partir del contenido de nuestra página y de plantillas generan de forma automática el código HTML para subirlo al servidor web.
- La ventaja fundamental es que el acceso a estás páginas es muy rápida.

Ventajas y desventajas de las aplicaciones web dinámicas

- Principal ventaja: Ofrecen muchas funcionalidades. Se está pasando de usar aplicaciones de escritorio a aplicaciones web, ya que las funcionalidades ofrecidas cada vez nos permiten realizar más tareas.
- Si necesitamos solucionar un problema concreto necesitamos desarrollar una aplicación web desde 0, pero si necesitamos una funcionalidad general (blog, wiki, comercio electrónico,...) podemos usar CMS ya desarrollados.
- Suele ser sencillo desarrollar el diseño de los CMS (con distintos temas) y sólo nos tenemos que preocupar de desarrollar el contenido que se suele guardar en una base de datos.
- Si estamos usando CMS es necesario actualizar la aplicación cada cierto tiempo para no tener problemas de seguridad.
- La desventaja fundamental es que el acceso es más lento: la ejecución del código es lenta y el acceso a la base de datos aún más.

Implantación o despliegue de aplicaciones Web

La **Implantación** o **Despliegue** de aplicaciones web es el conjunto de tareas que hay que realizar para poner en un servidor web (**Entorno de Producción**) por parte de los **Administradores de Sistemas**, una **versión** de una aplicación web que ha desarrollador los **Desarrolladores** (**Entorno de Desarrollo**) después de probar que funciona de forma correcta (**Entorno de Pruebas**).

Tenemos que tener en cuenta que el proceso de implantar una aplicación web es crítico, ya que normalmente se le da soporte a muchos clientes y el fallo de la aplicación en el entorno de producción puede causar una perdida de dinero muy importante para la empresa.

Equipos de trabajo

Como vemos en la definición anterior tenemos al menos dos equipos de trabajo:

- **Administradores de sistemas (ops)**: Son los encargados de que el entorno de producción funcione de manera adecuada. Conocen el hardware y los sistemas operativos instaladores en los servidores. Reciben la aplicación del equipo de desarrollo y son los responsables de hacerla funcionar en los servidores.
- **Desarrolladores (devs)**: Son los encargados de crear y añadir nuevas funcionalidades a la aplicación. No acceden a los servidores, suelen desarrollar en sus propios equipos (entorno de desarrollo), también suelen probar la aplicación antes del despliegue. Su objetivo principal es ofrecer nuevas funcionalidades a los clientes desarrollando nuevas versiones de la aplicación que se tendrán que desplegar.

Equipos de trabajo

Pueden existir otros equipos:

- **Equipo de calidad**: Se aseguran que las nueva versiones de la aplicación funciona de forma correcta (entorno de pruebas).
- **Equipo de diseño**.
- **Equipo de soporte**.
- ...

Entornos de trabajo

- **Desarrollo**: entorno donde el equipo de desarrollo crea el software, normalmente es su propio PC, que ellos suelen denominar “local”. Se suelen realizar muchos cambios al día, del orden de decenas o incluso cientos. También es común ver entornos de desarrollo compartidos entre miembros del equipo de desarrollo.
- **Pruebas**: Se configura prácticamente igual al de producción y sirve para validar cambios e intentar reproducir los errores de producción.
- **Producción**: es el entorno donde se ejecutan las aplicaciones y pueden usarlas los clientes finales. Normalmente este entorno suele ser, al menos, un servidor dedicado, si no un cluster, y lo normal es que tenga un sistema operativo diferente al entorno de desarrollo e incluso una arquitectura diferente. Debe permanecer estable. Suele estar gestionado por el equipo de Sistemas.

¿Cómo se realiza el despliegue de aplicaciones web?

Podemos tener distintos flujos de trabajo o metodologías:

- **Tradicional**:
  - No hay equipos de trabajo diferenciados, ni entornos separados.
  - Los propios desarrolladores se encargan de desplegar los cambios en los servidores de producción.
  - Cada vez que se aplica un cambio es posible que haya nerviosismo y aparezcan nuevos cambios precipitados para corregir algún bug.
  - No se suele actualizar el servidor de producción para que no se produzcan errores en la aplicación.
  - **Si funciona no lo toques!!!**
  - **Pues en mi local funciona**

- **Burocrática**:
  - En este flujo de trabajo los equipos y los entornos están diferenciados.
  - El problema que pueden tener es que muy poco flexible y los plazos para el despliegue están prefijados y se necesita la autorización de alguna persona para desplegar un posible cambio.
  - Por ejemplo: está fijado que una vez a la semana, siempre el mismo día y a la misma hora, se realiza el despliegue a producción. Siempre se hace el despliegue por una persona autorizada y se exige que el software haya pasado por los entornos de desarrollo y pruebas. Por lo tanto, los cambios tardan en llegar al usuario final cerca de una semana.
  - Ejemplo: Procedimiento Gestión de Despliegues - Junta de Andalucía

¿Es fácil realizar el despliegue de aplicaciones web?

Poner un proyecto de software en marcha comprende más cosas que desarrollar código. El software es algo vivo, normalmente una vez que se pone online o se entrega al cliente se sigue desarrollando y generando nuevas versiones. En la mayor parte de las empresas, la puesta en producción de una nueva versión es un momento crítico por las incertidumbres que genera.

- El objetivo de los desarrolladores es **añadir nuevas funcionales** al programa para darle más valor al producto que se está ofreciendo al cliente.
- El objetivo de los "de sistemas" es que los servidores ofrezcan de manera óptima la aplicación, están dando también valor al pproducto poruqe la experiencia del cliente en el uso del programa es bueno.

En muchas ocasiones estos objetivos no son compartidos por los dos equipos y se pueden producir "malos rollos".

¿Es fácil realizar el despliegue de aplicaciones web?

Por ejemplo:

- En el flujo de trabajo tradicional: Como no hay equipo de sistemas especializados, no hay separación de entornos, el despliegue puede producir que la experiencia del cliente no sea positiva al usar el programa (se pueden producir fallos, el rendimiento no es adecuado, …).
- En el flujo de trabajo burocrático: Al contrario: el equipo de desarrollo tiene nuevas funcionalidades, pero la rigidez del proceso conlleva que el cambio lo recibirá mucho más tarde el cliente final.

En ambos casos no estamos aportando valor a nuestro producto. Necesitamos otra metodología que nos permita trasladar valor a los clientes de forma frecuente, sencilla y segura.

Despliegue ágil de aplicaciones web

- Los ciclos de desarrollo de software actuales son ágiles y rápidos, de forma que continuamente se están arreglando errores, programando nuevas características y desplegándolas en producción.
- Necesitamos una forma de despliegue que permita de una manera rápida, sencilla y segura pasar las nuevas versiones de software a producción.
- A esta manera de despliegue se le suele llamar **ágil**.
- Para ello es necesario cambiar la mentalidad de los equipos de desarrollo y sistemas (tenemos objetivos comunes: **ofrecer valor al producto**).
- **DevOps** es un acrónimo inglés de development (desarrollo) y operations (operaciones o sistemas), que se refiere a una metodología de trabajo que se centra en la comunicación, colaboración e integración entre desarrolladores de software y los profesionales de operaciones en las tecnologías de la información (IT).
- Qué demonios es eso de Devops (y porque debería interesarme)

Las metodologías y herramientas que se suelen usar en este modelo de despliegue son:

- Entornos de producción, pruebas y desarrollo comparables: Es decir, hacer que se parezcan lo máximo posible los distintos entornos, para ello podemos utilizar:
  - Herramientas de construcción de infraestructura, por ejemplo **vagrant**. Podemos conseguir que todos los desarrolladores trabajen en el mismo escenario, que será parecido en el que luego se va a desplegar la aplicación.
  - Entornos virtuales de prueba: Normalmente los distintos entornos tienen distintos sistemas operativos, y por lo tanto distintas veriones de las librerías usadas en el desarrollo de la aplicación. El uso de entornos virtuales solucionar los problemas con las versiones de los interpretes y librerías. Por ejemplo: venv para Python,...

- Infraestructura como código: Automátización de la configuración. Aunque actualmente se automatiza todas las partes posible del ciclo de desarrollo de software, no se hace lo mismo con la administración de sistemas. El concepto de infraestructura como código se refiere a intentar gestionar la infraestructura como si de un desarrollo software se tratase. Por ejemplo con herramientas como Puppet, chef, ansible, podemos automatización de los procesos de instalación y configuración.

- Uso de Sistema de control de versiones. Tanto para el departamento de desarrollo como para el de sistema es imprescindible el uso de un sistema de control de versiones. En Sistemas, al gestionar la infraestructura como código es necesario gestionar los distintos ficheros de configuración con un sistema de control de versiones. Ejemplo: git,...
- Herramientas de integración y despliegue continúo: Herramientas que nos permite automatizar el ciclo de vida de desarrollo de software: compilación, testeo, control de código, documentación, despliegue, ... Ejemplo: Jenkins,...

¿Cuál puede ser el entorno de producción?

- Tradicional
  - Servidor dedicado: Servidor físico o virtual. Al que tenemos acceso y control total. Características de hardware fijas.
  - Hosting compartido: Servidor físico o virtual compartido entre varios usuarios. No se suele tener acceso a realizar instalaciones específicas. Características hardware fijas.

¿Cuál puede ser el entorno de producción?

- Moderno
  - Cloud Computing:
    - IaaS: Infraestructura como servicio en la nube. Se proporciona principalmente capacidad de cómputo, redes y diversos modos de almacenamiento. Elasticidad y pago por uso. AWS, Azure, GCE, OpenStack
    - PaaS: Plataforma de desarrollo web en la nube. Se proporciona toda la plataforma de desarrollo y despliegue de una aplicación al desarrollador. Elasticidad y pago por uso. Openshift, Azure, Heroku, Google App Engine, …
  - Contenedores: Virtualización ligera. Parecidas a MV, pero mucho más ligeras y transportables.