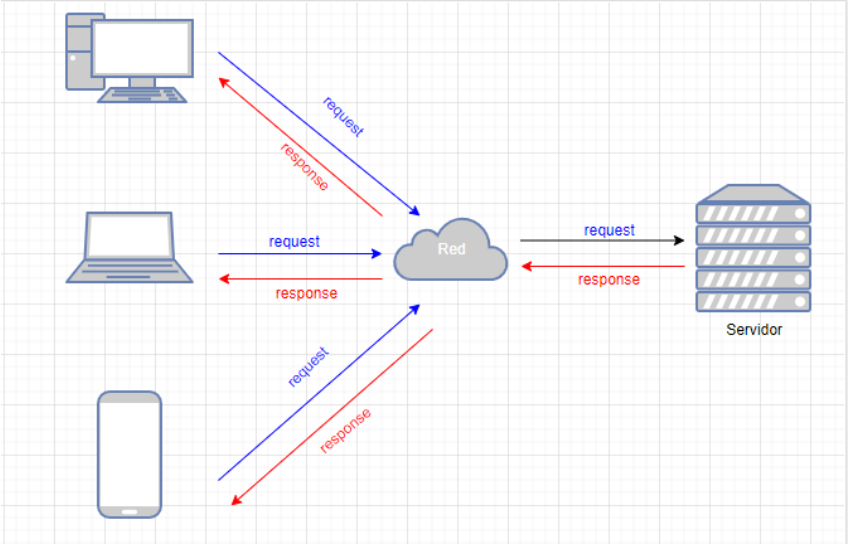
Despliegue de Aplicaciones Web

Tema 1: Arquitecturas Web

1. Introducción

1.1. Aspectos generales

Teniendo en cuenta la infraestructura hardware, el modelo de desarrollo web va ligado a una arquitectura cliente-servidor. En esta arquitectura hay dos actores principales, cliente y servidor, de tal forma que el servidor se está ejecutando de continúo esperando a que los clientes realicen una solicitud, denominada request, de los servicios que ofrece. Así un navegador se conectará al servidor para solicitar una página web, información a un servicio web o servicio REST, etc. La respuesta del servidor se denomina response.



El servidor será un software multitarea que pueda atender a varios clientes a la vez.

1.2. Modelo en capas

Desde un punto de vista del desarrollo, existen diversos modelos de arquitecturas. Estos modelos siguen basándose en la arquitectura cliente-servidor, pero se especifican cosas más concretas como que software se utilizará en cada actor o cómo interactúan las diferentes tecnologías o aplicaciones.

Las aplicaciones web se modelan mediante modelos de capas, donde una capa representa un elemento que procesa o trata información. Las capas son una forma de separar responsabilidades y administrar dependencias. Cada capa tiene una responsabilidad específica.

No hay que confundir capas con niveles, ya que las capas son a nivel lógico mientras que los niveles hacen referencia a la organización y división física de los distintos componentes o elementos de diseño que conforman el sistema.

Por ejemplo, una aplicación de tres capas no tiene por qué tener tres niveles, pueden estar las tres capas desplegadas en el mismo nivel. Es decir, las tres capas funcionales podrían estar ejecutándose en el mismo nivel físico, dentro de la misma máquina. También podrían estar en distintos niveles, pero no tiene por qué coincidir el número de capas lógicas con el número de niveles físicos.

Modelo de dos capas

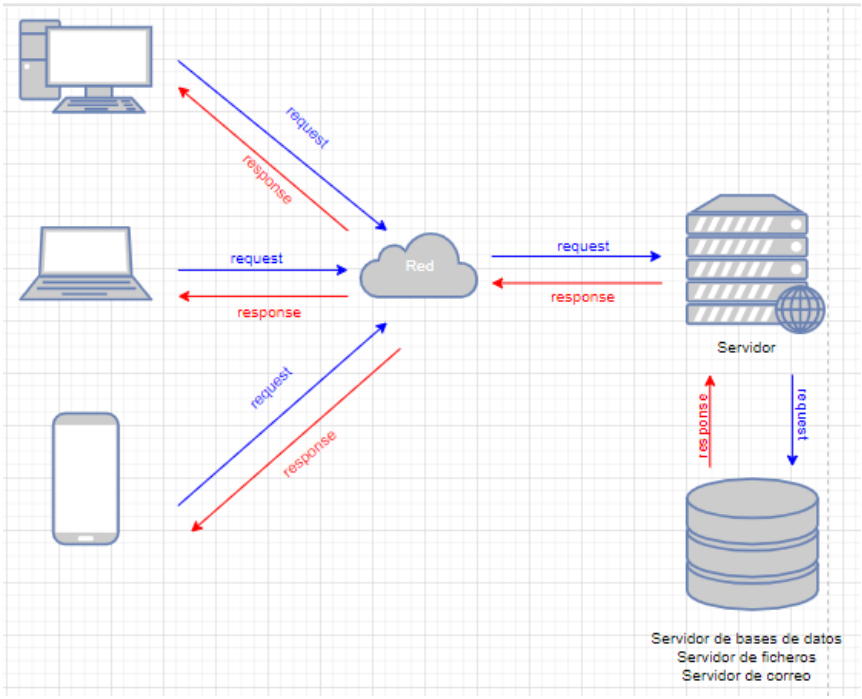
El modelo de dos capas es el tradicional, consecuencia de la arquitectura cliente-servidor. La información atravesará dos capas entre la interfaz y la gestión de los datos. Una arquitectura en 2 capas distribuye la aplicación en dos componentes lógicos, donde normalmente es el cliente quien mantiene la lógica de presentación, de negocio, y de acceso a los datos, y el servidor únicamente gestiona los datos.

Este modelo presenta las siguientes desventajas:

- Es difícilmente escalable.
- El número de conexiones es reducido.
- Se produce una carga alta de la red.
- La flexibilidad es restringida.
- La funcionalidad es limitada.

Modelo de N capas

El modelo de N capas más común es el de tres, aunque para aplicaciones complejas puede haber más. Una aplicación tradicional de tres capas tiene una de presentación, una intermedia y una de datos. El siguiente es un sistema de tres capas organizado en tres niveles distintos:



Las tres capas del modelo homónimo son:

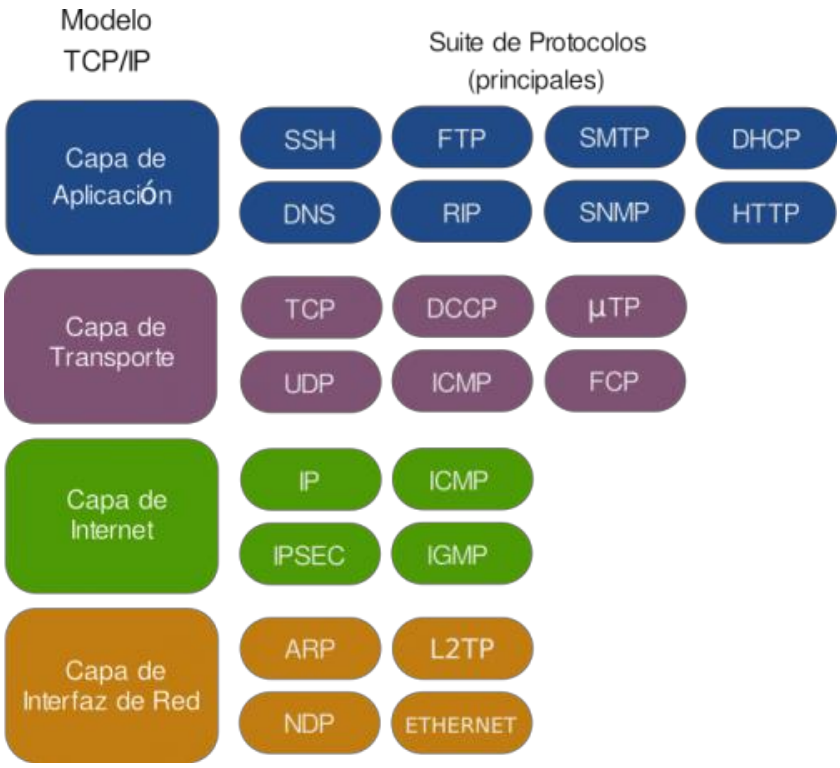
- Capa de presentación: capa que se encarga de la navegabilidad, validación de los datos de entrada, formateo de los datos de salida, del aspecto de la web, etc. Se comunicará únicamente con la capa de negocio.
- Capa de negocio: capa encargada de recibir las peticiones del usuario y de enviarle las respuestas a las mismas, además de verificar que las reglas (lógica de negocio) establecidas se cumplen. Se comunica con las capas de presentación y de datos.
- Capa de datos: capa en la que residen los datos y que se encarga de acceder a los mismos. Estará formada por determinados gestores de datos que almacenen, estructuren y recuperen los datos solicitados por la capa de negocio.

2. Servicios en red y Protocolos

Un servicio en red es un programa que funciona en el nivel de aplicación y que puede proporcionar almacenamiento y manipulación de datos, acceso remoto a sistemas, monitorización de equipos o cualquier otra funcionalidad, siempre marcada por alguno de los protocolos establecidos. Es muy habitual que use una arquitectura cliente servidor en la que el servicio es proporcionado por uno o más servidores y que será accesible a través de la red por aplicaciones cliente que se encuentren funcionando en otros dispositivos.

Para llevar a cabo una comunicación completa entre equipos a través de una red y que todos esos servicios de red puedan ser accesibles, entran en juego un amplio número de protocolos ya sea siguiendo el modelo OSI o el modelo TCP/IP. En la capa de aplicación es donde se encuentran los protocolos de la web, los que usan navegadores y servidores (web y aplicaciones) para comunicarse. Los servicios más fácilmente reconocibles en Internet son el servidor Web (protocolo HTTP) y el de correo electrónico (protocolo SMTP), aunque existen otros muchos como pueden serlo FTP, SSH, etc.

2.1. Protocolos de red



HTTP

HTTP (HyperText Transfer Protocol) es un protocolo para la transferencia de páginas web. Cuando un usuario quiere acceder a una página web a través de un navegador, éste realiza una petición(request) al servidor. El servidor contestará con una respuesta (response) HTTP y, de tenerla, la página web solicitada.

HTTP no tiene estado, lo que supone que un servidor web no almacene ninguna información sobre los clientes que se conecten a él, por lo tanto, cada par petición/respuesta constituye una conexión única y aislada.

SSL/TLS

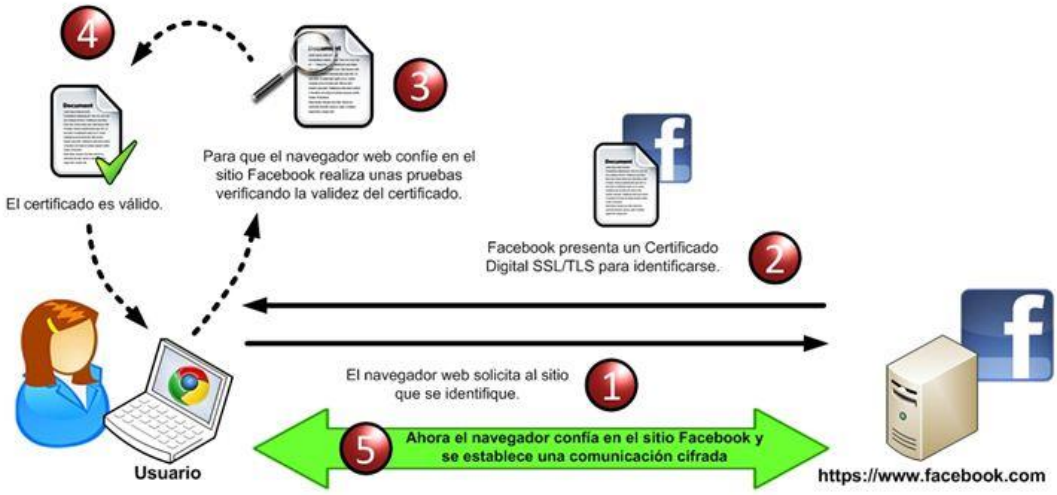
SSL (Secure Sockets Layer) y TSL (Transport Layer Security) son protocolos para cifrar las comunicaciones en Internet para garantizar la seguridad digital. No se trata de protocolos que se utilicen indistintamente sino que más bien TLS es el sucesor de SSL quedando este último en desuso.

TLS se ubica en la capa de transporte del modelo TCP/IP. No solo se utiliza en las comunicaciones con páginas web, por ejemplo, es de gran utilidad en redes VPN.

HTTPS

HTTPS (HTTP Secure) es un protocolo de comunicación en Internet que protege la integridad y la confidencialidad de los datos. HTTPS se basa en la comunicación de HTTP cifrando el contenido con TLS. TLS aporta cifrado e integridad de los datos para mantenerlos a salvo de otros usuarios y que no sean modificados o dañados durante las transferencias, y autenticación para asegurar que los usuarios se comunican con el sitio web previsto.

Cuando en la URL de un sitio web aparece https, quiere decir que dicho sitio está protegido por un certificado TLS(aún llamados certificados SSL). Los sitios web que instalan y configuran un certificado TLS pueden usar el protocolo HTTPS para establecer una conexión segura con el servidor.



Telnet

Telnet (Teletype network) permite la comunicación interactiva basada en texto (sin cifrado alguno) entre dos máquinas. Proporciona acceso remoto a una máquina (donde se ejecuta el lado servidor) permitiendo ejecutar comandos en ella y visualizar los resultados. En el lado cliente, el usuario debía utilizar una aplicación cliente como putty o shell. Actualmente su uso está prácticamente limitado a redes internas aisladas de la red del exterior. En los demás casos, siempre se utiliza el protocolo SSH.

SSH

SSH (Secure SHell) es un protocolo, al igual que Telnet, que proporciona comunicación entre dos máquinas remotas basada en texto, solo que en este caso existe cifrado tanto para la autenticación como para la comunicación en sí. Su uso más habitual está en la gestión remota de máquinas Unix/Linux.

SCP

SCP (Secure Copy Protocol), basado en el protocolo SSH, permite transferir ficheros entre dos equipos, tanto del equipo local al remoto como en el sentido contrario, de forma segura.

Los desarrolladores full-stack utilizan con frecuencia SCP para sus funciones de autenticación y cifrado sin requerir servicios de alojamiento de terceros como Github.

FTP

El protocolo FTP (File Transfer Protocol) es un protocolo para la transferencia de ficheros entre una máquina cliente y otra que aloja al servidor de ficheros. Se trata de un protocolo que ha evolucionado mucho desde su creación y existe una gran cantidad de software en torno a él, pudiéndose, por ejemplo, realizarse transferencias de archivos entre máquinas con distintos sistemas operativos, aunque FTP no cifra ni la autenticación con usuario/clave ni tampoco los datos transferidos.

Por ejemplo, al descargar un PDF pulsando un enlace de una página web, dicha descarga se hace a través del navegador siendo transparente para el usuario final que el archivo estará alojado en una máquina distinta a la que nos muestra la página web. Es decir, por un lado, estará el servidor web y por otro el servidor FTP.

En el ejemplo anterior la descarga se hace a través del navegador, pero también existen programas independientes que actúan como cliente FTP como Filezilla.

El problema de la confidencialidad en la autenticación y en la transferencia de datos se solucionó añadiendo una capa de seguridad SSL/TLS al propio protocolo FTP. Los siguientes protocolos también se conocen como FTP over TLS/SSL y están basados en FTP:

- FTPS, o FTPS Implícito, es la forma antigua de proporcionar seguridad y privacidad al protocolo FTP.
- FTPES, o FTPS Explícito, es el protocolo que se utiliza actualmente cuando se quiere activar la seguridad en FTP.

SFTP

SFTP (SSH FTP) no tiene nada que ver con FTP aunque lo lleve en su nombre, no se trata de la versión segura de FTP ya que está totalmente basado en SSH. Por lo tanto, no es una mejora de FTP si no que se trata de un protocolo diferente. Este protocolo permite autenticaciones y realizar transferencias de ficheros entre equipos como si fuera un servidor FTPES, pero utilizando criptografía del protocolo SSH que se encuentre instalado en el servidor de archivos.

2.2. Identificación de equipos

Para poder acceder a cualquier servicio que ofrezca un servidor, lo primero es tener acceso desde el cliente al servidor y, para ello, éste deberá tener una IP conocida y accesible:

- Si se trata de una red local, la IP será privada y perteneciente a alguno de los rangos reservados para ello.
- Si el servicio va a estar disponible en Internet, habrá que conseguir una dirección pública. Al contratar internet, el router proporcionado dispondrá de una IP pública para poder navegar por Internet y otra privada para la red de área local, pero esa IP pública lo normal es que sea dinámica y que cambie con el tiempo.

Los servicios de red necesitan direcciones IP públicas fijas, por lo tanto, habrá que contratarlas con un proveedor de Internet o ubicar el servicio en la nube.

EJERCICIO 1

- Busca la manera de conocer la IP pública de tu ordenador.
- Comprueba que la IP pública de tu equipo es diferente a la IP privada y que ésta última pertenece a un rango reservado para ello. ¿De qué clase es tu IP privada?

2.3. Identificación de servidores – DNS

En una red, aunque los equipos se identifiquen entre sí a través de una dirección IP, el usuario utiliza nombres (direcciones web) para indicar los servicios a los que quiere acceder. De la asociación entre esos nombres y sus respectivas direcciones IP se encarga el sistema de nombre de dominio, DNS (Domain Name System). Según este sistema, una serie de servidores organizados de manera jerárquica resuelven los nombres de cualquier equipo conectado a Internet. Estos servidores, al igual que los sistemas operativos, mantienen una cache con las últimas consultas realizadas.

El sistema DNS tiene tres partes bien diferenciadas:

- Cliente DNS: estará instalado en el cliente (los diferentes sistemas operativos suelen traerlo de serie) y realiza peticiones de resolución de nombres a los servidores DNS.
- Servidor DNS: encargado de contestan las peticiones, resolviendo los nombres mediante un sistema estructurado en árbol. Cuando en la configuración de una conexión se indica una IP DNS, se trata de la IP de un servidor DNS.
- Zonas de autoridad: uno o más servidores que se encargan de resolver un conjunto de dominios determinado (.es, .org, etc.).

2.4. Comandos

Se puede hacer uso de múltiples comandos a la hora de comprobar conectividad y funcionamiento de un servidor, algunos son:

- En Windows:
  - ping.
  - ipconfig.
  - nslookup.
  - tracert.
- En sistemas Linux:
  - ping.
  - ifconfig.
  - nslookup.
  - traceroute.

3. Despliegue en Internet

3.1. Escalabilidad

Capacidad de adaptación y respuesta de un sistema a medida que aumenta el número de usuarios del mismo, que puede variar significativamente dependiendo del momento. El diseño del sistema influye en el rendimiento, por lo que la escalabilidad va fuertemente ligada al mismo. En una aplicación bien diseñada, la escalabilidad no supone un problema.

La escalabilidad de un sistema puede ser vertical u horizontal.

Vertical

El escalado vertical se consigue aumentando los recursos del servidor, principalmente en cuanto a capacidad de procesamiento, memoria y almacenamiento, aunque también se puede llegar al extremo de cambiar a un servidor nuevo.

Horizontal

Implica agregar más nodos a un sistema y dividir la carga de trabajo entre todos. Esto conlleva la aparición de elementos nuevos, los que se van a encargar de distribuir esa carga de trabajo, lo que se llama balancear la carga. El balanceador de carga puede ser:

- Por software: Este tipo de balanceador examina el paquete http e identifica la sesión del usuario, guardando registro de cuál de las máquinas se está encargado de servir a dicha sesión. Esto permite almacenar información relativa a la sesión del usuario ya que todas las peticiones irán al mismo servidor.
- Por hardware: Un dispositivo, respondiendo únicamente a algoritmos de reparto de carga, redirecciona una petición http del usuario a la máquina que convenga que se haga cargo. Son mucho más rápidos que los anteriores, dado que se basan en conmutación de circuitos y no examinan ni interpretan el paquete http, pero el diseño de la aplicación se ve condicionado ya que la información de la sesión del usuario deberá ser almacenada bien en cookies o bien en base de datos.
- Balanceador hardware http: A medio camino entre los dos anteriores, es un dispositivo hardware pero que examina el paquete http y mantiene la relación usuario-servidor. Mucho más rápido que el balance por software, pero menos que el hardware, hoy en día es una solución bastante aceptada.
- Cluster sobre servidores: El concepto de clustering introduce la capacidad de unir varios servidores para que trabajen en un entorno en paralelo como si fueran uno solo. Dependiendo de su uso se podría considerar escalabilidad vertical u horizontal. El clustering no presenta dependencias a nivel de hardware (no todos los equipos necesitan el mismo hardware) ni a nivel de software (no necesitan el mismo sistema operativo).

Por norma general, el escalado vertical es más que suficiente para la mayoría de las aplicaciones, pero cuando se espera un crecimiento elevado de los usuarios y, por tanto, una cantidad alta de solicitudes en un espacio corto de tiempo es posible que sea necesaria la planificación de una estrategia de escalado horizontal.

3.2. Tipos de servidores

Actualmente, es bastante habitual recurrir a una empresa de hosting que ponen sus instalaciones a disposición de los clientes. Los diferentes tipos existentes son:

- Servidor dedicado
- Servidor compartido
- Servidor virtual (VPS - Virtual Private Server)
- Nube (Cloud)

Servidores dedicados

Están pensados para servir a usuarios independientes, creando una configuración específica para cada caso, tanto en software como en hardware. El usuario dispondrá de una máquina física que controla totalmente.

Servidores compartidos

En el alojamiento Web compartido, existe un servidor con unos recursos limitados. De estos recursos, algunos se comparten entre todos los sitios alojados en él, y otros estarán en exclusiva para cada uno, como el espacio en disco, del cual se dispondrá de una porción de total del disco o discos duros del servidor. Por lo tanto, el desarrollador no tendrá control total sobre el servidor.

Servidores privados virtuales (VPS)

Muy similares a los servidores dedicados desde el punto de vista de los clientes. El usuario dispone de una máquina virtual que, esa sí, controla totalmente, pero ésta se ejecuta en un servidor físico que comparte con otros usuarios.

Alojamiento en la nube o cloud computing

La principal ventaja del alojamiento en la nube es su escalabilidad. El usuario puede aumentar los recursos contratados siempre que quiera y pagar únicamente por lo que utiliza. Los recursos, en la nube, suele ser virtuales, aunque es posible que sean máquinas físicas. La diferencia básica con el resto de tipos de alojamientos recae en la elasticidad de los recursos.

Los tipos de servicios en la nube son:

- IaaS: infraestructura como servicio. En lugar de contratar infraestructura física, los usuarios pueden comprar solo los recursos que necesitan.
- PaaS: plataforma como servicio. Proporciona a los usuarios un entorno en el que pueden desarrollar, gestionar y distribuir aplicaciones. Además del almacenamiento y otros recursos informáticos, los usuarios pueden utilizar herramientas prediseñadas para desarrollar, personalizar y probar.
- SaaS: software como servicio. Los usuarios no instalan aplicaciones en sus dispositivos locales, sino que las aplicaciones residen en la nube a la que se accede a través de la red. Mediante la aplicación, los usuarios pueden almacenar y analizar datos, además de colaborar en proyectos. Por ejemplo, Google Docs.

EJERCICIO 2

- Busca tres ventajas y tres inconvenientes de cada tipo de servidor.
- Apoyándote en la respuesta dada al punto anterior, busca un caso para cada tipo de servidor en el que sería el tipo más adecuado.
- Busca un ejemplo de proveedor de cada tipo de servidor.

4. Servicios web y Aplicaciones web

Una aplicación web es una aplicación de software que un usuario ejecuta en el navegador web. Tiene las siguientes características.

- Tiene una interfaz de usuario
- Se ejecuta en el cliente - entorno del servidor
- El cliente lo ejecuta con la ayuda de un navegador web
- El servidor procesa los datos según la solicitud del cliente y proporciona una respuesta

Por su parte, un servicio web es una interfaz de programa de aplicación (API) que se ejecuta en el servidor, que proporciona datos al cliente a través de http a través de un sistema de mensajería estandarizado. (XML, JSON, etc.). Se trata de una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

4.1. Servicios web

No es raro que una aplicación comparta información con otras. Por ejemplo, para el tratamiento de envíos internacionales de paquetes, Correos necesita almacenar cierta información de cada uno de ellos (datos del destinatario, del remitente, precios de aduanas, etc). Esa información, la proporcionará a la parte cliente cuando esta lo requiera, cuando un trabajador de correos quiera consultar esos datos. Además, cuando un paquete se envíe, esa información deberá pasarse a la aplicación del país receptor del mismo. Esta transacción no necesita de una interfaz visible para el usuario, pero el lado servidor proporcionará un servicio web para que pueda realizar la petición de datos dicha aplicación del país receptor.

Siguiendo el ejemplo, también se puede dar que, estando el sistema en marcha se quiera ampliar con un nuevo cliente que ofrecerá esa información al usuario final a través de la web. Esa nueva aplicación a desarrollar solicitará la información al mismo servicio web anterior.

Como se ha indicado con anterioridad, un servicio web es un método que permite que dos equipos intercambien información a través de una red. Al utilizar servicios web, el servidor puede ofrecer un punto de acceso a la información que quiere compartir. De esta forma controla y facilita el acceso a la misma por parte de otras aplicaciones.

EJERCICIO 3

- Lee el artículo del siguiente enlace:

https://www.redhat.com/es/topics/integration/whats-the-difference-between-soap-rest

4.2. Aplicaciones web

Normalmente, las dos partes de una aplicación web son el front-end, parte pública que ven los usuarios externos; el back-end, parte privada que usan los administradores.

Ejecución de código

Cuando un navegador solicita a un servidor web una página, probablemente éste, antes de enviarla, haya tenido que ejecutar algún programa para obtenerla. Ese código, que genera, en parte o en su totalidad la página web, se ejecuta en el entorno del servidor web.

Además, una vez la página esté en el cliente, probablemente se tenga que ejecutar algún fragmento más de código (normalmente Javascript).

Por ejemplo, en el caso de una aplicación de correo web, los mensajes y su contenido se obtendrán en el servidor de una base de datos, mientras que el navegador ejecutará el código encargado de avisar que no se ha incluido asunto cuando se quiere enviar un mensaje.

Desde hace unos años existe una técnica de desarrollo web, AJAX, que posibilita realizar programas en los que el código JavaScript se puede comunicar con un servidor para obtener información con la que modificar la página web y así, sin salir de ella poder modificar su contenido en base a la información que se almacena en un servidor de Internet.

Tecnologías / Plataformas web

Al ir a programar una aplicación web, se hace utilizando una tecnología y arquitectura determinadas. Hay una gran variedad de opciones, por ejemplo:

- Java EE (Enterprise Edition): Es una plataforma orientada a la programación de aplicaciones en Java. Es una buena solución para el desarrollo de aplicaciones de medianas o grandes. Una de sus principales ventajas es la multitud de librerías existentes en ese lenguaje y la gran base de programadores que lo conocen.
- AMP(Apache, MySQL, PHP/Perl/Python): Será LAMP (Linux), WAMP (Windows) o MAMP (para Mac). Todos los componentes de esta arquitectura son de código libre. Es una plataforma de programación que permite desarrollar aplicaciones de tamaño pequeño o mediano con un aprendizaje sencillo.
- XAMPP(Apache, MySQL/MariaDB, PHP/Perl): distribución de Apache que incluye varios softwares libres. La inicial X se usa para representar a los sistemas operativos Linux, Windows y Mac OS X.
- CGI/Perl: Perl, lenguaje de código libre creado para la administración de servidores, y CGI, estándar para permitir al servidor web ejecutar programas genéricos escritos en cualquier lenguaje que devuelven páginas web (HTML) como resultado de su ejecución. Es la más primitiva de estas arquitecturas.
- ASP.Net: Es la parte de la plataforma .Net para la generación de páginas web dinámicas. El lenguaje de programación puede ser Visual Basic.Net o C#. Utiliza el servidor IIS, y puede utilizar varios gestores de bases de datos entre los que se incluye, Microsoft SQL Server. Plataforma comercial de código propietario.

Despliegue

Cuando se está desarrollando una aplicación, es necesario tener, al menos, dos entornos: desarrollo y producción. El entorno de desarrollo suele ser el equipo de cada uno de los desarrolladores que participan en el proyecto, mientras que el entorno de producción, es el equipo utilizado cuando se va a desplegar la aplicación y hacerla pública.

Pero es bastante habitual que, durante el ciclo de desarrollo y despliegue de una aplicación, existan entornos adicionales para cada fase por la que pasa el proyecto:

- Un entorno para que el cliente pueda realizar pruebas y testing
- Un entorno de preproducción o staging para el propio equipo de programadores.
- Un entorno para que el QA (Quality Assurance) encuentre bugs.

En el entorno de desarrollo se programa el software. Puede ser el propio ordenador del programador o incluso un servidor compartido por todos los participantes en el desarrollo para que creen la aplicación. Este entorno debe parecerse lo máximo posible al entorno de producción para evitar problemas en el despliegue final.

El entorno de pruebas o testing es en el que los desarrolladores ejecutan los tests y realizan las pruebas de una determinada funcionalidad. Cuando se trabaja como parte de un equipo en el que muchas personas desarrollan de forma individual tareas para el mismo proyecto, es necesario probar cada integración para verificar, por un lado que sea lo que se espera y por otro que no haya interferido con el trabajo realizado por otro integrante del equipo. De la misma manera que el entorno de desarrollo, este entorno debe parecerse al máximo al entorno de producción.

Las pruebas de aceptación de usuario (User Acceptance Testing) forman la última fase de un proceso de pruebas. En este entorno, los clientes realizan pruebas para asegurarse de que los requisitos de desarrollo de software se han cumplido, es decir, que los desarrolladores han hecho una funcionalidad tal y como se ha pedido y el software es completamente usable. Esta es una de las fases más importantes en el ciclo de desarrollo de software porque en ella se asegura que el usuario final está contento con el resultado.

El entorno de preproducción o staging es, como su propio nombre indica, el último entorno sobre el que se va a desplegar la aplicación antes de que vaya a producción. En este entorno, lo mejor sería que además de tener la misma configuración software que en el entorno de producción, se tuviese también la misma configuración hardware. De esta manera, un QA podría identificar errores incluso de rendimiento.

El entorno de producción es accesible a todo el mundo. Si se han configurado todos los entornos de la misma manera, realizado pruebas exhaustivas del software, tests automatizados y seguido buenas prácticas, no debería aparecer ningún problema en el despliegue.