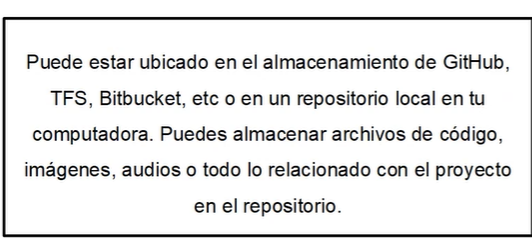
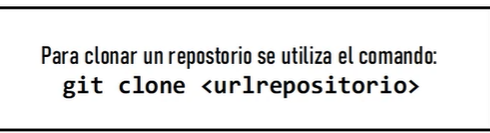
**Se usa con visual estudio y TFS.**

**GIT** - sistema de control de versiones, todo tipo de archivos. no solo codigo, mantener registro de cambios y versiones. GIT es distribuido no es centralizado.

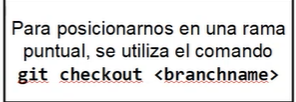
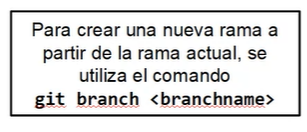


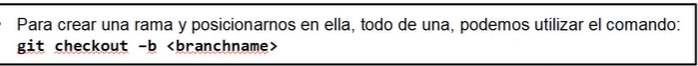
TFS codigo repositorios

REPOSITORIOS CLONAR: generar copia de manera local, ORIGIN es el nombre std.



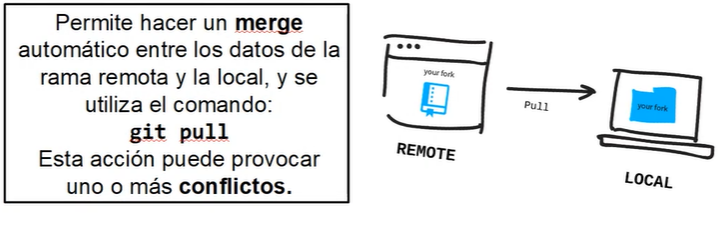
RAMA O BRANCH, puntero que señala una versión del código, cuando clonas se abre una rama que se llama MASTER que es la principal y es la que parten las otras ramas, y sirve para tener código que está en producción.

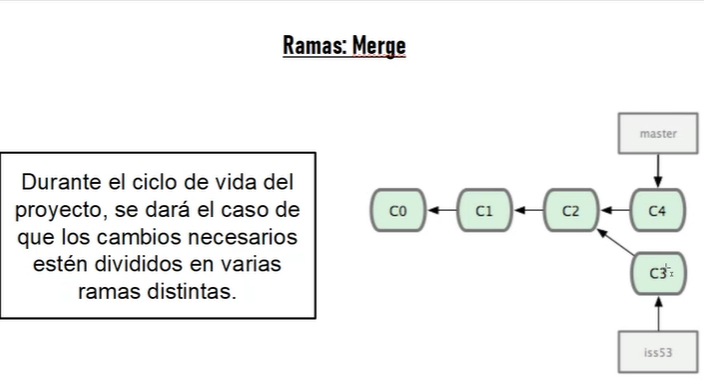


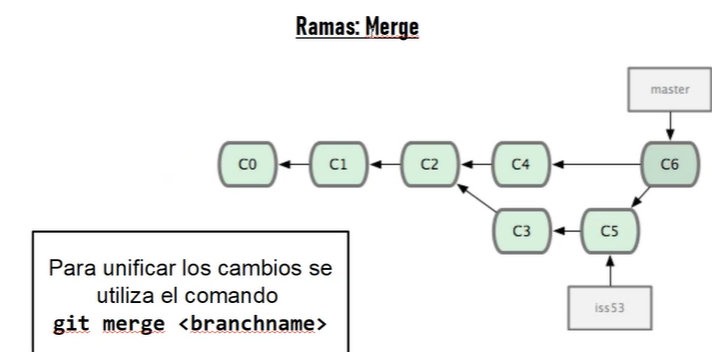


una rama se usa para hacer desarrollos paralelos de manera independiente, y luego se van integrando, en el ciclo de vida el desarrollo lo vas a tener en varias ramas. El merge unifica dos ramas

MERGE entre rama local y rama remota

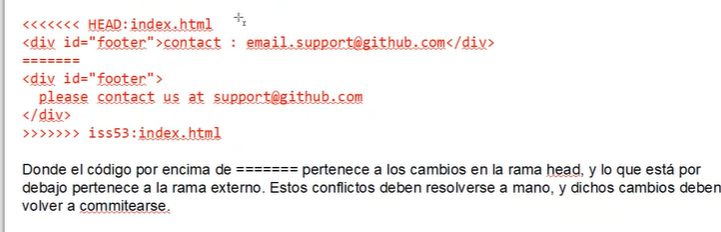




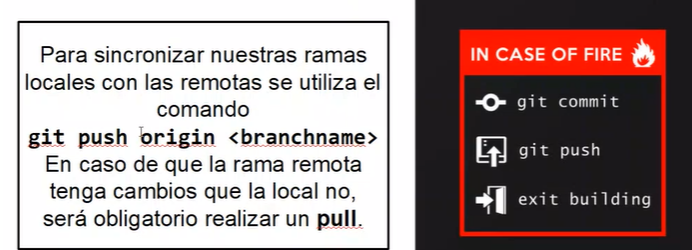


si estamos posicionados en MASTER se hace git merge (iss53)pasa los cambios de iss53 a MASTER.

si en ambas ramas se modifico una misma linea de codigo git no sabe fucionar y genera un conflicto. se resulve de forma manual



Las ramas no están remoto, sino local entonces hay que pushear. Tambien es para actualizar tablas existentes.



**FLUJO DE TRABAJO GIT**

los flujos son

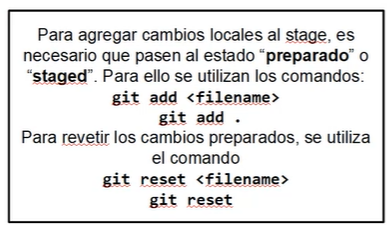
**working directory- trabajo local**

primer lugar donde trabajamos copia local del proyecto.

**stage- preparado**

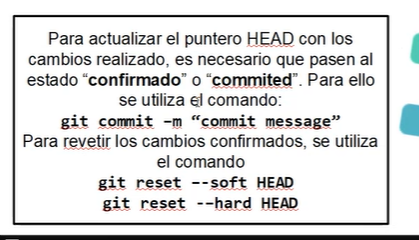
cuando quiero que mis cambios se guarden de forma temporal pero sin confirmar,

no esta disponible en el repositorio local, git add . ( punto) es todo



head- ya están en repositorio local

apunta a la rama que estamos trabajando

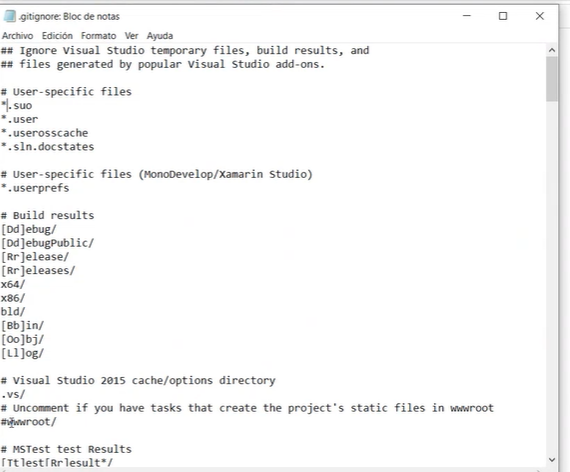


buena práctica dentro del msj comit explicar que se hizo

**soft** nos revierte confirmacion pero guarda cambios

**hard** nos revierte y borra todo





**MVC- PATRÓN DE DISEÑO ARQUITECTONICO- modelo vista controlador.**

separa en tres capas los patrones se utiliza en frameworks como es angular

**vista** representacion visual al usuario. html java scrip, representamos mediante una grilla, imputs, como presentamos la informacion.

**modelos** independiente de la vista,

**controlador** interaccion entre modelo y grilla, recibe los eventos del usuario, y conoce como recupera los datos y se los pasa a la vista para que los presente

organizamos codigo en capas, mas facil de mantener y de escalar.

accesible a la modificacion.

Para cualquier lenguaje

ASP.NET MVC (C# , VBasic)

Laravel (PHP)

django (Python)

Ruby on Rails

El usuario solicita una acción al servidor

El servidor atiende la petición y manda a llamar al controlador

El controlador llama al modelo necesario

El modelo atiende la petición y realiza las operaciones de datos correspondientes

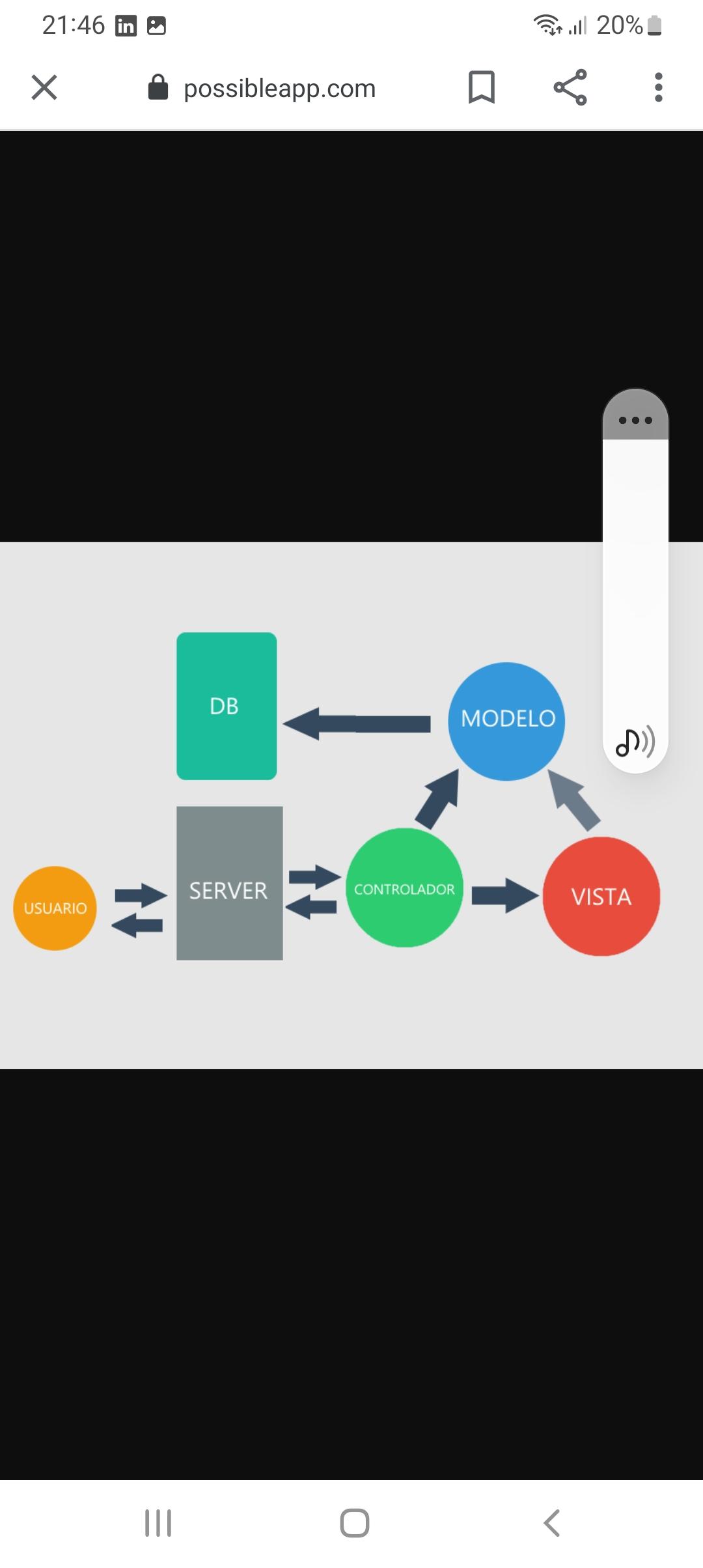
El modelo regresa el resultado

El controlador llama a la vista, enviándole los datos procesados del modelo

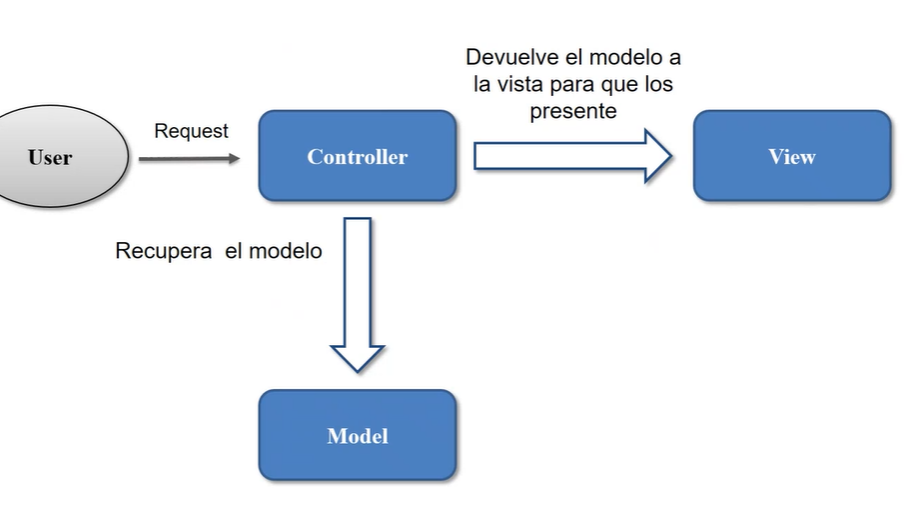
La vista presenta los datos

El controlador devuelve la vista al servidor

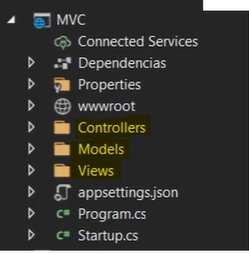
El servidor presenta el resultado al cliente



ASP.NET CORE MVC

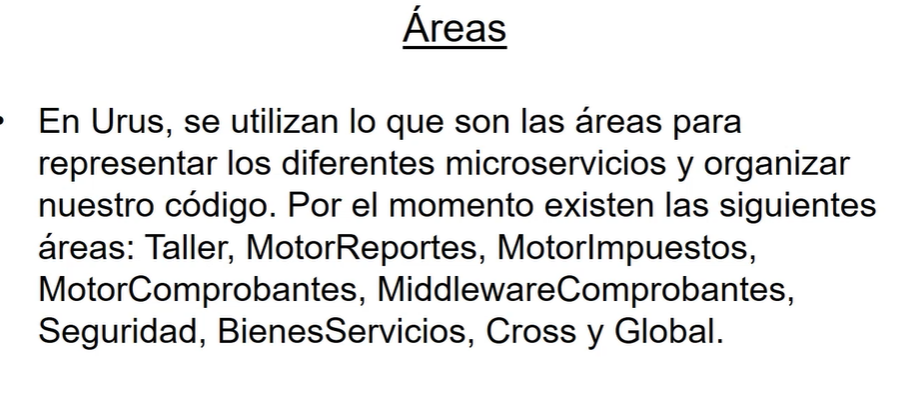


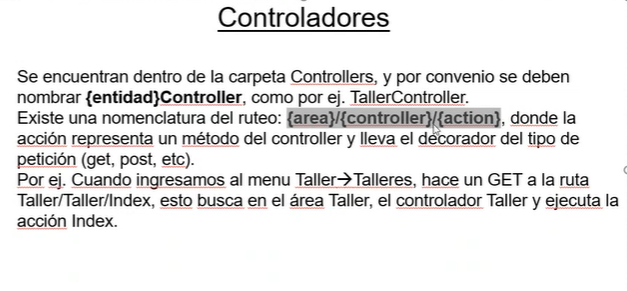
genera un proyecto con estructura



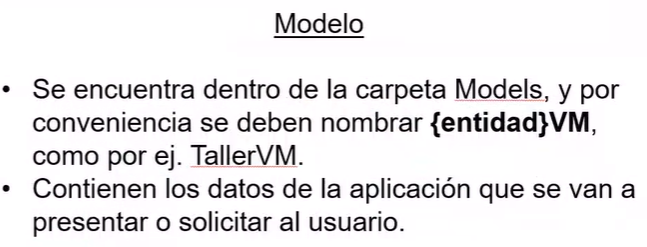
wwwroot guardan los archivos ccs las imagenes java scrip

microservicios- areas- carpetas que esta dividido el frontent



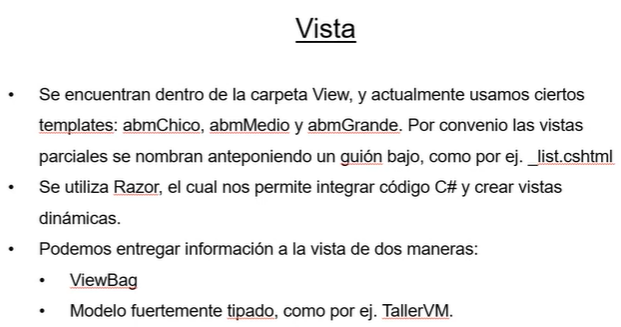


**MODELOS** todos los modelos se finalizan con vm



**VISTAS** Un fichero cshtml, es un fichero que le permite añadir código c# dentro del mercado del HTML utiliza Razor.

genera vistas dinámicas. ABM chico -mediano-grande.



dos formas se vincula la vista con modelo:

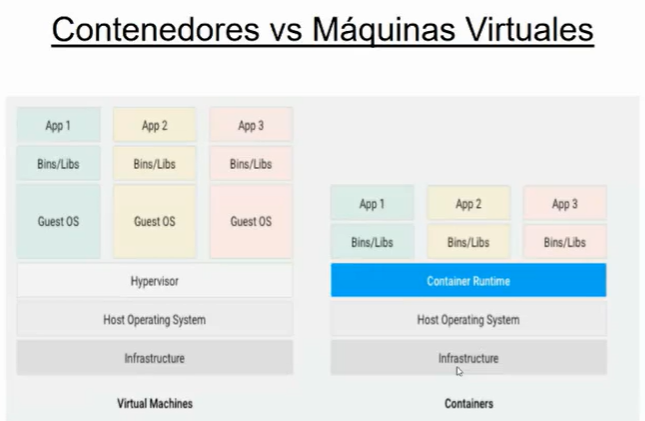
1. sin angular: no sobreescribimos elemento a guardar, ejecuta por atras en wroot funciones que ya estan.
2. con angular: template ABM grande ya vienen los métodos de angular.

**DOCKER**

todo lo netcore que corre sobre linuk pueden

1. CONTENEDOR caja donde tengo código fuente, lenguaje, todo.

unidad de software estandarizado, compararlo con una maquina virtual pero funciona distinto más optimizada



La máquina virtual es un sistema operativo sobre otro, y el tamaño son gigas.

en el contenedor comparten sistema operativo y son megas.

DOCKER ENGINE- producto

tecnología que nos permite correr contendores. arquitectura cliente - servdor. se expone una api

**Conceptos básicos:**

**imagenes:** dockerfile todas las instrucciones a partir del cual se crea el contenedor, <https://hub.docker.com/> se bajan.

en c:

docker pull baja la imagen te copias del hub la linea de codigo.

docker images para ver la imagenes

docker rmi para borrar imágenes.

para imagenes publica, para crear propias imagenes tengo que crear mi docker file