# **UD3**. ADMINISTRACIÓN DE SERVIDORES DE APLICACIONES

ies severo ochoa

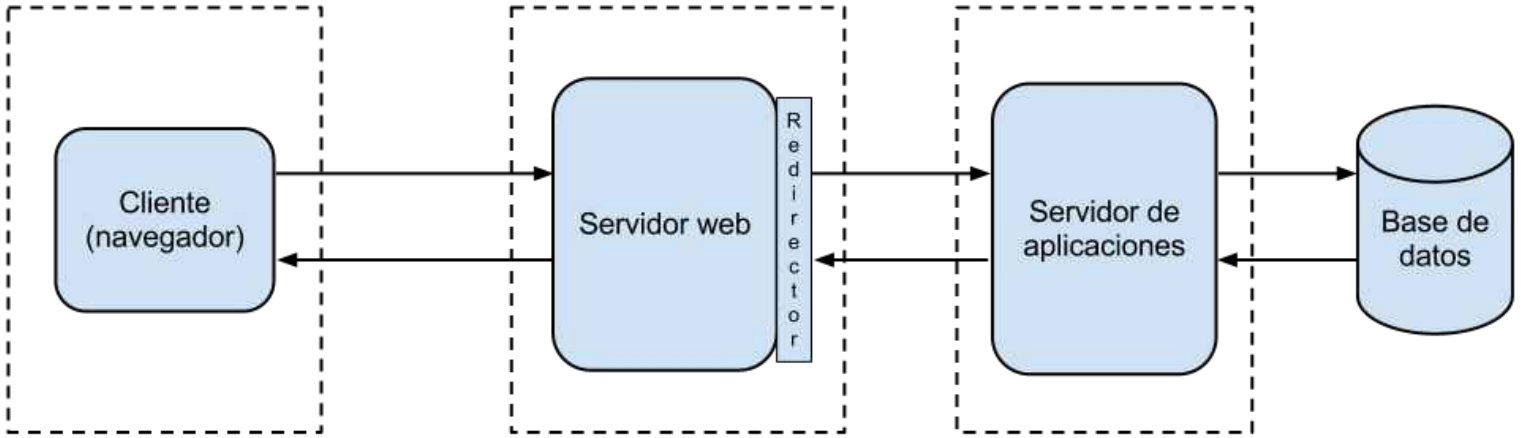
Despliegue de aplicaciones web

**2º DAW**

# 0. ÍNDICE

# 1. INTRODUCCIÓN

- El concepto de **servidor de aplicaciones** es posterior al de **servidor web**.
- Cuando aparecen las primeras **tecnologías** de generación de contenido **web dinámico** (CGI, PHP, ASP, JSP,...) aparece el concepto de servidor de aplicaciones.

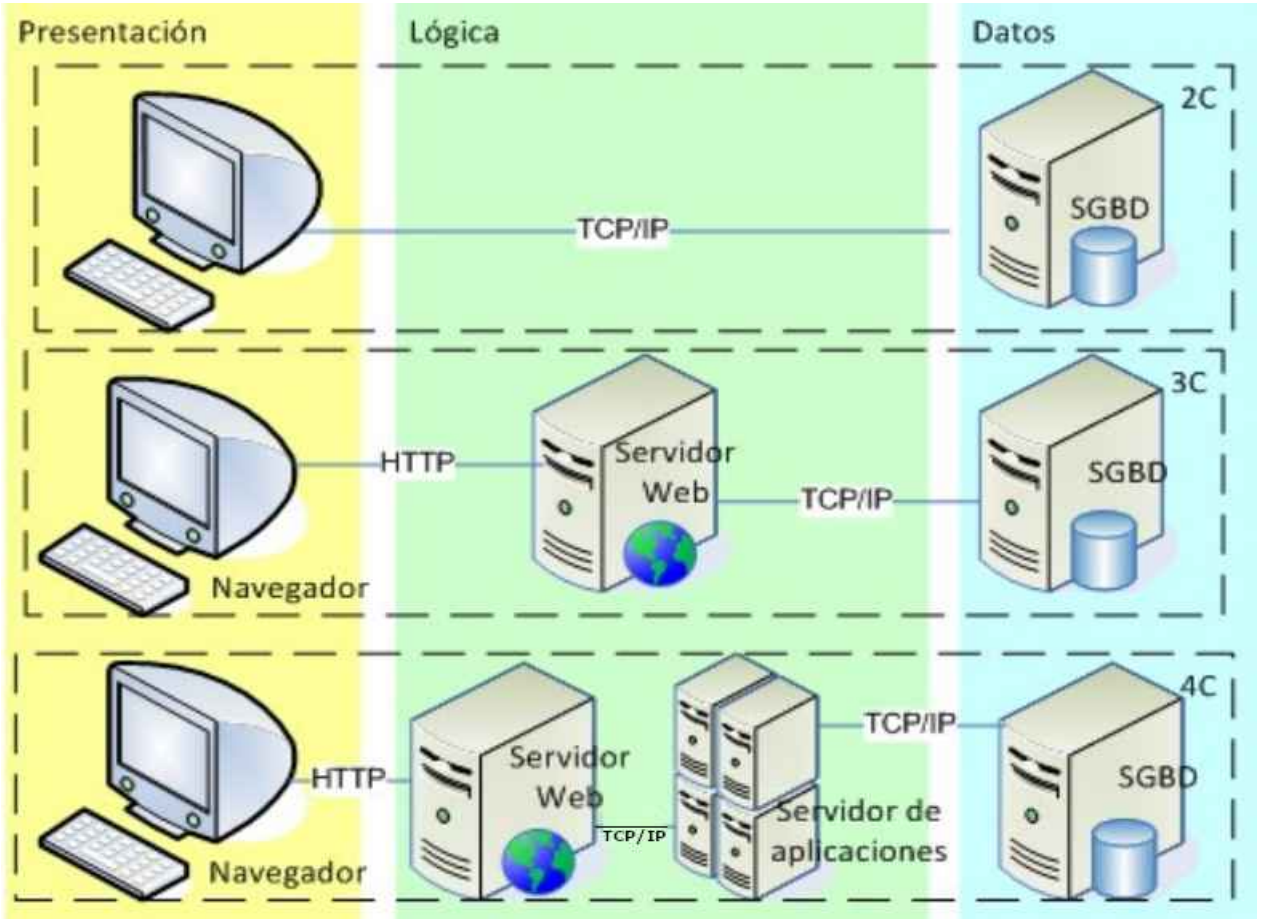


# 1. INTRODUCCIÓN

- Actualmente es difícil distinguir la frontera entre servidor web y servidor de aplicaciones. Podemos destacar las siguientes características de un servidor de aplicaciones:
  - Sistemas de autenticación (seguridad)
  - Gestión de sesiones de usuario
  - Acceso a los componentes o librerías de la plataforma utilizada (PHP, Java, .NET,...)
  - Gestión de las conexiones con servidores de bases de datos
  - En algunos casos implementan servicios como clustering, load-balancing o fail-over
  - Monitoriazación del servicio, gestión de procesos, estadísticas,...

# 1. INTRODUCCIÓN

- Evolución



# 2. SERVIDORES DE APLICACIONES

ies severo ochoa

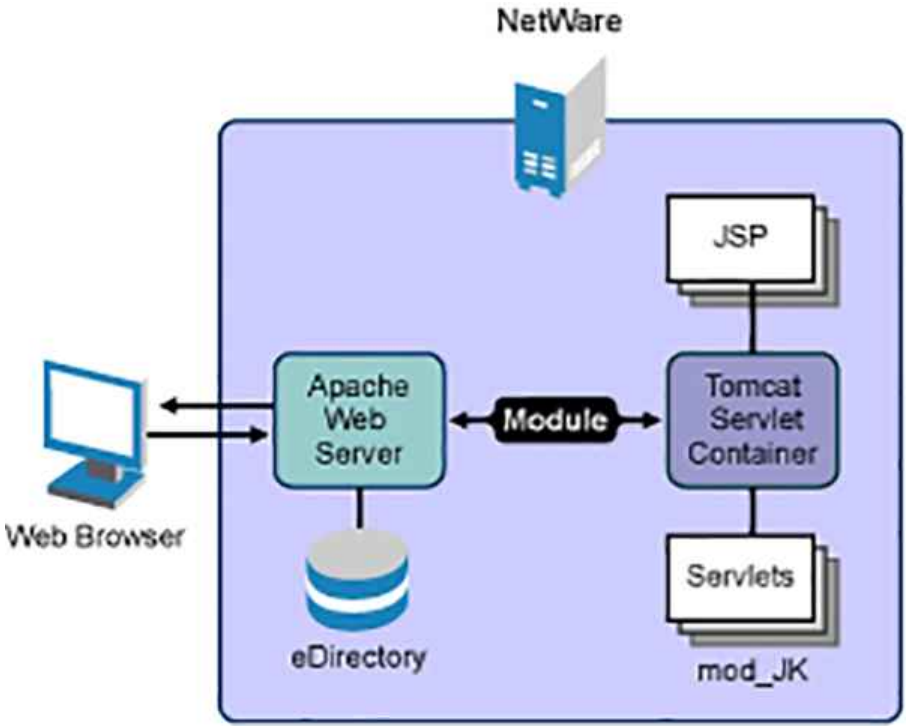
- Los servidores de aplicaciones están muy ligados a la tecnología utilizada para el desarrollo de la aplicación.
  - **ASP.Net:** Esta tecnología (.NET Framework), utiliza *ASP.NET core* junto al servidor **Internet Information Server (IIS).**


# 2. SERVIDORES DE APLICACIONES

- **JSP/Servlets:** La tecnología Java cuenta con diferentes servidores de aplicaciones que no necesitan la interacción con servidores web, como son:
  - **JBoss**: también llamado **WildFly**, se trata de un servidor de aplicaciones de código abierto basado en la especificación Java EE (Java Enterprise Edition) y escrito completamente en Java.
  - **Oracle (BEA) Weblogic:** Servidor de aplicaciones basado en Java EE y también servidor web http. (licencia propietaria de Oracle)
  - **Websphere application server**: Servidor de aplicaciones de **IBM** basado en Java EE.

# 2. SERVIDORES DE APLICACIONES

- **Apache tomcat:** integrado con el servidor web Apache, funcioan como un contenedor de servlets. Implementa las especificaiones de los *Servlets* y de *Java Server Pages* (JSP).



# 2. SERVIDORES DE APLICACIONES

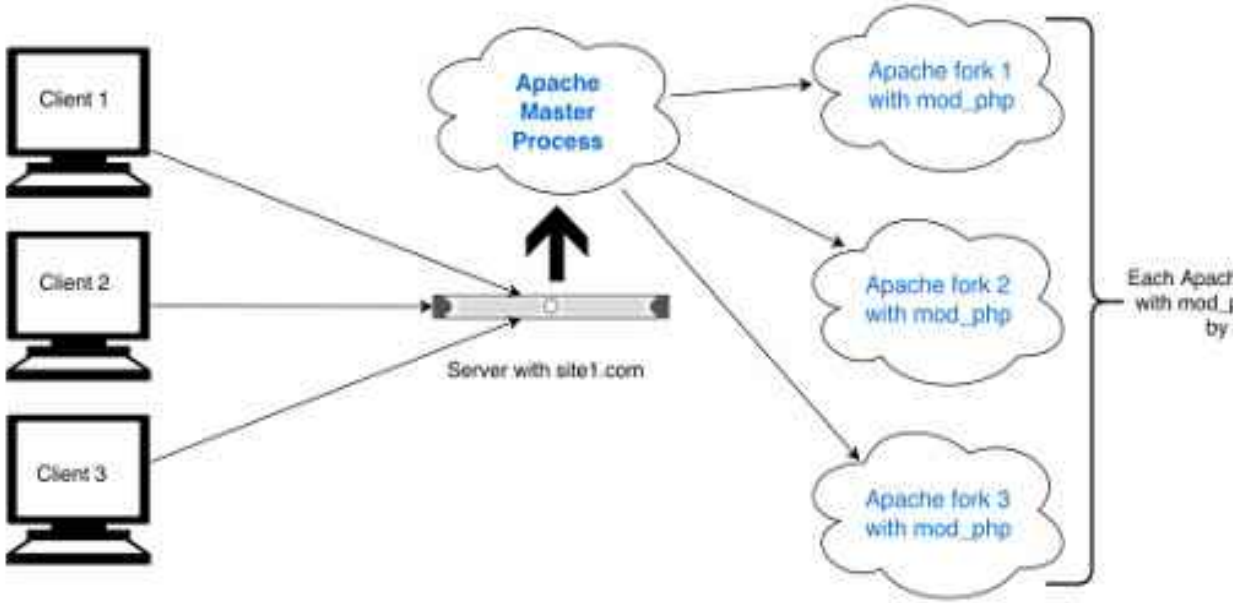
- **PHP**
  - La tecnología **php** hace uso de un servidor web, , generalmente **apache** o **Ngnix**, y de un **intérprete de php**. Podemos encontrar principalmente dos tipos de ejecución de las aplicaciones:
    4. **Apache** con la activación de un módulo interno **(mod_php)** que se encargará de la interpretación de las páginas PHP.
    5. **Apache / Ngnix** + servidor de aplicaciones **(phpFPM)**: El servidor web se encargará de atender todas las peticiones y confiará en un tercer servidor para la interpretación de las aplicacioens escritas en PHP.

# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Apache** con módulo interno: Se trata de un método más antiguo y rápido, siempre y cuando nuestro servidor no atienda muchas peticiones concurrentes.
    - El módulo de PHP ha de cargarse en cada petición
    - No podemos limitar ni restringir recursos para cada aplicación
    - No podemos aplicar diferentes configuraciones para cada aplicación
    - La seguridad de las diferentes aplicaciones se puede ver comprometida por las otras aplicaciones, ya que todas las aplicaciones se ejecutarán bajo el mismo usuario

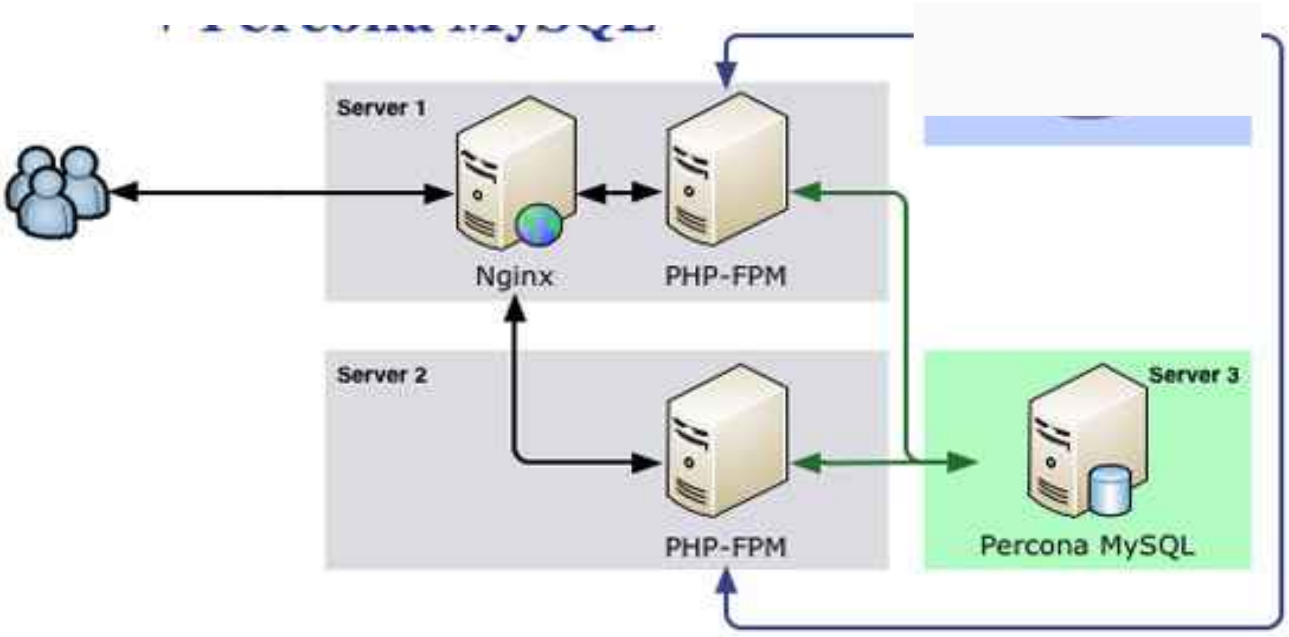
# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Apache** amb mòdul intern



# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Servidor web (Apache/nginx) + PHP-FPM**
    - El Servidor web y de aplicaciones son procesos independientes que pueden estar en el mismo host o en hosts diferentes *(Comunicación a través de la red)*



# 2. SERVIDORES DE APLICACIONES

ies severo ochoa

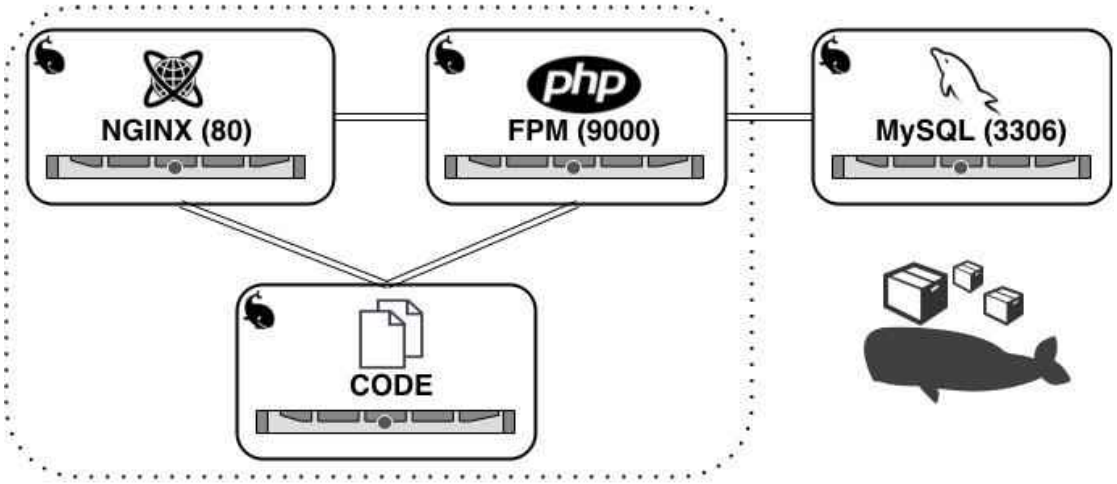
- **PHP**
  - **Servidor web (Apache/nginx) + PHP-FPM**
    - El **servidor web** se comunica con el servidor de aplicaciones mediante una versión mejorada de la tecnología **Fast CGI** *(Fast Common Gateway Interface).*
    - Se trata de un protocolo estándar que habilita la comunicación entre dos procesos.
    - *Independencia del lenguaje utilizado*
    - *Se utiliza un proceso separado*
    - *Es posible la ejecución en host separado*

# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Servidor web (Apache/nginx) + PHP-FPM**
    - ✔ El servicio de **PHP-FPM** permite establecer uno o más procesos persistentes, con cantidades específicas de recursos, que se mantienen a la espera de atender peticiones mediante un **socket**.
    - ✔ Permite establecer diferentes configuraciones
      - Puertos de escucha
      - Tipos de socket Unix/TCP.
      - Usuarios de ejecución
      - Módulos/Librerías de PHP a cargar.
      - Diferentes dominios de seguridad.

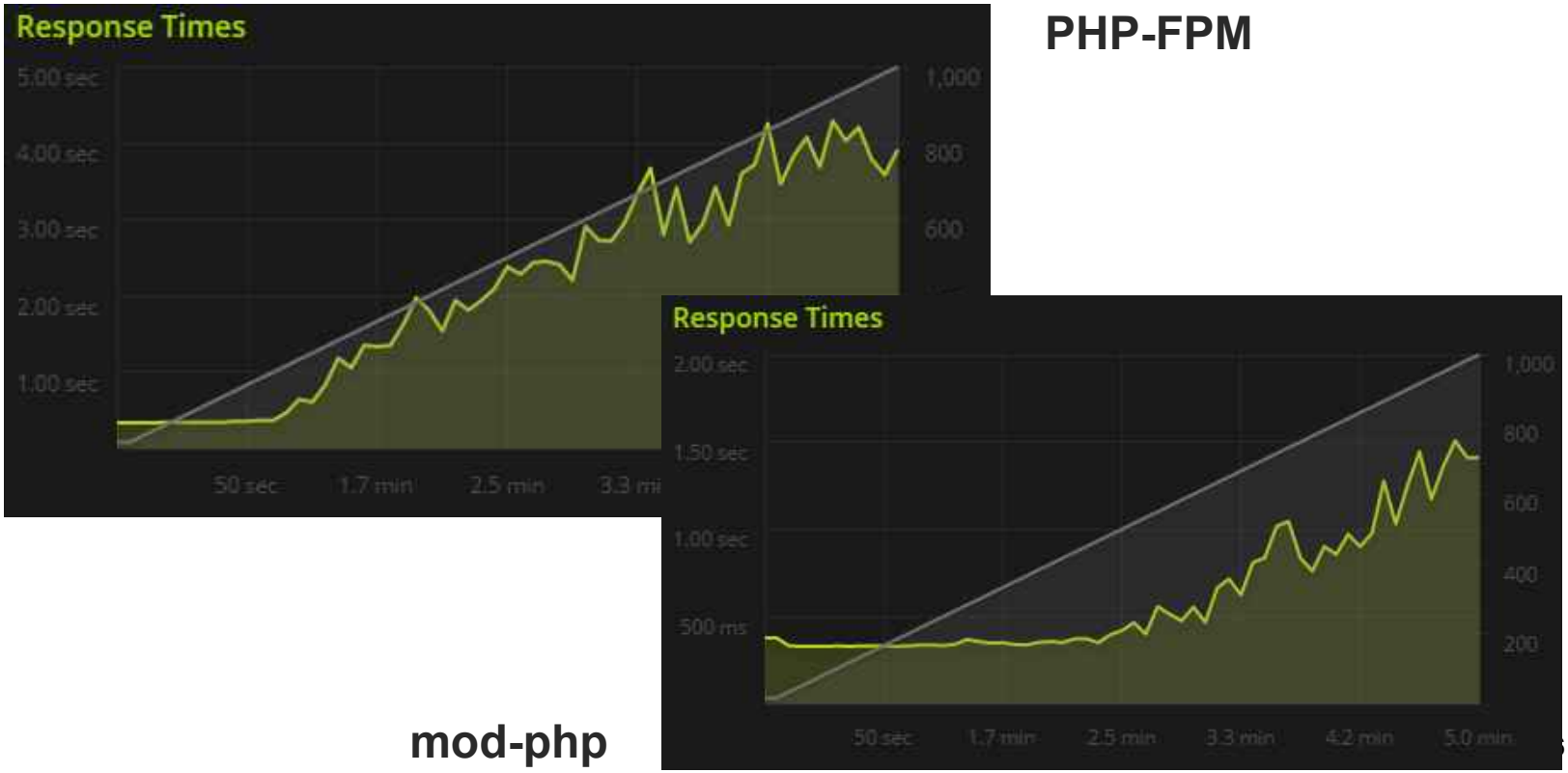
# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Servidor web (Apache/nginx) + PHP-FPM**
    - ✔ Permite mantener diferentes ficheros de log para cada aplicación
    - ✔ Monitorización y estadísticas de los procesos lanzados
    - ✔ Permite aplicar configuraciones y reiniciar el servicio de forma aislada



# 2. SERVIDORES DE APLICACIONES

- **PHP**
  - **Servidor web (Apache/nginx) + PHP-FPM**




**PHP-FPM**

**mod-php**

# 2. SERVIDORES DE APLICACIONES

- **Python**
  - Al igual que en el caso anterior, tenemos diferentes formas de ejecución de un script en python:
    - ✔ Mediante un módulo interno de apache:
      - **libapache2-mod-wsgi** → python 2
      - **libapache2-mod-wsgi-py3** → python 3
    - ✔ Mediante un servidor wsgi externo:
      - En este caso, el servidor web actúa como un Reverse proxy, que reenvía las peticiones al servidor wsgi **(gunicorn)**

https://www.digitalocean.com/community/tutorials/a-comparison-of-web-servers-for-python-based-web-applications


# 2.1 ENTORNO DE DESARROLLO

- La mayoría de frameworks y/o tecnologías de desarrollo web nos proporcionan un servidor de aplicaciones para el entorno de desarrollo:
  - El intérprete php → `php -S localhost:8001`
  - Symfony (php) → `Symfony Local Web Server`
  - Django (python) → `Django Web Server`

*"Estos servidores usan el intérprete del lenguaje disponible de forma global al sistema operativo.*
*Cualquier configuración sobre el intérprete se hará a nivel global"*

# 3. ARQUITECTURA APLICACIONES WEB. STACK

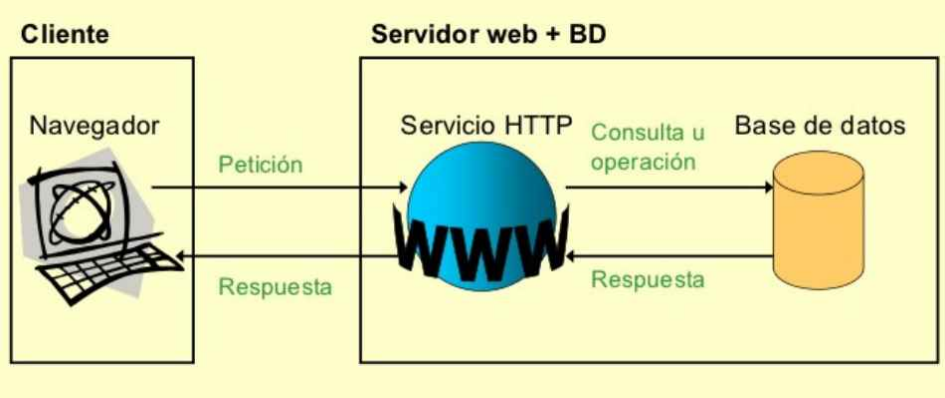
ies severo ochoa

- Llamamos "**Stack"** al conjunto de tecnologías escogidas como herramientas para la implementación de la solución de un proyecto.
- Generalmente se utilizan para dar una descripción de la arquitectura de una forma rápida. Se pueden destacar las siguientes:
  - LAMP (**L**inux - **A**pache - **M**ySQL - **P**HP)
  - LEMP (**L**inux - **E**ngine X - **M**ySQL - **P**HP)
  - MEAN (**M**ongo - **E**xpressJS - **A**ngular - **N**odeJS)

NGINX node.js mongoDB

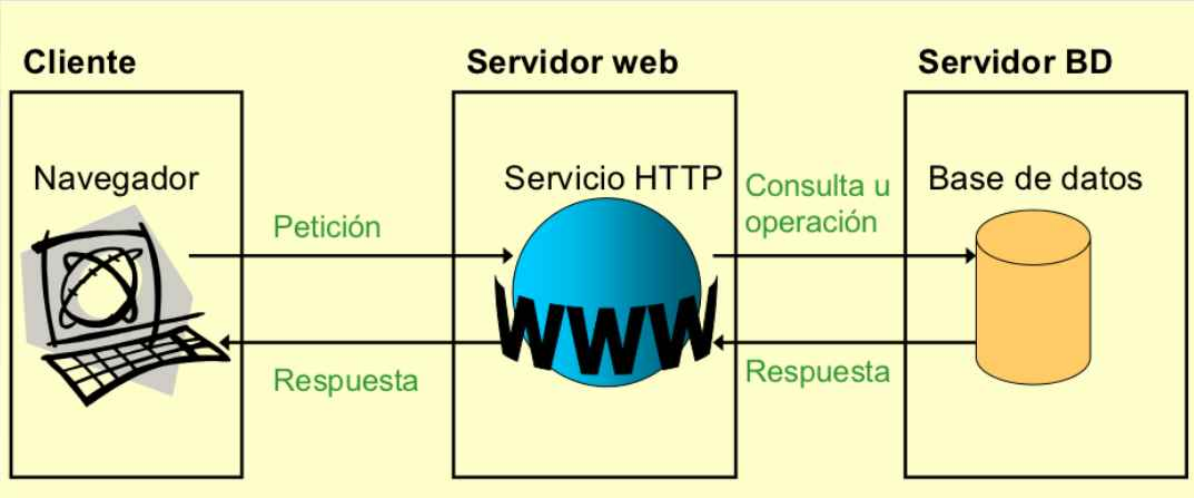
# 3. ARQUITECTURA APLICACIONES WEB. STACK

- Podemos encontrar diferentes distribuciones del stack en función del nivel de escalabilidad que necesitemos
  - Servidor de aplicaciones y base de datos en el **mismo Host**
    - **Más rápido**
    - No podemos **escalar** horitzontalmente el servicio.
    - **No** podemos **compartir el SGBD** con diferentes aplicaciones. (+ Gestión y mantenimento)
    - **No independencia** de **servicios**. (Seguridad / Failover)



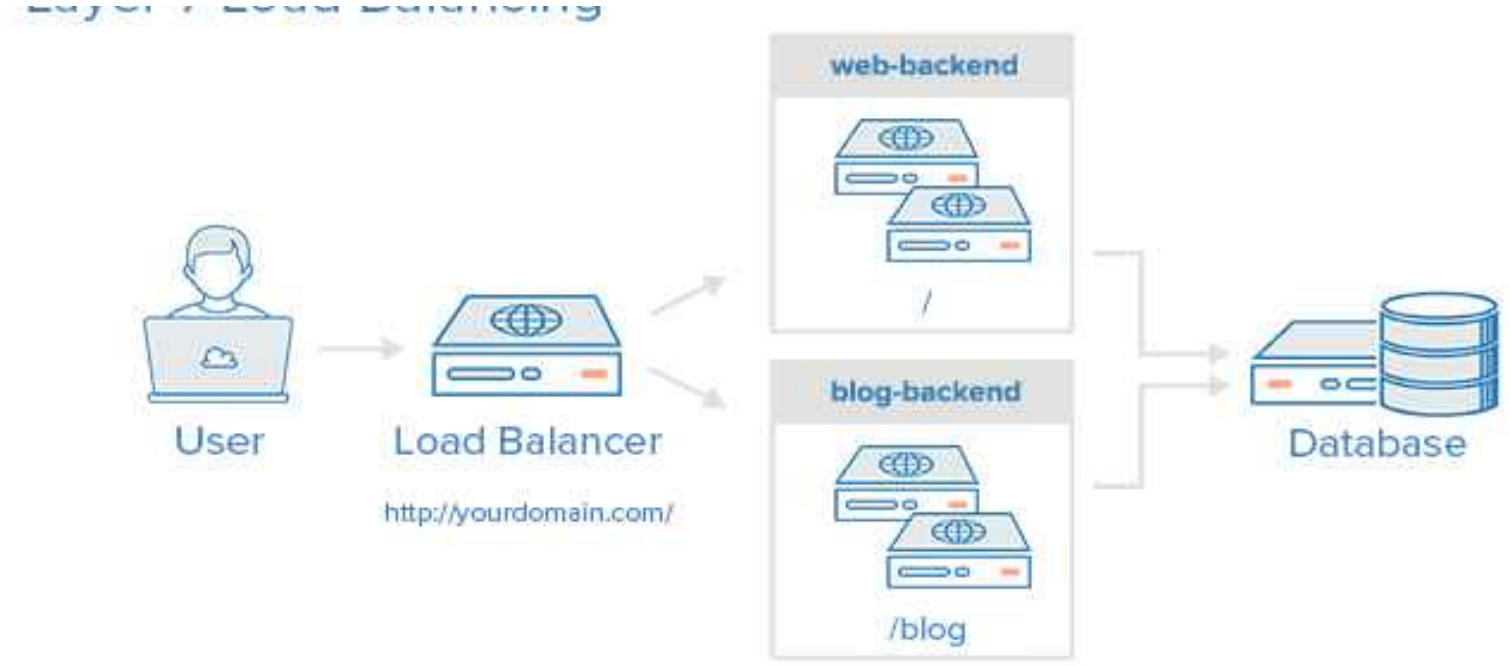
# 3. ARQUITECTURA APLICACIONES WEB. STACK

- Servidor de aplicaciones y base de datos en **diferentes Hosts**



# 3. ARQUITECTURA APLICACIONES WEB. STACK

- Servidor de aplicaciones y base de datos en **diferentes Hosts**



# 4. HTTP: PROTOCOLO SIN ESTADO

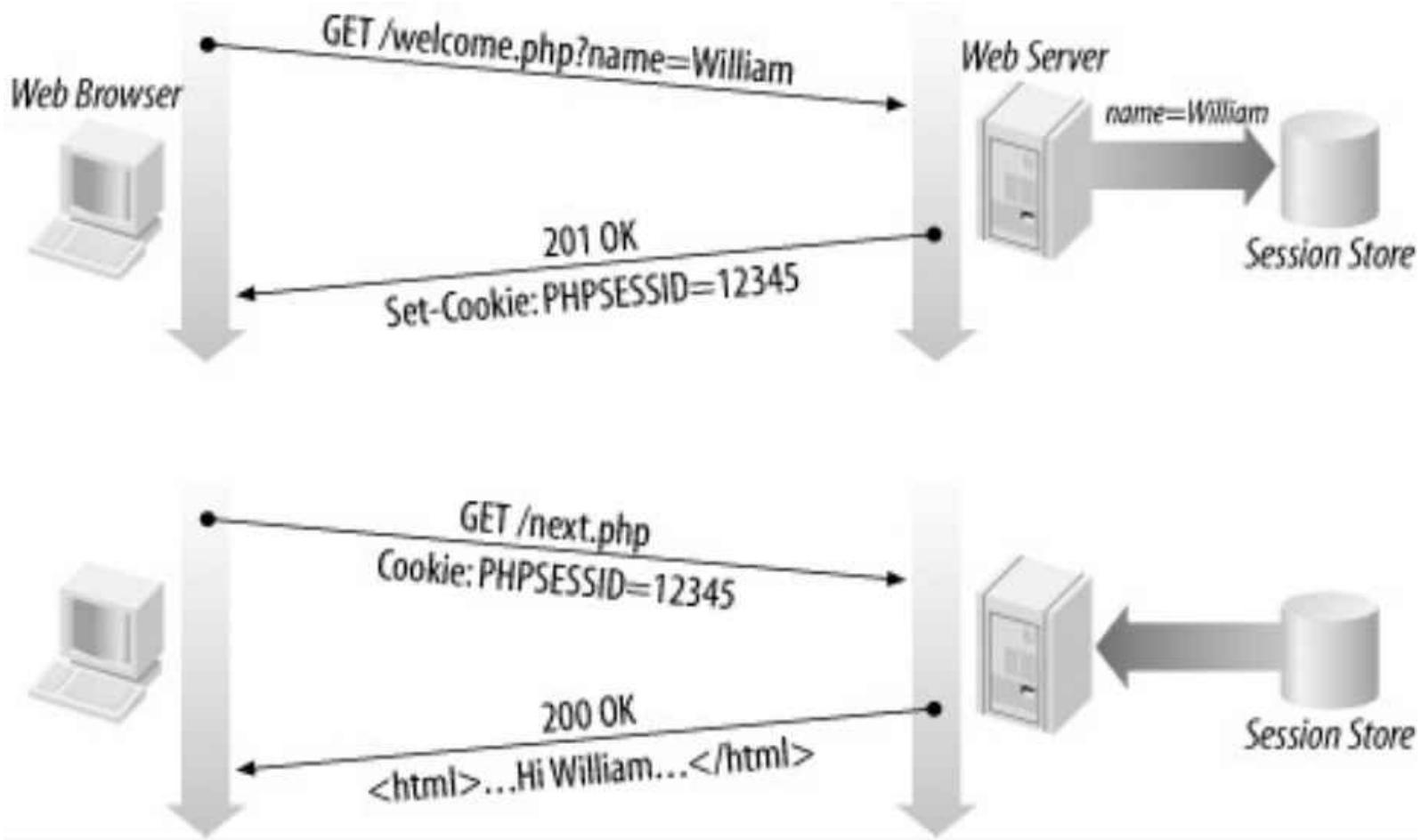
ies severo ochoa

- El protocolo **HTTP no** mantiene el **estado** entre **peticiones.**
- A veces, en el contexto de una aplicación web, se hace necesario mantener el estado entre peticiones HTTP que se producen entre cliente y servidor web. Por ejemplo:
  - **Saber** si un **usuario** ha hecho **"login"**.
  - Mantener **información** sobre acciones realizadas por el usuario.

# 4.1 GESTIÓN DE SESIONES

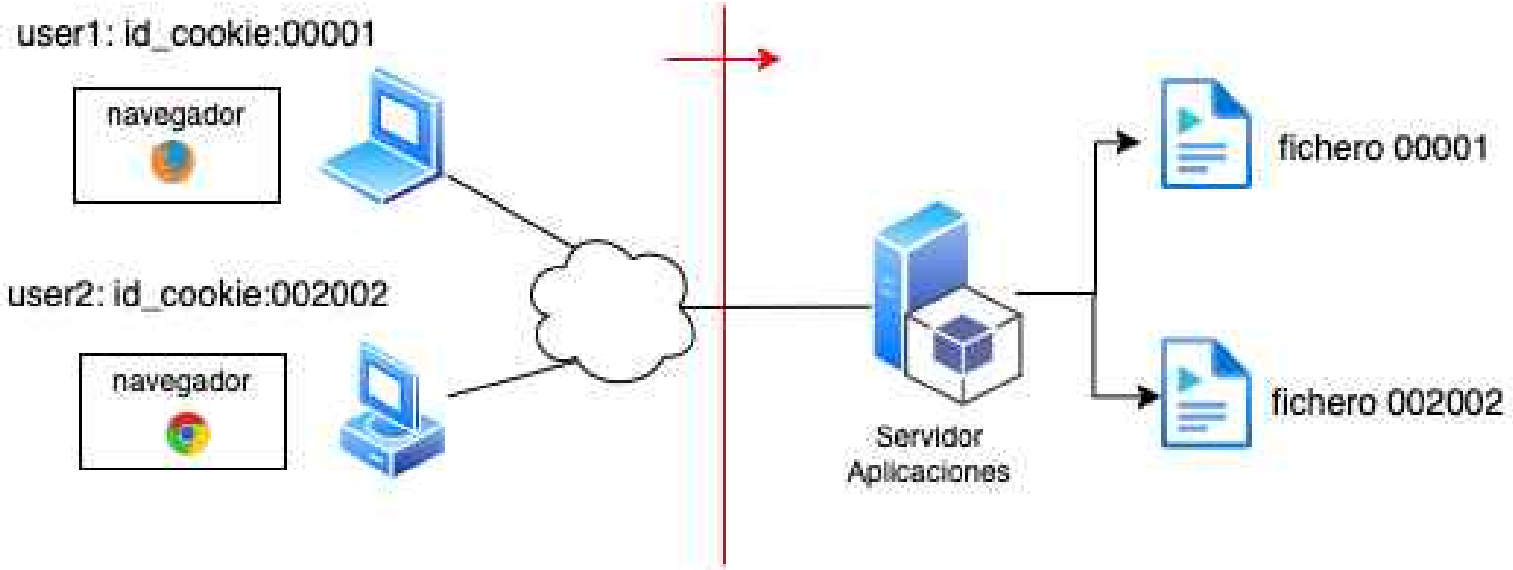
- Las sesiones son un mecanismo que utilizan los servidores web para guardar información sobre el usuario y su actividad.
- Una sesión permite establecer una comunicación entre el cliente y el servidor, manteniendo un estado (consistente en un conjungo de variables).
  - Se consigue asignando a cada cliente un **ID de sesión** al iniciar la comunicación.
  - Este **ID** ha de circular entre el client y el servidor en todas las peticiones HTTP.
    - Lo más habitual es **utilizar una cookie.**

# 4.1 GESTIÓN DE SESIONES



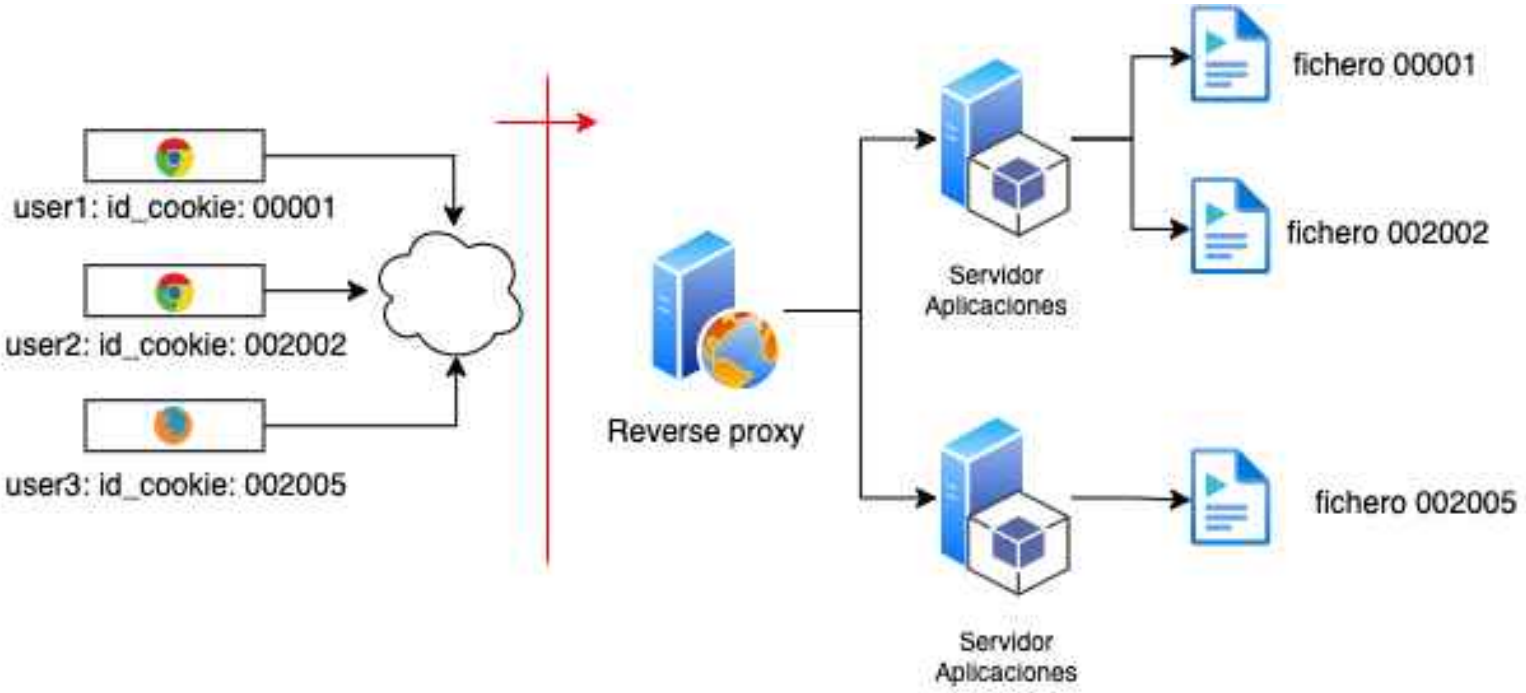
# 4.1 GESTIÓN DE SESIONES

- El servidor de aplicaciones asociará cada ID de sesión con un **archivo** en el servidor donde almacenará la información.
  - Su lectura penalizará el **rendimiento** del servidor.



# 4.1 GESTIÓN DE SESIONES

- Los archivos que guardan la información de las sesiones se han de gestionar (borrar cuando no son necesarios) y tenerlos en cuenta en el momento de escalar un servicio.



# 4.1 GESTIÓN DE SESIONES

- Se podrían guardar en un sistema de base de datos en memoria (Redis, memcache,...)
  - En el fichero **php.ini** se pueden configurar diferentes parámetros sobre el comportamiento de nuestro servidor con las sesiones.
  - También se puede hacer a nivel de aplicación.