UT 1: SELECCIÓN DE ARQUITECTURAS Y HERRAMIENTAS DE PROGRAMACIÓN.

1.1. EVOLUCIÓN DE LA WEB.

Las aplicaciones Web se basan tanto en Internet (1969) como en la World Wide Web (1990) y es importante tener en cuenta que no son la misma cosa.
Internet, **es la red de redes** es la superautopista de la información fundamentada en las mismas tecnologías de comunicación que la red ARPAnet en la cual tuvo su origen; mientras que **WWW** o "la Web" o W3 o World Wide Web es una colección de **información multimedia basada en hipertexto que es accesible a través de Internet**.

El origen. 1990. El CERN (Consejo Europeo para la Investigación Nuclear) utilizaba un sistema de hipertexto para gestionar su información y esta herramienta de trabajo fue el origen de la actual WWW. Los sistemas de navegación eran en modo texto.
La era Web 1.0

- 1991-1993: CERN liberó esta tecnología y aparecieron los nuevos servidores web fuera de este entorno y el primer navegador gráfico (Mosaic). A continuación Netscape Navigator y Microsoft Internet Explorer y más tarde Opera, Mozilla Firefox y Safari**. Los servidores web se limitaban a proporcionar páginas estáticas**.
- 1994: Se creó el **W3C** (World Wide Web Consortium) comunidad internacional para crear estándares ante la proliferación de tecnologías web y evitar las posibles incompatibilidades entre los diferentes sistemas.
- 1993: **CGI** (Common Gateway Interface) es la primera herramienta para la generación de páginas web que permite la interacción cliente-servidor al ejecutar código. A través de la URL solicitada por el navegador se pasan los parámetros de entrada para el script que se ejecuta en el servidor; éste devuelve una página HTML y que es interpretada por el navegador. PHP, Java y .NET de Microsoft entre muchas otras, son tecnologías que han hecho posible desarrollar en el lado del servidor aplicaciones web dinámicas e interactivas cada vez más complejas.
- 1995: Sun Microsystems y Netscape presentan la primera versión de JavaScript, permitiendo el diseño de aplicaciones web interactivas también en el lado del cliente.

La era Web 2.0.
La llamada web 2.0, donde los **contenidos dinámicos y personalizados y la interacción con el usuario** son la principal novedad, hace que el uso de aplicaciones ejecutadas en un servidor **crezca exponencialmente**. Este tipo de aplicaciones son cada vez más complejas y deben dar servicio a un número de usuarios cada vez mayor y no es suficiente con utilizar plataformas de desarrollo rápido, como por ejemplo Wix, que se basan en HTML. Comienzan a desarrollarse aplicaciones modeladas por capas donde servidores especializados realizan funciones concretas. Es la era de las **plataformas colaborativas y las redes sociales.**

La era web 3.0. El uso de dispositivos móviles como tabletas y smartphones que usan intensivamente las aplicaciones de la red hace que el desarrollo de las tecnologías usadas en el servidor tenga que dar nuevas respuestas a nuevos problemas. Los lenguajes de programación deben ofrecer soluciones nuevas a estos problemas. Es la era de la semántica y de la inteligencia artificial.

La era Web 4.0. Básicamente consiste en potenciar el uso de la inteligencia artificial para la obtención de la información solicitada por el usuario.

La World Wide Web ha tenido tal impacto en nuestras vidas desde su aparición en 1990 que hoy en día la mayoría de las empresas a través de las tecnologías Web tienen presencia en Internet, hasta el punto que grandes empresas como Google, Amazon,…tienen la base de su negocio en dichas tecnologías**. El éxito en un negocio hoy precisa de una aplicación web** que ofrezca un amplio abanico de servicios y de fácil acceso y gestión.

1.2.- LAS APLICACIONES WEB FRENTE A LAS APLICACIONES DE ESCRITORIO.

Las **aplicaciones web** se encuentran alojadas en un servidor (local o remoto) y son accesibles a través de un navegador y la URL correspondiente, pudiendo ser accesibles por muchos usuarios que se conectan a este servidor para utilizar la aplicación. **Están basadas en la arquitectura de sistema cliente-servidor.**

No todo el código de una aplicación web se ejecuta en el servidor; por ejemplo, en el se ejecutará PHP, JSP, Perl, ASP,… y en el cliente JavaScript, VBScript, Applets Java, HTML, CSS,…

En cambio las **aplicaciones de escritorio** es preciso que estén instaladas de forma local en un equipo y son accesibles simultáneamente por un único usuario.

Ventajas de las aplicaciones web.

- **No es necesario instalar software especial en los clientes.** Sólo se necesita disponer de un navegador web. Además las actualizaciones se realizan principalmente en el servidor y es donde se ejecuta la mayor parte del código de la aplicación.
- **Acceso a la última versión de la aplicación web.** El acceso se realiza de forma centralizada a un servidor web evitándose las posibles versiones y desactualizaciones de la aplicación.
- **Seguridad y copias de seguridad.** Los datos también están centralizados con lo cual es más sencillo establecer y llevar un control de políticas de seguridad.
- **Movilidad.** Cualquier usuario con un dispositivo móvil puede acceder a un servidor web en Internet.

Inconvenientes de las aplicaciones web.

- **La disponibilidad depende de un tercero** o bien de la conexión a internet o bien del que provea el enlace entre la aplicación y el cliente.

1.3.- TIPOS DE APLICACIONES WEB.

Cualquier proyecto que se quiera desarrollar en Internet, bien sea comercio electrónico, reservas de billetes de vuelo on-line, información meteorológica, registro de usuarios, simuladores de hipotecas, etc, conlleva el desarrollo de una aplicación web.

Las aplicaciones web pueden ser clasificadas en base a muchos criterios (tecnología utilizada, modelo arquitectónico elegido,…). Debes saber que:

- Una **aplicación web estática** está implementada en HTML y ofrecen siempre el mismo contenido.
- **Una aplicación web animada** utiliza la tecnología FLASH entre otras y permite que una página web presente el contenido con ciertos efectos animados continuados (objetos en movimiento tales como banners, GIF animados, vídeos, etc.)
- Una **aplicación web dinámica** surgen cuando el contenido que ofrece la aplicación parte se genera en el cliente y parte en el servidor; son aplicaciones con código embebido en HTML es decir, es posible incluir código de un lenguaje de programación web como PHP, JSP, ASP.NET, Python, Ruby,...
- Una **aplicación web interactiva** es aquella en la que el usuario además interactúa con la aplicación, de tal forma a través de sucesivos eventos es posible "personalizar" la aplicación. *(por ejemplo, si ejecutas un juego en red es muy posible que quieras seguir donde lo dejaste en la partida anterior, ello conlleva un control de sesión.)*

Son muchas las aplicaciones web dinámicas e interactivas que podemos encontrar como son los portales, buscadores, tiendas on-line, gestores de contenido, etc.

1.4.- PLATAFORMAS WEB. DEF Y TIPOS. LIBRES Y PROPIETARIAS.

Una plataforma web es el entorno de desarrollo de software empleado para diseñar y ejecutar aplicación web.
Básicamente, consta de cuatro componentes:

- El **sistema operativo,** bajo el cual opera el equipo donde se hospedan las páginas web y que representa la base misma del funcionamiento del computador. En ocasiones limita la elección de otros componentes.
- El **servidor web** es el software que maneja las peticiones de los clientes web adecuadamente.
- El **gestor de bases de datos** se encarga de almacenar los datos que maneja la aplicación web
- **Un lenguaje de programación** que será la base a utilizar para el desarrollo de la lógica de las aplicaciones web.

Son muchas las combinaciones posibles de estos cuatro componentes que dan lugar a numerosas plataformas web libres o propietarias. Cabe destacar las siguientes:

- **Plataforma LAMP,** trabaja enteramente con componentes de **software libre** y no está sujeta a restricciones propietarias. Componentes que la integran:
  - ✓ **L**inux: Sistema operativo.
  - ✓ **A**pache: Servidor web.
  - ✓ **M**ySQL: Gestor de bases de datos.
  - ✓ **P**HP: Lenguaje interpretado PHP, aunque a veces se sustituye por Perl o Python.
- **Plataforma WISA,** basada en tecnologías desarrolladas por la compañía Microsoft (**software propietario).** Componentes que la integran:
  - ✓ **W**indows: Sistema operativo.
  - ✓ **I**nternet Information Services: servidor web.
  - ✓ **S**QL Server: gestor de bases de datos.
  - ✓ **A**SP o ASP.NET: como lenguaje para scripting del lado del servidor.
- **Plataforma WAMP, plataforma híbrida**. Se utiliza principalmente para desarrollo web local. Componentes que la integran:
  - ✓ **W**indows: Sistema operativo.

- ✓ **A**pache: Servidor web.
- ✓ **M**ySQL: Gestor de bases de datos.
- ✓ **P**HP: Lenguaje interpretado PHP

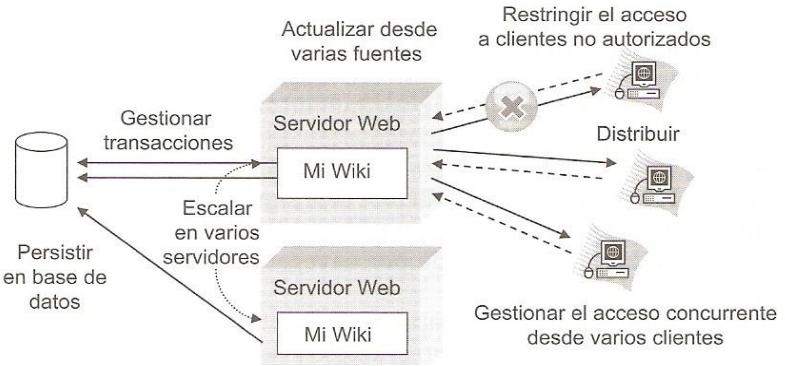
- **Plataforma WIMP,** basada en tecnologías desarrolladas por la compañía Microsoft (**plataforma híbrida).** Componentes que la integran:
  - ✓ **W**indows: Sistema operativo.
  - ✓ **I**nternet Information Services: servidor web.
  - ✓ **M**ySQL: Gestor de bases de datos.
  - ✓ **P**HP: Lenguaje interpretado PHP

Existen muchas otras plataformas que trabajan con otros distintos sistemas operativos (MacOS, Solaris) y servidores web.

1.5.- SERVIDORES WEB FRENTE A SERVIDORES DE APLICACIONES.

¿Qué nos permite una aplicación web?:

- Distribuir información por Internet.
- Gestionar la concurrencia, permitiendo el acceso a un mismo recurso web a muchos usuarios simultáneamente.
- Generar contenido dinámico, es decir construir páginas Web sobre la marcha a partir de las aportaciones realizadas por los usuarios.
- Incluir seguridad, es decir controlar el acceso al recurso web.
- Conectar con una base de datos que almacene los datos de manera persistente, con actualizaciones fiables y consistentes.
- Ejecutar la misma aplicación en distintas máquinas permitiendo la escalabilidad del sistema y su portabilidad.



¿Cuándo utilizar un servidor web y cuando un servidor de aplicaciones?

Teniendo en cuenta **qué debe permitir una aplicación web**, **se elige la tecnología a utilizar para su desarrollo y su posterior ejecución**, tecnologías estrechamente relacionadas. Si la aplicación web por ejemplo, está basada en **la tecnología PHP,** será necesario un **servidor web** como **Apache** y si la tecnología para su desarrollo implica utilizar componentes propios de la **plataforma Java EE** es necesario un **servidor de aplicaciones** como puede ser **Tomcat.**

Es importante elegir una tecnología que permita "crecer" a la aplicación web; hay aplicaciones que mueren de éxito, son muy solicitadas, pero no han sido pensadas en su diseño para que puedan soportar muchas peticiones y por ello mueren.

Se dice que una **aplicación web es una extensión dinámica de un servidor web o un servidor de aplicaciones** y que un **servidor web o de aplicaciones es el software que alberga las aplicaciones web.**

**Nota: No todo servidor web es un servidor de aplicaciones, pero todo servidor de aplicaciones sí en un servidor web. Apache no es de aplicaciones pero Tomcat si es un servidor web (en realidad "incluye" un servidor web denominado Coyote).*

1.7.- LENGUAJES DE PROGRAMACIÓN DE SCRIPTS DEL LADO DEL SERVIDOR

Son múltiples; se cita alguna de ellas:

- **PHP** *(Hypertext Processor).* Tecnología muy utilizada y soportada por la mayor parte de los servidores web.
- **ASP** *(Active Server Pages).* Actualmente, ASP.Net. Puede ser ejecutada en otros servidores web pero fue diseñada para ejecutarse el IIS *(Internet Information Server)*, es decir el servidor web de Windows.
- **JSP** (Java Server Pages). Tecnología Java. Requiere para su ejecución de un servidor de aplicaciones como puede ser Tomcat.

1.6.- ARQUITECTURAS WEB.

1.6.1.- ¿Qué es una arquitectura web?

Al igual que en un determinado momento fue necesaria la definición de un estándar de comunicaciones ya que la conexión de equipos no sólo era propia de ARPAnet, se define el modelo OSI, surgen las arquitecturas de red "reales" (denominadas "tecnologías de red") y se definió la arquitectura de red TCP/IP, debido a la evolución de las aplicaciones web ha sido también necesaria la definición de modelos de arquitecturas web que corren a cargo de W3C y la definición de arquitecturas web "reales" (que también se denominan tecnologías web). En ambos casos el punto de partida es el mismo: el modelo cliente-servidor.

Una arquitectura web es un conjunto ordenado de subsistemas, cada uno de los cuales está constituido en términos de los que tiene por debajo y proporciona la base de la implementación de aquellos que están por encima de él. Cada subsistema utiliza una tecnología web diferentes entre varias posibles para implementar su función.

A medida que la sofisticación de los servicios ofrecidos vía Web ha ido aumentando, se ha ido añadiendo, al menos a nivel lógico y no necesariamente a nivel físico, **nuevos elementos en la arquitectura de las aplicaciones Web** (por ejemplo el servidor de aplicaciones, bases de datos,...).utilizando para ello como referencia los patrones arquitectónicos propios de las aplicaciones distribuidas.

1.6.2.- Características generales de las arquitecturas web.

- **La Escalabilidad**. Es muy importante que, ante un aumento de usuarios, una aplicación esté preparada para su adaptabilidad a una nueva situación. Generalmente, se requiere de un cluster de servidores.
  - **Escalabilidad horizontal (** se clona el sistema y se balancea la carga, es decir la peticiones de los usuarios; problemas con el mantenimiento de sesión (utilizar herramientas como las cookies)
  - **Escalabilidad vertical** (las capas lógicas se implementan físicamente separadas; es necesario un *Middleware* entre las capas para permitir la comunicación remota)
- **Separación de responsabilidades.** Consiste en ubicar los distintos componentes de una aplicación web en diferentes capas; se divide con ello **la funcionalidad de la aplicación web**. Cada componente software tiene un rol perfectamente diferenciado. *(es la base de las arquitecturas multicapas)*
- **Portabilidad.** Una aplicación web debe poder adaptarse a las distintas arquitecturas físicas posibles en el despliegue. La adaptación a un nuevo entorno deben limitarse al ámbito de la configuración, no del desarrollo (por ejemplo, hay navegadores que no están adaptados a saber interpretar las diferentes tecnologías de componentes J2EE como son los EJBS.
- **Gestión de la sesión de usuario.** Es un aspecto crítico en el desarrollo de una aplicación.
- **Aplicación de patrones de diseño.** Es necesario que sea viable diseñar patrones a partir de las arquitecturas web, tal es el caso del **patrón de diseño MVC.**

1.6.3.- Tipos de arquitecturas web.

- **ARQUITECTURA WEB ORGANIZADAS EN CAPAS** *(cuando se asocia una función a una componente software)*

Una arquitectura en capas es aquella donde vemos diferentes partes del software cada una con un rol perfectamente diferenciado. Ésta es la forma conceptual (y no necesariamente física) de dividir componentes de un sistema de capas.

Las arquitecturas en capas **dividen la funcionalidad de la aplicación web** para optimizar el uso de recursos. Se consiguen soluciones mucho más **flexibles y escalables**.

La arquitectura web actual definen 3 capas:

Capa de presentación
Capa de lógica de negocio
Capa de datos

- ✓ **Capa de presentación:** procesa las peticiones del cliente web y, tras procesarlas, las reenvía a la capa de nivel inferior (capa de lógica de negocio), y viceversa, procesa los datos procedentes de la capa inferior y los adapta al "lenguaje" entendible por la capa de cliente. Funciones asociadas a esta capa:
  - Navegabilidad del sistema
  - Validación de datos de entrada
  - Formato datos de salida
  - Internacionalización
  - Renderizado de presentación,…

- ✓ **Capa de lógica de negocio:** Lo que distingue una **aplicación Web** de un mero sitio Web reside en la posibilidad que ofrece al usuario de actuar sobre la **lógica de negocio en el servidor**. Por lógica de negocio se entiende un conjunto de procesos que implementan las reglas de funcionamiento de la aplicación Web. Por ejemplo, en el caso de una tienda Web, la lógica de negocio son los procesos que implementan el procedimiento de compra: selección de productos, carro de la compra, confirmación de compra, pago, seguimiento de la entrega,...Pero lo importante es que a los **usuarios** se les ofrece la posibilidad de **actuar sobre la lógica de negocio**. Lo cual conlleva que en las aplicaciones Web exista una **lógica de negocio sensible a las interacciones del usuario;** es el punto donde se puede llevar a cabo la coordinación de transacciones (operaciones de múltiples usuarios). Las operaciones con un uso intensivo de datos deben ejecutarse en este nivel.

- ✓ **Capa de datos o persistencia:** Es la capa que interactúa con los sistemas de información como son las bases de datos (*inserción, borrado, búsquedas y actualizaciones*). Se especializa en dar un servicio de persistencia a los datos de la aplicación y permite manejar grandes volúmenes de ellos.

  La utilización de este tipo de arquitecturas es la base del desarrollo de aplicaciones web "orientadas a presentación" las cuales generan páginas web interactivas en distintos lenguajes de marcado (HTML, XML, etc.) y contenido dinámico a las peticiones de los clientes. *(Por ejemplo, en el caso de una tienda Web la aplicación permitirá al usuario registrarse con lo que se crea un perfil personalizado, se hace un seguimiento de sus compras a lo largo de una sesión (carro de la compra) pero también a través de sesiones (perfil de comprador), con sus acciones se modifican stocks, etc.,...)*

- **ARQUITECTURA WEB EN NIVELES. SISTEMAS DISTRIBUIDOS. (***(cuando se asocia una función a una componente hardware)*

Cuando hablamos de niveles, indirectamente estamos hablando de capas. Sin embargo, la diferencia es que el término "niveles" generalmente se utiliza cuando se trata de dispositivos físicamente separados.
Un sistema con varios niveles es por tanto aquel que se encuentra en diferentes nodos (equipos).
El uso de diferentes niveles es necesario cuando queremos hacer nuestras aplicaciones distribuidas y escalables.

Este tipo de arquitecturas son parte fundamental de las aplicaciones Web ya que la capa de presentación (que se ejecuta en los navegadores) está totalmente dispersa y el elevado número de usuarios que van a utilizarlas hace que las capas de lógica de negocio y de gestión de datos tengan que estar distribuidas también entre diferentes máquinas.

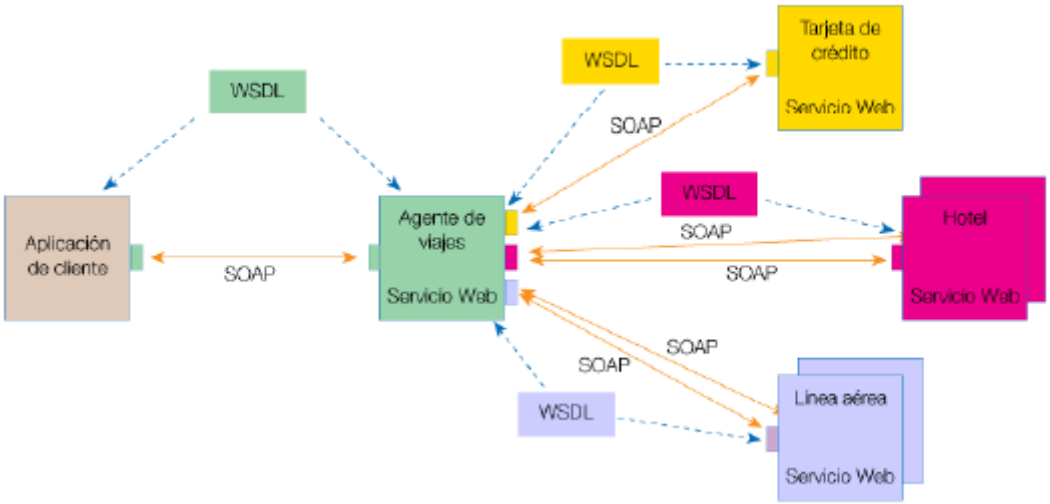
Por ejemplo, en un sistema de banca en la Web, **la capa de presentación** (un navegador) estará **distribuida entre los equipos de los distintos usuarios**, **la capa de negocio** se estará ejecutando de forma centralizada en un único equipo, o en varios equipos, dentro del banco (gestión de cuentas, transferencias de fondos, etc.) y **la capa de gestión de datos** también puede ejecutarse en una máquina distinta (o varias).

- **ARQUITECTURA ORIENTADA A SERVICIOS (SOA**, *Service-Oriented Architecture*).

Esta arquitectura permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización y la exposición e invocación de servicios (*servicios web*), lo cual facilita la interacción entre diferentes sistemas propios de una misma organización o de otras. La tecnología más comúnmente utilizada es SOAP y WDSL.

Al comienzo de la era Web, el centro del desarrollo de aplicaciones consistía en dónde se iban a utilizar. Por ejemplo, las guerras entre navegadores entre Netscape y Microsoft a mediados de los noventa consistían en ver qué aplicación se impondría para acceder a la Web.
En la actualidad, el desarrollo de aplicaciones se ha centrado mucho más en el contenido que se hará disponible en vez de la aplicación que se va a utilizar para hacerlo visible. Este contenido se pone a disposición del resto del mundo utilizando diferentes versiones de servicios Web.
Los servicios Web son fuentes de datos que se ponen a disposición de cualquiera utilizando XML; un ejemplo sencillo de un servicio Web es RSS (Sindicación Realmente Sencilla) y que utiliza XML para proporcionar fuentes de información actualizada con mucha frecuencia como pueden ser noticias o información meteorológica.

La utilización de este tipo de arquitecturas es la base del desarrollo de aplicaciones web "orientadas a servicio".



1.6.4.- Modelos arquitectónicos. Modelo-Vista-Controlador (MVC)

Este modelo se utiliza ampliamente en el entorno Web. La división de la aplicación en niveles supone la necesidad de establecer interfaces entre ellas. Al ser una aplicación Web, **la interfaz con el cliente** sigue siendo básicamente **HTTP. Para la interfaz entre el servidor y la base de datos las opciones son múltiples**, dependiendo del

lenguaje de programación (Java, C, PHP, VisualBasic,...), el tipo de base de datos (relacional, XML,...), la base de datos concreta (MySQL, Oracle, ...), etc. En cualquier caso con cada configuración hay más de una opción y lo normal es plantearse la elección tras definir la BD o el lenguaje de programación a utilizar.

La estructura de este modelo o patrón arquitectónico es la siguiente:

- **Vista:** presentan la información a los actores de la aplicación.
- **Controlador:** gestiona los eventos de los usuarios y permite la comunicación entre la Vista y el Modelo.
- **Modelo:** la lógica de negocio y los datos asociados a la aplicación.

Al mantener buena parte del código en la capa de lógica de negocio, **la aplicación se aísla de la interfaz de usuario y de la base de datos**. Por lo tanto, se pueden hacer cambios en el código en este nivel sin que esto afecte al resto. Además, los **datos** están **centralizados** y fácilmente accesibles sin necesidad de moverlos completamente hasta el cliente.

1.7.-ENTORNOS DE DESARROLLO.

Son varias las herramientas que intervienen en el desarrollo de una aplicación web:

- **Navegadores o browsers** (Edge, Chrome, Firefox,…). A través de una url se solicita el recurso web (aplicación web) al servidor
- **Editores específicos** (text markers, ej. Notepad++, phpStrom, Brackets, Atom,…)
- **Framework** (Laravel, Symfony,…)
- **IDE** (entornos de desarrollo integrados: editar, depurar y compilar)
  - Genéricos (Eclipse para Java, C, PHP, Visual Studio Code….)
  - Específicos (Visual Studio para .NET, Netbeans para Java,…)
- **Herramientas tratamiento de imágenes (**ej. GIMP)
- **Creación y administración de BBDD** (ej: PhpMyAdmin)

1.7.1.- Entorno local.

Principalmente, en este módulo trabajaremos en un entorno local. Utilizaremos el **IDE Visual Studio Code** y la **plataforma web WampServer**. Con ella nuestros proyectos web se alojarán por defecto en Wamp64\www.

Para poder acceder a las aplicaciones web que vayamos desarrollando hay que solicitar al servidor una petición de acceso, lo cual se lleva a cabo escribiendo en el navegador web su dirección **URI** o **URL** (*términos equivalentes que identifican o localizan un recurso universal en la Web*)
La estructura de dicha dirección es:

- El **protocolo** (*por ejemplo, http*)
- La **dirección IP o el nombre de dominio** del servidor (*por ejemplo,* www**.**educa.jcyl.es,***www.debian.org*** o por ejemplo, localhost como nombre DNS de propósito específico)
- El **puerto** donde se realiza la petición (*por defecto, el puerto 80*)
- EL **directorio y el archivo** (*por ejemplo, index.html*), donde se encuentra ubicado el documento que el servidor debería enviar al cliente.

Todos estos campos van separados por caracteres específicos.
La dirección resultante de los ejemplos sería: ***http://www.debian.org:80/index.html*****.**

- Normalmente, tanto el servidor web como el cliente web pueden asumir valores por defecto, tal es el caso de:
  - el protocolo (http)
  - el puerto (80)
  - el directorio (real o virtual)
  - el archivo predeterminado (index.html, index.htm, index.php, etcétera).

1.7.2.- Entorno remoto. Servicio de hosting.

El servicio de hosting consiste en contratar a un proveedor de servicios de internet ciertos servicios.

Los servicios más comunes que se pueden contratar:

- Servidor web o de aplicaciones.
- Entornos de desarrollo.
- Alojamiento de aplicaciones o páginas web.
- Alojamiento de ficheros y acceso vía web a los ficheros para subidas, descargas, edición, borrado, etc.
- Acceso a ficheros vía FTP.
- Creación de bases de datos y administración vía web
- Gestión de cuentas de correo electrónico de un dominio propio y acceso vía webmail a estas cuentas.
- Discos duros virtuales que se pueden configurar como unidad de red en un equipo local
- Copias de seguridad.
- Gestión de dominios y subdominios.
- Estadísticas de tráfico.
- Asistentes para la instalación rápida de paquetes software libre populares como WordPress, Joomla, etc.

Las características del servicio de hosting dependerán de las necesidades del solicitante siendo posible los siguientes tipos:

Alojamiento compartido: Servidores que alojan múltiples sitios web, donde cada sitio web está asociado a una aplicación web. En este tipo de servicio **se alojan varios sitios en un mismo servidor**, gracias a la configuración del servidor web. Ventaja: reducción de costes. Inconveniente: disminución de la velocidad de acceso y de

recursos del servidor. Especial atención a la seguridad ante el elevado número de accesos.

Servidores virtuales (*VPS, Virtual Private Server*): En este caso el alojamiento web se realiza utilizando servidores configurados en máquinas virtuales y aparentemente el servicio es no compartido. De esta manera se mejora la administración del sitio web y las necesidades de un cliente al instalar los programas que necesite en dicho servidor. Es un tipo de alojamiento muy utilizado por las empresas de diseño y programación web.
Servidores dedicados: La principal diferencia con respecto al alojamiento compartido es el costo del servicio siendo muy superior al del alojamiento compartido debido a:

- el uso "exclusivo" del servidor y de sus recursos hardware
- la necesidad de contratar los servicios para la administración y configuración del servidor. En este caso se puede hablar de:
    - ***Hosting administrado***: la empresa a la que se contrata el servicio de alojamiento web además de ofrecer la conectividad, recursos, panel de control y todas las herramientas necesarias para administrar el servicio contratado incluirá la asistencia para los fallos, desconfiguraciones, o errores causados por la aplicación web que se estén ejecutando e incluso asesoramiento en la creación de los scripts.
    - ***Hosting no administrado***: el servicio de asistencia anteriormente mencionado no se cubre.

Alojamiento web en la nube: El alojamiento web en la "nube" (cloud hosting) está basado en las tecnologías más innovadoras (*virtualización de servidores*) que permiten a un gran número de máquinas actuar como un único sistema conectadas a un grupo de medios de almacenamiento (NAS). Ventajas:

- ***Los recursos:*** las tecnologías de computación en la nube eliminan cualquier limitación física para el crecimiento en tiempo real.
- ***La seguridad:*** La seguridad de un sitio web alojado en la "nube" (cloud) está garantizada por numerosos servidores en lugar de sólo uno.

La computación en la nube es un modelo de acceso a los sistemas informáticos, en el que los datos y las aplicaciones están hospedados en Internet y en centros de cómputo remotos, de tal modo que pueden ser utilizados desde cualquier punto que tenga conexión a la red mundial. La computación en la nube permite que los consumidores y las empresas gestionen archivos y utilicen los programas, sin necesidad de instalarlos localmente en sus computadores. Esta tecnología ofrece un uso mucho más eficiente de los recursos, tales como almacenamiento, memoria, procesamiento y ancho de banda.

El término "nube" se utiliza como una metáfora de Internet, y se origina en la nube utilizada para representar Internet en los diagramas de red, como una abstracción de la infraestructura que representa.

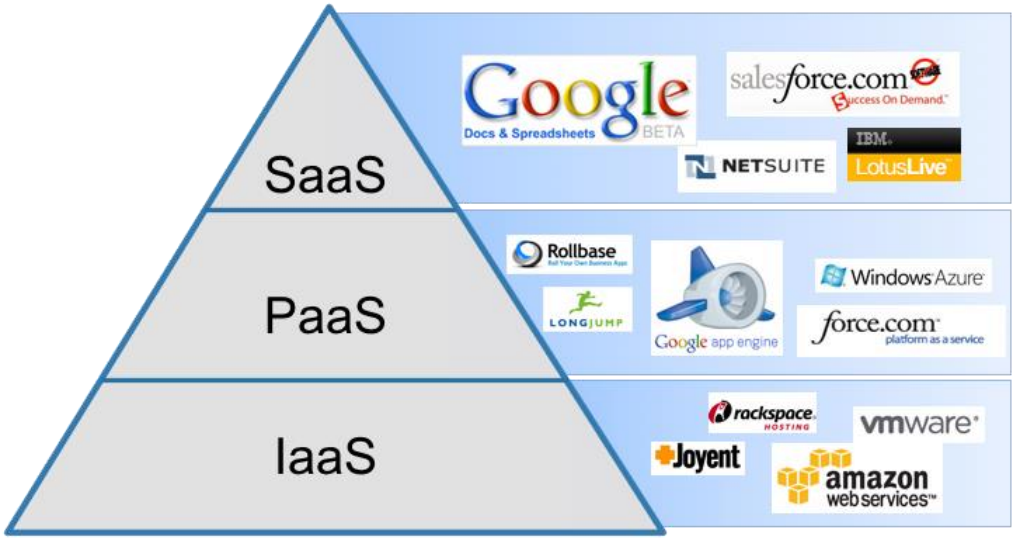
Un ejemplo sencillo de computación en la nube es el sistema de documentos y aplicaciones electrónicas Google Drive / Google Apps. Para su uso no es necesario comprar ni instalar software o disponer de un servidor, basta con una conexión de banda ancha para poder utilizar cualquiera de sus servicios. El servidor y el software de gestión se encuentran en la nube (Internet) y son directamente administrados por el proveedor de servicios. De esta manera, la tecnología de la información se convierte

en un servicio, que se consume de la misma manera que consumimos la electricidad o el agua.

En el pasado (y presente), el uso tradicional del PC no ha cambiado mucho: Instalamos un sistema operativo. Tras instalarlo, buscamos aplicaciones (software), que también instalamos en nuestro equipo y que nos permitirá realizar diferentes tareas. Tenemos, pues, aplicaciones de Oficina (Word, Excel), aplicaciones de diseño o fotografía (Photoshop, Illustrator), aplicaciones multimedia (Reproductor Musical, de Videos). Se podría mencionar infinidad de software que descargamos, instalamos, y utilizamos. En contraposición a lo anterior, la idea del cloud computing es que, gracias a los avances tecnológicos, ya no necesitamos que estas aplicaciones residan en nuestra PC, pues podemos acceder a servicios similares, desde el navegador. ¿Necesitan editar documentos? Existen alternativas como Google Drive u Office 365, que nos ofrecen herramientas similares ¿Música? Existe spotify; ni siquiera tienen que descargar o tener las canciones. Éstas pueden vivir tranquilamente en la nube, y podemos acceder a las canciones no sólo localmente (desde un computador), sino que podemos hacerlo de otra PC, de nuestro teléfono, o cualquier otro dispositivo conectado a la web.

Desarrollo de aplicaciones en la nube. Existen distintas formas de hacerlo:

- **Software-as-a-Service (SaaS)**, básicamente se trata de cualquier servicio basado en la web; el desarrollo y mantenimiento, así como la realización de copias de seguridad es responsabilidad del proveedor del servicio. Por ej: Google Docs, Salesforce, Dropbox,…
- **Platform-as-a-Service (PaaS), plataformas que ofrecen al desarrollador toda la infraestructura necesaria para desarrollar y desplegar sus aplicaciones.** El desarrollador sólo se preocupa de optimizar el código de sus aplicaciones con el fin de consumir los menos recursos posibles (accesos a disco, espacio para la aplicación, tiempo de respuesta, etc.). La plataforma ofrece la infraestructura adecuada para garantizar la escalabilidad, número de peticiones, etc. **Por ej: Windows Azure, Google app Engine, OpenShift (Red Hat)**
- **Infraestructure-as-a-Service (IaaS)**, son plataformas en las que, además de encargarnos del desarrollo de aplicaciones, también nos encargamos de la gestión de la infraestructura necesaria para su despliegue (recursos necesarios para la escalabilidad de la aplicación). Por ej: AWS (Amazon Web Services)



ACTIVIDAD: ¿Qué es Windows Azure? ¿Qué es Google App Engine?