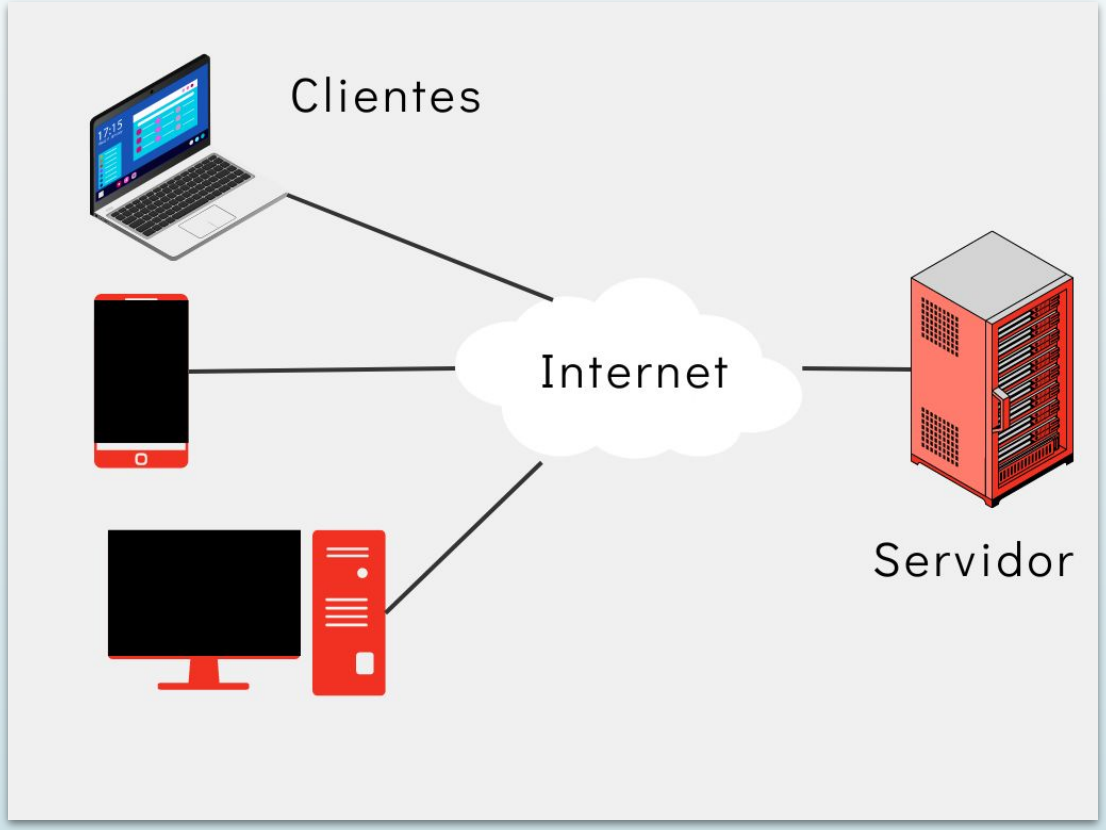
TEMA 1

Arquitecturas web

Arquitecturas Web: Modelo Cliente-Servidor

- Arquitectura Web: Define el modo en que las páginas de un sitio web están estructuradas y enlazadas entre sí.
- Estas arquitecturas están basadas en el modelo cliente-servidor, que es un modelo distribuido en el que hay dos tipos de elementos:
  - Servidor: Provee servicios, como información o funcionalidad, a los clientes
  - Cliente: Solicita un servicio a un servidor
- El cliente inicia el proceso, enviando una solicitud al servidor, que a su vez envía un mensaje de respuesta
- Cliente y servidor se comunican mediante un protocolo
- En el caso de los servidores web, los clientes suelen ser navegadores, y el protocolo el HTTP o el HTTPS
- El desarrollo web está condicionado por las características del modelo cliente-servidor y del protocolo HTTP(S)

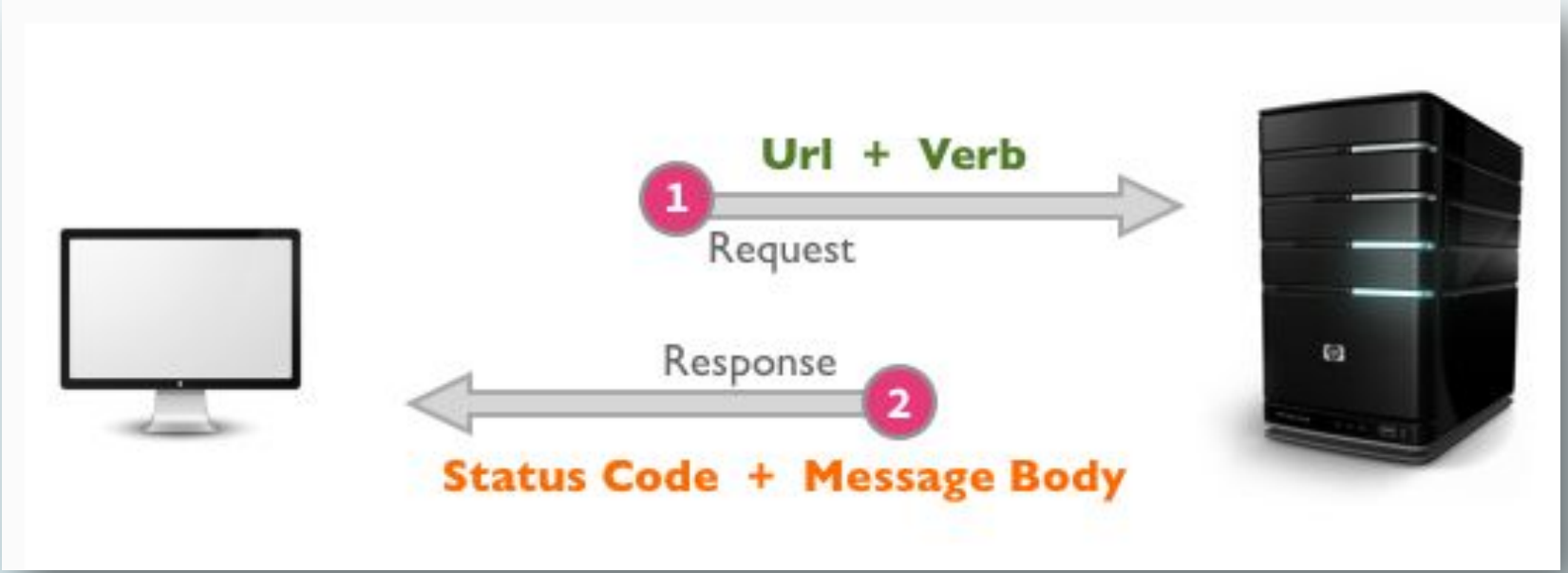
Modelo Cliente-Servidor



Protocolo HTTP: Request-Response

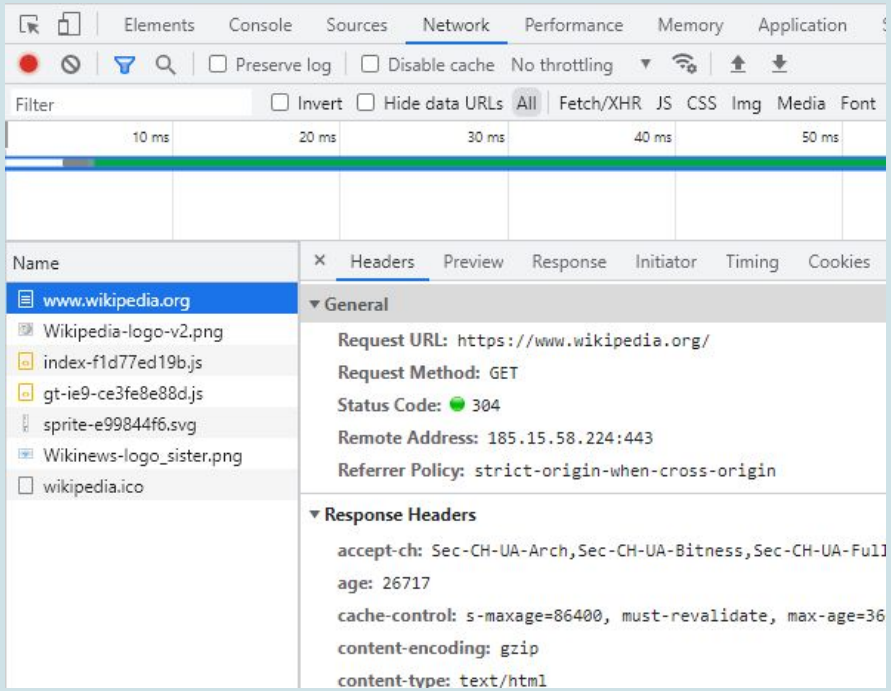
- Este protocolo permite el entendimiento entre clientes y servidores web:
  1. El cliente realiza una petición (**request**) al servidor, a través del puerto 80 (en el caso de HTTP) y del puerto 443 (en el caso de HTTPS)
  2. La petición consiste en la solicitud de un documento a un servidor concreto a través de un puerto.
  3. La **URL** (uniform resource locator) es la ruta completa para la localización de dicho documento
  4. El servidor envía como respuesta (**response**) el documento solicitado

Protocolo HTTP: Request-Response



Ejercicio 1:

Inspección de request-response en un navegador

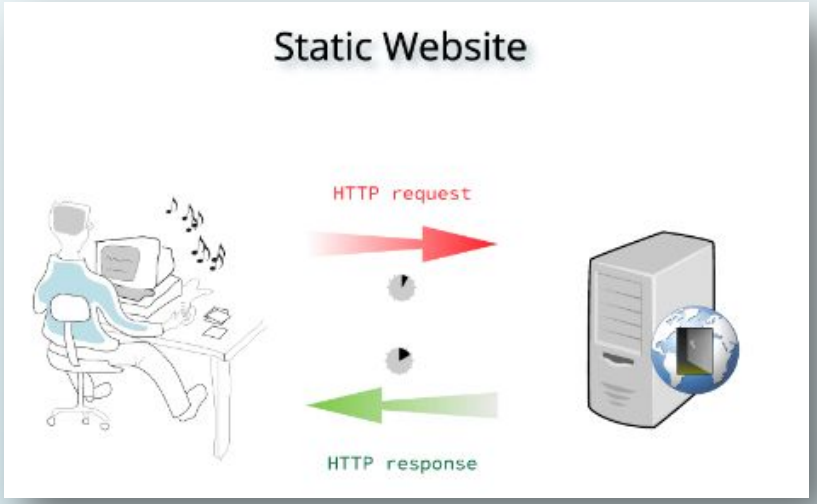


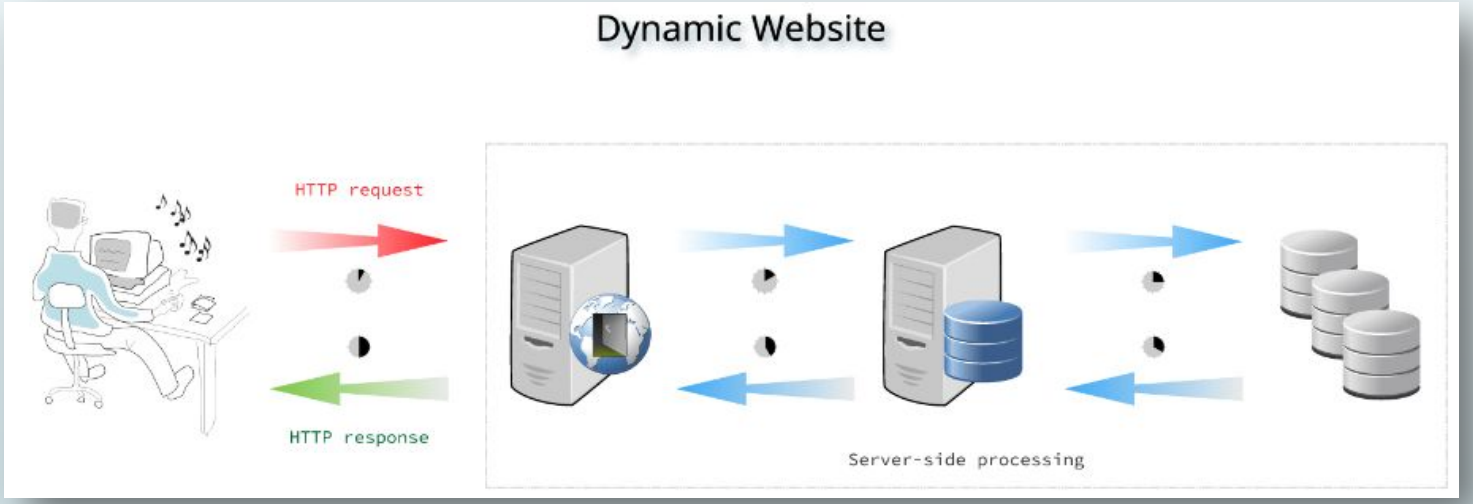
1. Abre un navegador web y accede a la wikipedia
2. Abre el inspector
3. Accede a la pestaña de red y refresca la página
4. Haz click en la última conexión, a la derecha podrás ver:
    1. Las cabeceras de las request, donde se indica el método (GET o POST), el protocolo, idioma y otras características
    2. Las cabeceras del response, donde se indica el tipo de documento enviado, fecha y otras características

Páginas web estáticas y dinámicas

- El lenguaje básico para la red es HTML. Una página escrita sólo con HTML y CSS será **estática**, es decir, mostrará siempre el mismo contenido.
- Cuando se utiliza un lenguaje de programación en el servidor, se pueden generar páginas en función de la solicitud del cliente. A estas páginas se les llama **dinámicas**.
- **Ejemplo** de página web dinámica: Una página web muestra distintos anuncios según donde esté el cliente que la visualiza.
- **Formas de hacer páginas dinámicas**:
  - Utilizar un lenguaje en servidor que genere contenido mediante una Base de Datos o servicios externos
  - Utilizar servicios de terceros invocados desde JavaScript
- **CGI (Common Gateway Interface)**: La petición del cliente se pasa a un ejecutable en servidor. Es este ejecutable el que genera la salida para que el servidor se la pase al cliente.
- **Las tecnologías utilizadas** para generar páginas web dinámicas son:
  - **Front end** (cliente): Navegador web y HTML + CSS + JavaScript
  - **Back end** (servidor): Servidor Web + BBDD y PHP, Python, Ruby, Java/JSP, .Net, .asp
- **Perfil full-stack**: En las ofertas de trabajo, cuando hablan de perfil full-stack se refieren a alguien que domine tanto el desarrollo front-end (cliente) como back-end (servidor)

Páginas web estáticas y dinámicas





Lenguajes de programación en servidor

- Existen multitud de lenguajes para desarrollo en servidor, los más habituales son:
  - **PHP**: Sin duda, el más extendido en el lado del servidor. Normalmente se ejecuta como un módulo del servidor. Pueden utilizarse frameworks como Laravel o Symfony, y lo utilizan en páginas como WordPress.
  - **Java**: Altamente portable, ofrece multitud de ventajas. Se utiliza con frameworks como Spring y Struts, embebido en HTML mediante tecnología **JSP** (API servlet). Lo utilizan páginas como LinkedIn y Yahoo.
  - **PERL**: Se utiliza mucho para CGI. Es un lenguaje especialmente pensado para el procesamiento de expresiones regulares.
  - **Ruby:** Es un lenguaje orientado a objetos muy apreciado por desarrolladores web. Se utiliza con frameworks como Ruby on Rails, y lo utilizan páginas como Airbnb y Shopify.
  - **Javascript** en lado de servidor: También puede utilizarse en el lado del servidor con frameworks como Next.js Lo utilizan Google, eBay y Facebook.
  - **Python**: Aunque es de propósito general y famoso para machine learning, dispone de una inmensa variedad de librerías para desarrollo back-end, como el framework Django. Lo utilizan Spotify y Pinterest.
  - **ASP.NET**: Es la alternativa Microsoft a PHP. Está integrada en la plataforma .NET
  - **Otros**: Golang, C#

Integración con lenguajes de marcas

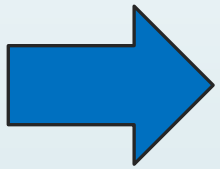
- Las páginas dinámicas se componen de:
  - Una parte estática en HTML
  - Una parte dinámica en algún lenguaje de programación
- Cuando se solicita una página, antes de enviarla el servidor busca en ella bloques de código . Si los encuentra los ejecuta y los sustituye por su salida.
- En PHP los bloques se delimitan por

```
<?php ... ?>
```

- Cuando el servidor recibe la petición de la página, ejecuta el bloque contenido por los delimitadores, y sustituye ese fragmento por la salida del programa.

Integración con lenguajes de marcas

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Número aleatorio</title>
5    </head>
6    <body>
7      <h1>Este número sale al azar...</h1>
8      <?php
9          echo rand(0,100);
.0     ?>
.1   </body>
.2 </html>
```

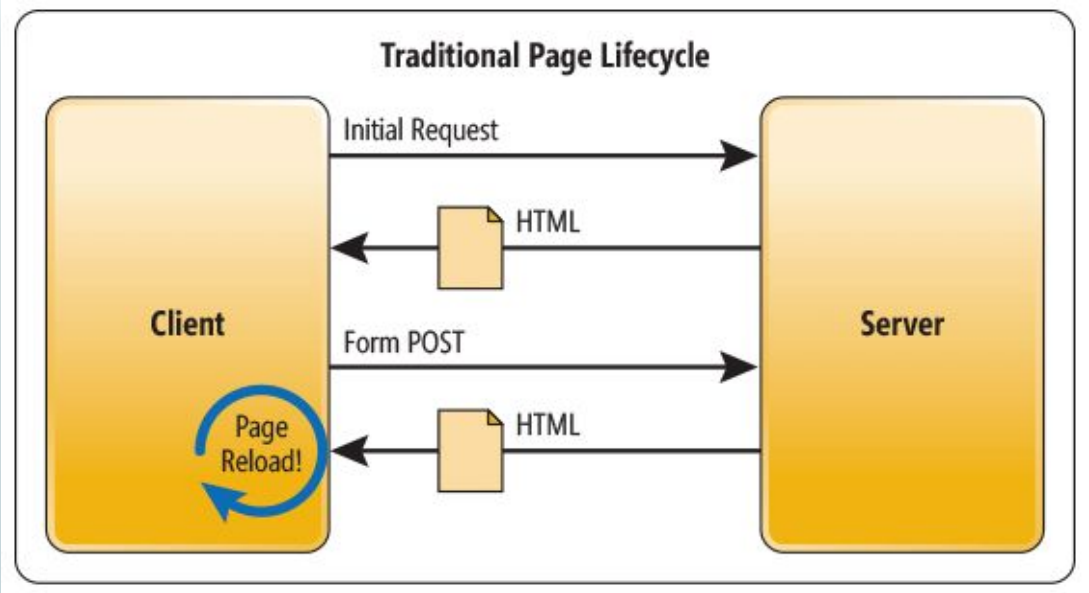


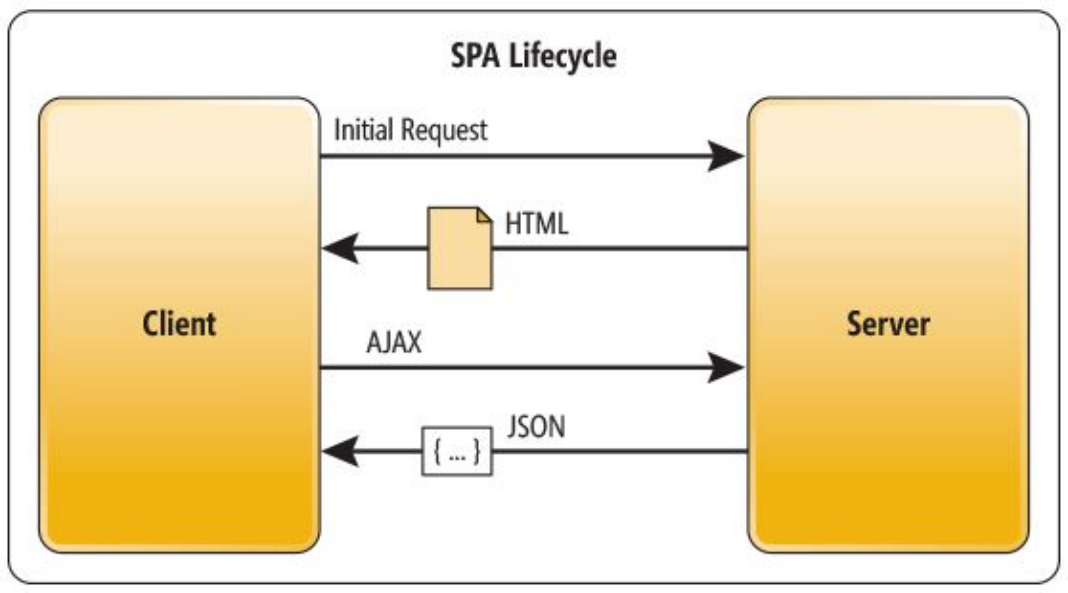
```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Número aleatorio</title>
5    </head>
6    <body>
7      <h1>Este número sale al azar...</h1>
8          33
9    </body>
10 </html>
11
```

SPA: Single page application

- La arquitectura web tradicional consistía en una llamada al back-end, que respondía con una página web HTML, que disponía links de acceso a otras páginas, que iban cargándose a petición.
- Actualmente se tiende a una arquitectura que proporciona mayor peso al cliente, la **SPA**: **Aplicación de página única**.
- SPA consiste en una sola página cargada en el cliente. El usuario interacciona con la página, y es el código javascript embebido el que realiza peticiones de forma dinámica.
- El código JavaScript cargará otros contenidos/páginas de forma dinámica durante la interacción con el usuario. A esto se le llama programación reactiva.
- Normalmente se accede a servicios remotos (**REST**), que realizan las operaciones comunicándose con **JSON**.

SPA: Single page application





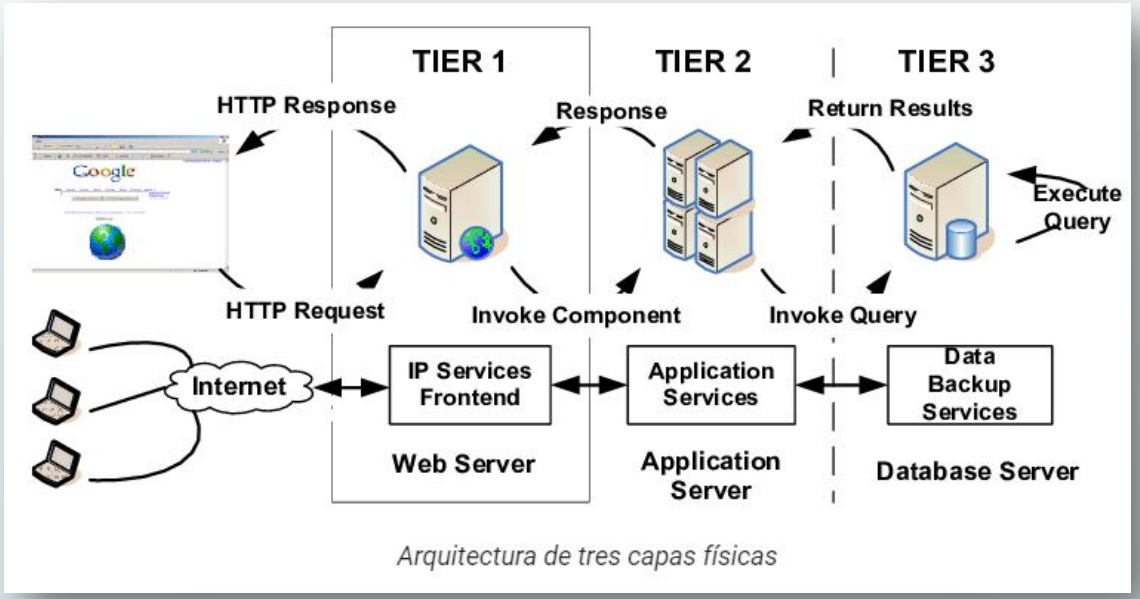
Arquitecturas y capas

- Cuando hablamos de capas, nos referimos a dos tipos:
  - Capa física o **tier:** HW.
  - Capa lógica o **layer**: SW
- **Capa física (tier)**: Elemento HW separado físicamente. Suelen estar protegidas por Firewall o VPN. Una capa física *NO* es un servidor. Puede corresponder con un cluster de servidores, que ofrece tolerancia a fallos y escalabilidad
- **Capa lógica (layer)**: Organizan el código por su funcionalidad. Cada capa puede implementarse en distinto lenguaje de programación

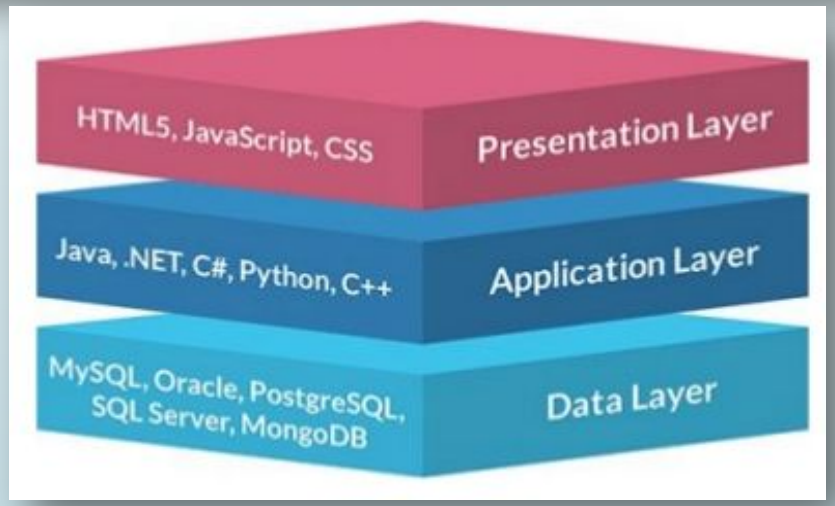
Arquitectura de 3 capas

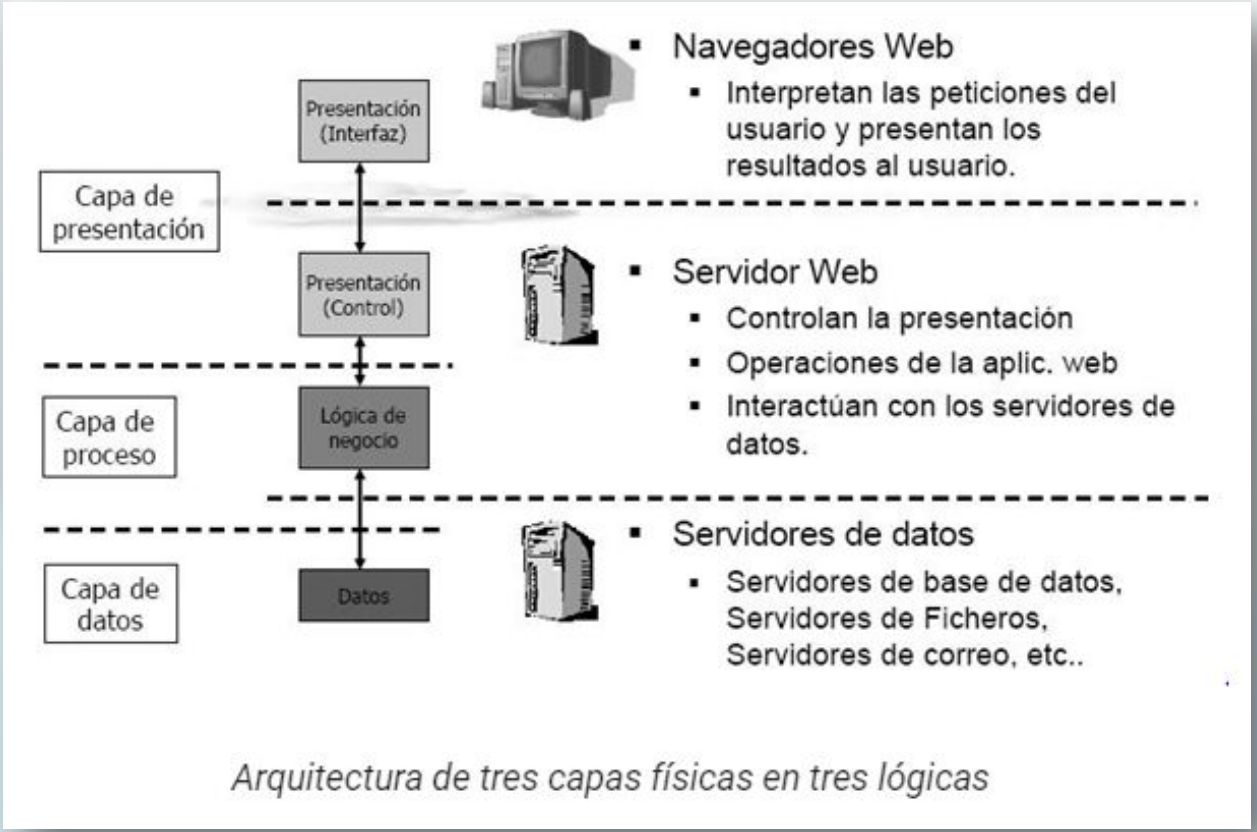
- Una arquitectura de 3 capas ofrece tres capas físicas y tres lógicas
- Capas físicas (tier):
  - Servidor Web
  - Servidor de Aplicaciones
  - Servidor de Base de Datos
- Capas lógicas (layer):
  - Software de presentación
  - Software de Negocio/Aplicación/Proceso
  - Software de Datos/Persistencia

Arquitectura de 3 capas



Arquitectura de tres capas físicas



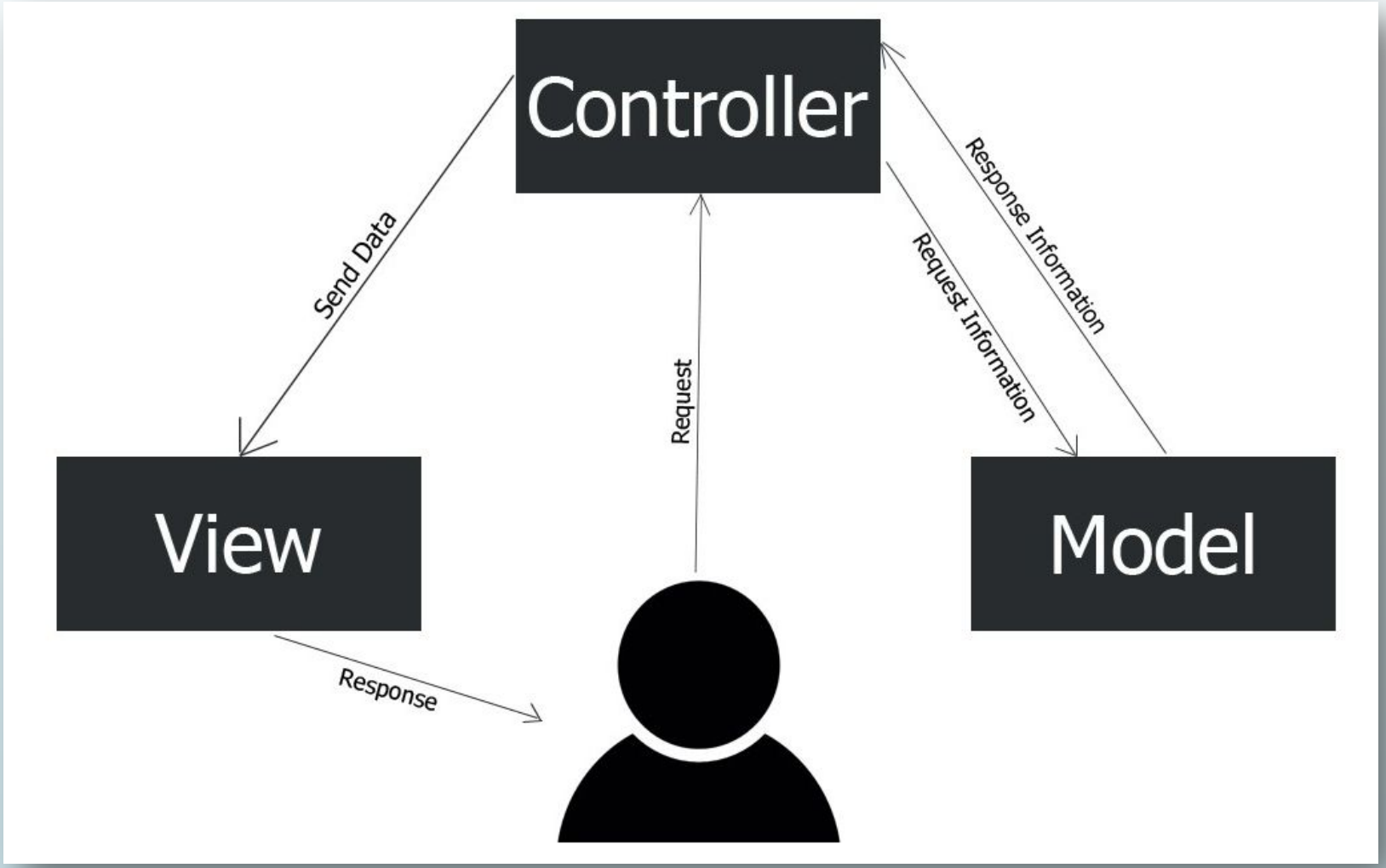


Arquitectura de tres capas físicas en tres lógicas

Arquitectura MVC (model-view-controller)

- La **arquitectura MVC** (modelo-vista-controlador): Es un modelo que separa en componentes conceptuales los datos almacenados, la presentación de la información y la lógica de negocio.
- Al separar el código en componentes conceptuales, facilita su reutilización y mantenimiento.
- **Modelo**: El componente "Modelo":
  - Representa la información y gestiona los accesos (queries, actualizaciones, etc).
  - Se accede al modelo mediante el controlador
- **Controlador**:
  - Responde a las acciones del usuario
  - Realiza peticiones al modelo para solicitar información
  - Envía la respuesta del modelo al componente vista para que gestione su visualización
- **Vista**:
  - Presenta al usuario de forma visual la información proporcionada por el controlador
  - El usuario interacciona con la vista, y realiza nuevas peticiones al controlador a través de ella

Arquitectura MVC (model-view-controller)



Decisiones de diseño

- A la hora de realizar un diseño para una aplicación web hay que tener en cuenta el tiempo y los recursos disponibles.
- Para determinar la planificación (recursos y tiempos), debemos preguntarnos lo siguiente:
  - ¿Qué tamaño tiene el proyecto?
  - ¿Qué lenguajes de programación conozco? ¿Vale la pena aprender uno nuevo?
  - ¿Voy a utilizar herramientas libres o propietarias? Si son comerciales, ¿cuál es el coste?
  - ¿Voy a programar sólo o como parte de un equipo?
  - ¿ Dispongo de servidor web y de BBDD disponible?
  - ¿Qué tipo de licencia aplicaré a la aplicación que desarrolle?

Servidor Web

- Un servidor web es un **software** que recibe peticiones.
- Las peticiones pueden ser de varios tipos: **GET**, **POST**, …
- Los protocolos utilizados son **HTTP** y **HTTPS**
- El servidor web más implantado es el **Apache Web Server** (https://httpd.apache.org/)
  - Se creó en 1995
  - Es un software libre y multiplataforma
  - Consta de módulos dinámicos
- Actualmente el servidor **ngingx** (https://www.nginx.com/) está ganando terreno, es un servidor web más moderno y en algunos escenarios ofrece mejor rendimiento.
  - Se creó en 2004
  - Es libre
  - Es escalable y admite alta concurrencia, hasta 500.000 peticiones por segundo con poco uso de CPU

Ejercicio 2:

Comparativa de Servidores Web

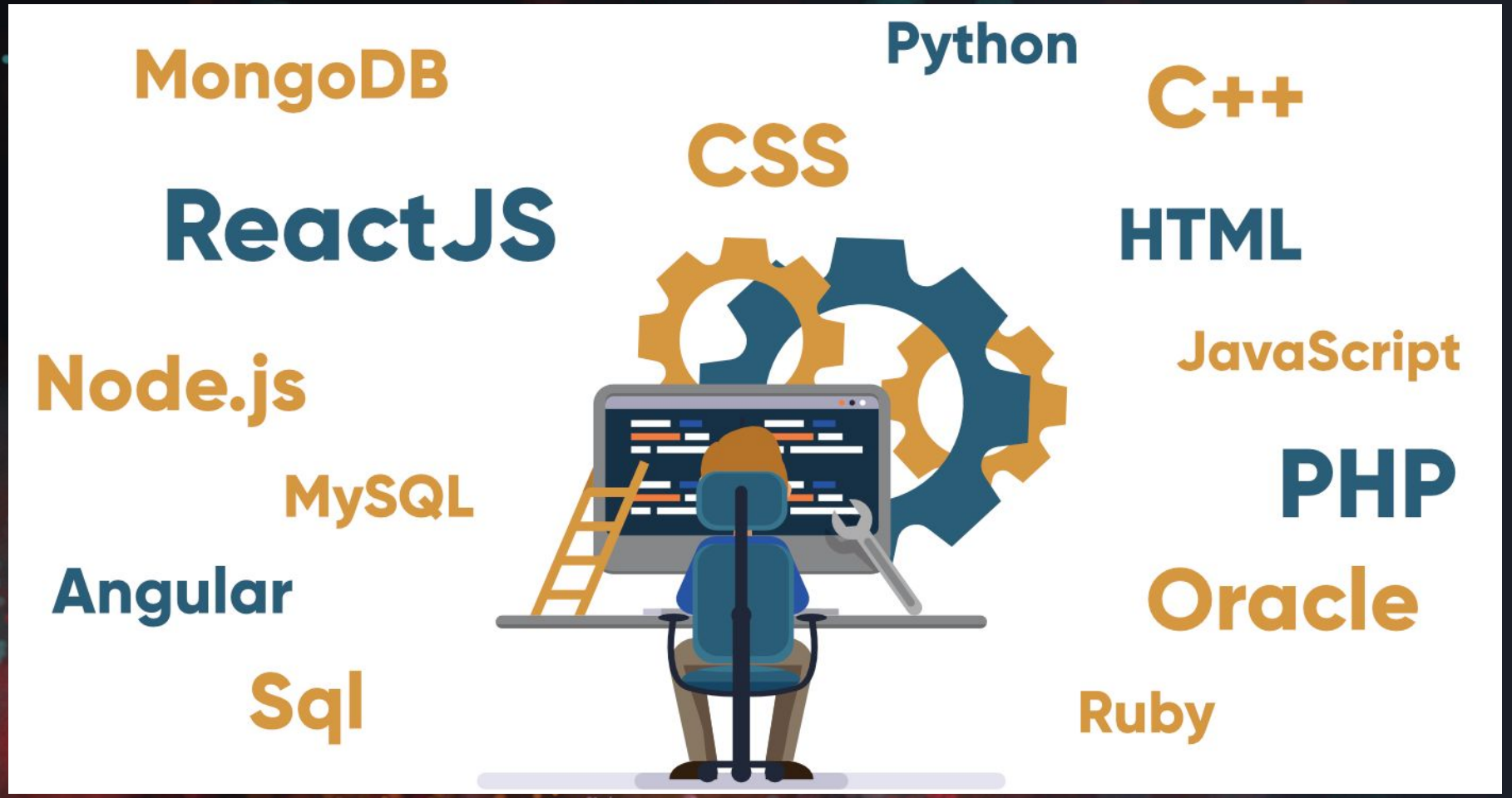
1. Accede a la siguiente página:

   `https://webhostinggeeks.com/best/web-server/`

2. Lee el artículo y elige las características más interesantes de un servidor web
3. Realiza un cuadro comparativo en una hoja de cálculo con las características de los servidores indicados

Servidor de Aplicaciones

- Un servidor de aplicaciones es un software que ofrece servicios adicionales a los de un servidor web:
  - Clustering
  - Balanceo de carga
  - Tolerancia a fallos
- Ofrecen un entorno de ejecución para las aplicaciones
- Su objetivo es liberar al programador de algunas tareas relacionadas con la infraestructura de la aplicación, como la seguridad o el balanceo de carga.
- Ejemplos:
  - **Tomcat** (http://tomcat.apache.org/): Servidor de aplicaciones opensource y multiplataforma de referencia para una arquitectura Java. Es un contenedor de Java servlets.
  - Servidores para aplicaciones JEE como **JBoss** y **Websphere** : Tienen contenedores para ejecutar los componentes JEE, como contenedores de servlets o persistencia.
  - **Ngingx**: Este servidor web ofrece servicios adicionales como los mencionados.



https://www.geeksforgeeks.org/what-is-full-stack-development