Desarrollo de aplicación:

1- Análisis
2- Diseño
3- Implementación + pruebas y volvemos a análisis.

El framework organiza las carpetas y dispone de recursos que usaremos en una aplicación y tendrá utilidades como accesos a BD, formularios, etc.

El ciclo servidor (lo que ejecuta el servidor tras una petición y entregárselo al cliente) - red (medio de comunicación entre cliente y servidor con los protocolos de comunicación) - cliente (el navegador que solicita un recurso en una IP. Estilo JavaScript)

A veces el recurso no se entrega porque no existe. Si existe puede que no tenga permisos el cliente para recibirlo. Puede que el servidor no esté disponible en ese momento.

El recurso se especifica en la URL, formada por diferentes partes.
1- **Protocolos** . nivel de aplicación. http (puerto 80), https (puerto 443), ffp,etc. Apache, ginx es un programa es el servidor que implementa estos protocolos. **Ruta** donde en el servidor están los directorios y subdirectorios donde están los recursos. Para acceder a una parte se hará mediante clave valor separado por un ? que nos mostrará una parte en concreto.
2- Transporte. Protocolos TCP (seguro, enviar paquetes y hasta que no tenga el recibí no envía el otro, más lento pero seguro) y UDP(más rápido, envía todo aunque tenga errores, se usa cuando se envía imágenes o videos)
3- Red. Protocolo IP
4- Acceso al medio.

¿Es posible tener apache en un pc que no tiene tarjeta de RED/ Protocolo TCP/IP?
No es posible. Porque necesitamos la tarjeta de red para recoger la solicitud.

Ejercicio: Describir mediante diagramas un recurso en un navegador. Localizamos la máquina mediante IP, solicitud recogida en el recurso se pone en la red y se entrega al cliente.

Gestor Hugo (contenido + theme). Herramienta para sitios web. Permite crear una pagina web con archivos de configuración con clave - valor y carpetas como la content con archivos .md (makedown). Es importante conocer el formato md. Los títulos los pondremos con la almohadilla #. Con weight podremos ordenar los md.En la carpeta theme estarán las plantillas. Toda esta estructura de ficheros se compilará en un dir que luego subiremos a un hosting. UTIL PARA REALIZAR EL PRIMER PORTFOLIO.
http://manuel.infenlaces.com/dwes/hugo/

Los sitios estáticos son más rápidos pero no tiene una BD pero con API podemos ir a almacenar datos.

En linux para instalar HUGO pondremos sudo snap install hugo.
Con hugo new site nombreWeb creamos la página web.
TEMA

Instalamos github (repositorio y control de versiones). Instalamos el tema
https://github.com/McShelby/hugo-theme-relearn.git
con git clone https://github.com/McShelby/hugo-theme-relearn.git
Nos vamos al directorio del sitio web y ponemos las instrucción hugo server para visualizar el despliegue. Antes, tenemos que ir al directorio de sitio web e editar el theme en config.toml a elegir la plantilla

CONTENIDO
cd hugo
creamos una pagina con **hugo new site**
con git clone https://github…. clonamos el theme que queramos.
El archivo que configura todo config.toml para dar a los parámetros un valor.
Con **hugo server** lanzaremos el sitio web

Tendremos que saber el formato markdown. Tenemos que organizar el contenido en cada sección (directorio). Tendrá un index. Son páginas de sitio con contenido. Sirve para organizar
Crearemos tres secciones con un index en cada una de ellas.

Lenguaje markdown.

Para crear un fichero de texto usaremos hugo new. Para fichero de tipo índice usaremos **hugo new chapter – kind seccion1/_index.md** (creamos cada sección en el que colgaremos un índice, en weight pondremos un número entero) En draft (borrador) habrá que ponerlo en false para que se pueda visualizar.
hugo new –– kind chapter seccion1/_index.md

Para crear imagen entre parentesis pondremos el texto alternativo si no llega a cargar la imagen`![Clasificacion](/images/clasificacion.png)`

También se puede poner imágenes en la propia md haciendo referencia a la URL de la imágen.
Todo que esté en la carpeta static lo va a trasladar a la carpeta public.

GITHUB. CREAR UN REPOSITORIO NUEVO Y LIGARLO AL LOCAL

1. Accedemos a github con nuestro usuario y pass
2. En nuestra cuenta creamos un repositorio
3. Nos mostrará una serie de comandos que hay que ejecutar en local. Las acciones son:

- Vamos a local del directorio de nuestro proyecto
- Ejecutamos
- git init (este comando inicializa nuestro directorio como un proyecto de git. Habrá creado un directorio llamado .git)
- git add . (con este comando añadimos los ficheros de mi directorio (todos por especifica .) al repositorio
- git commit -m “mensaje de checkpoint”. Con este comando especificamos un mensaje para los cambios actuales

- git branch -M main (creamos una rama o directorio de trabajo asociados a el repositorio local)
- git add remote origin https://github.com/Riverwing (ligamos el repositorio local al repositorio remoto)
- git push origin main (sube los ficheros añadidos al remoto)

COPIAR OTRO PROYECTO EN OTRO DESTINO

- Accedo a github y copio la url del repositorio
- git clone URL (creará un directorio con el nombre del proyecto y todo el contenido)
- git add *
- git commit -m (nuevos cambios)
- git push -u origin main (con esto actualizará el proyecto con los últimos cambios, los subirá)

ACTUALIZANDO EL PROYECTO MODIFICANDO EN OTRO LUGAR

- git pull (Actualiza en el proyecto local con el remoto)

Pareja **Clave pública y privada** (para mi ordenador) para luego autentificarse. Son parejas de operaciones asimétricas (privacidad y autentificación). Subiremos la clave pública a git para que luego cuando accedemos ya desde nuestra privada podamos.

1. Vamos en la terminal a .ssh
2. ssh-keygen generará la pareja una vez que le damos Enter. Con ls -la veremos que se han creado los ficheros id_rsa e id_rsa.pub
3. Luego con gedit nano id_rsa.pub iremos a github a SSH keys le pondremos un título y copiaremos el contenido del documento en Key.
4. Después iremos a repositorio y cambiaremos de HTTPS a SSH (protocolo cifrado seguro).

Shortcode. Es Html que vamos a incluir

```
{{ <youtube j9WyKTGAO2w>}}
{{% attachments sort="asc" /%}}
```

Podremos

PHP DOCKER