

# taller de diseño



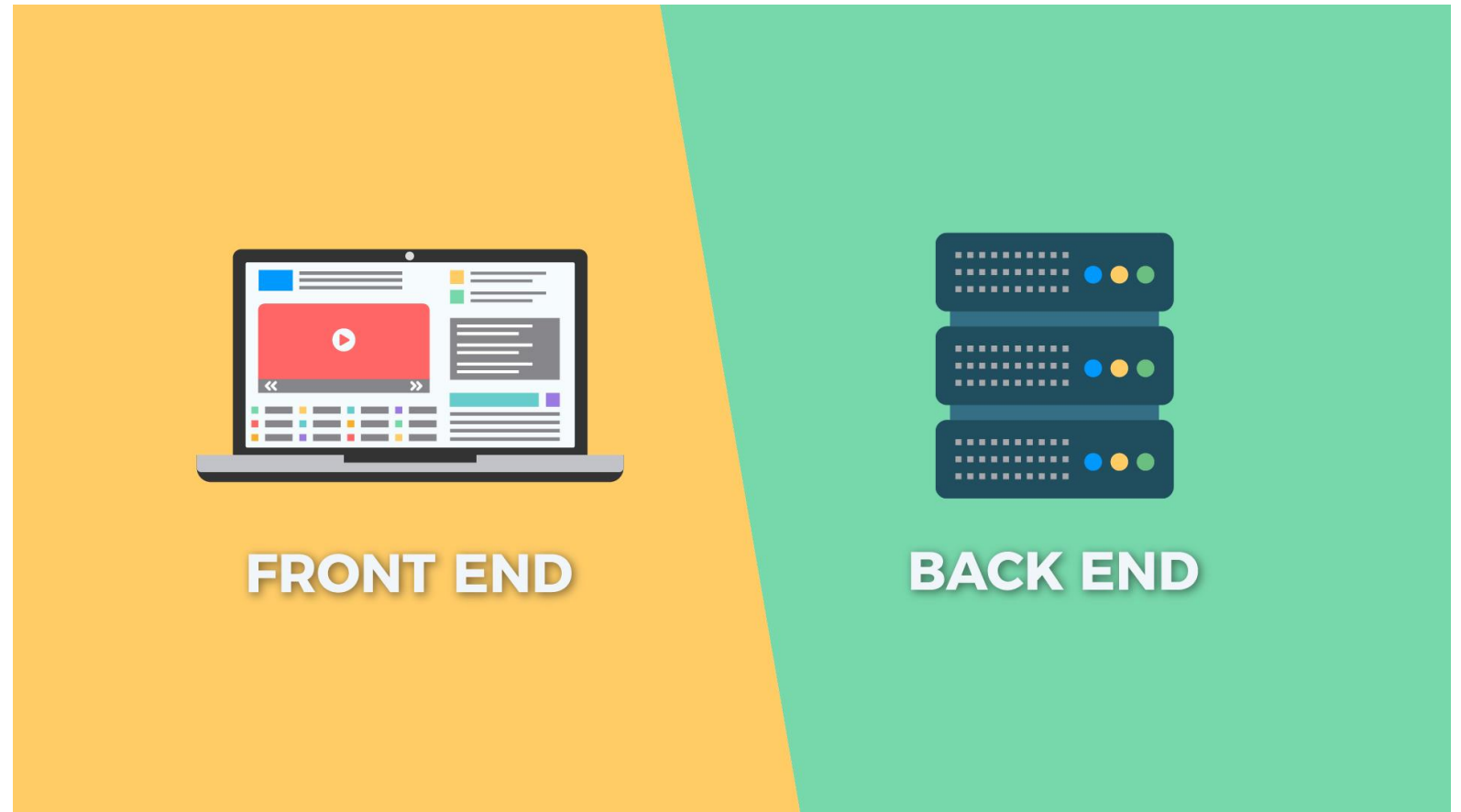


# Arquitectura de Aplicaciones

# Arquitectura de Aplicaciones

Patrones y Técnicas que se utilizan para diseñar y desarrollar aplicaciones.

Proporciona un plan y buenas prácticas.



El desarrollo de **frontend** se refiere a la experiencia del usuario con la aplicación

El desarrollo de **backend** implica proporcionar acceso a los datos, los servicios y otros sistemas actuales que permiten el funcionamiento de la aplicación

# Primer Paso

Definir los objetivos estratégicos.



Frecuencia con la que desea lanzar las actualizaciones



Facilidades para el desarrollo de software



Capacidad para brindar nuevos servicios y funciones



Cómo escalar en caso de aumento de usuarios o intensidad de uso

# Arquitectura de Software MVC

Su fundamento se basa en separar el código en tres capas diferentes, acotadas por su responsabilidad

Modelo

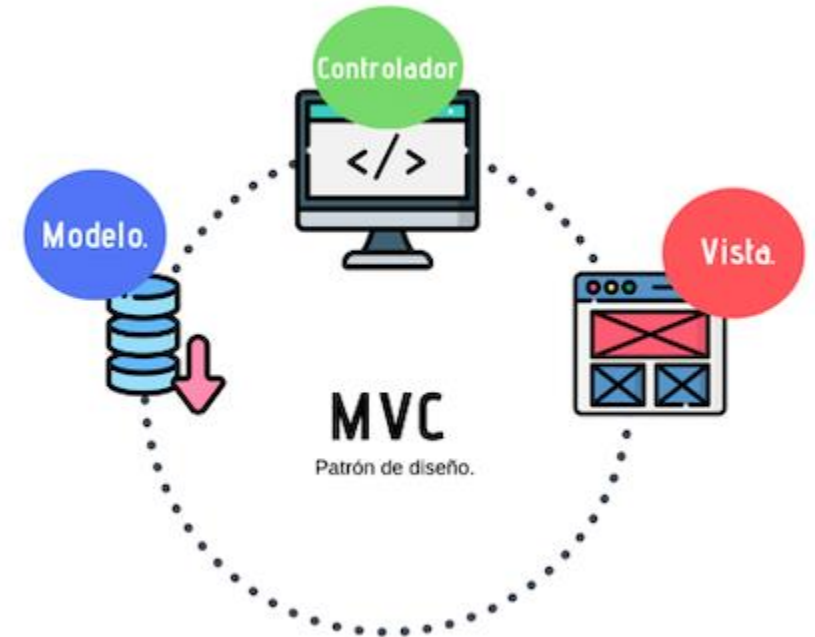
Vista

Controlador

Model

View

Controller



## Modelo

Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado, normalmente en una base de datos.

## Vista

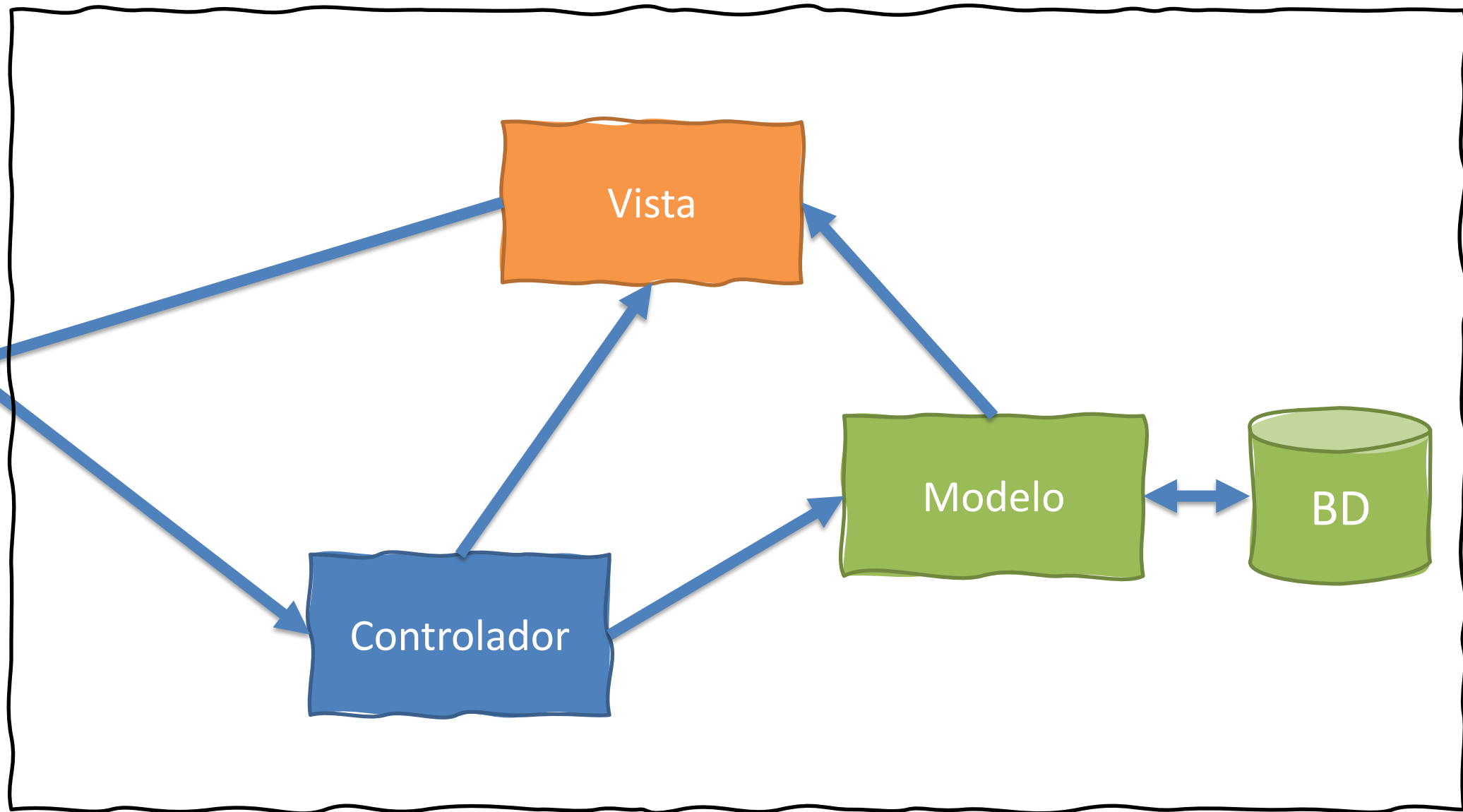
contienen el código de aplicación que genera la visualización de las interfaces de usuario

## Controlador

Contiene el código para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.



usuario



software



Trabajar utilizando arquitectura MVC no es complejo y se puede aplicar para cualquier código.

Pero también se puede utilizar un framework que te ayuda a que las cosas sean más fáciles.

- 1.- Evita escribir código repetido
- 2.- Utiliza buenas prácticas
- 3.- Facilita hacer cosas avanzadas
- 4.- Mejora los tiempos de desarrollo

# Algunos Frameworks



Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.



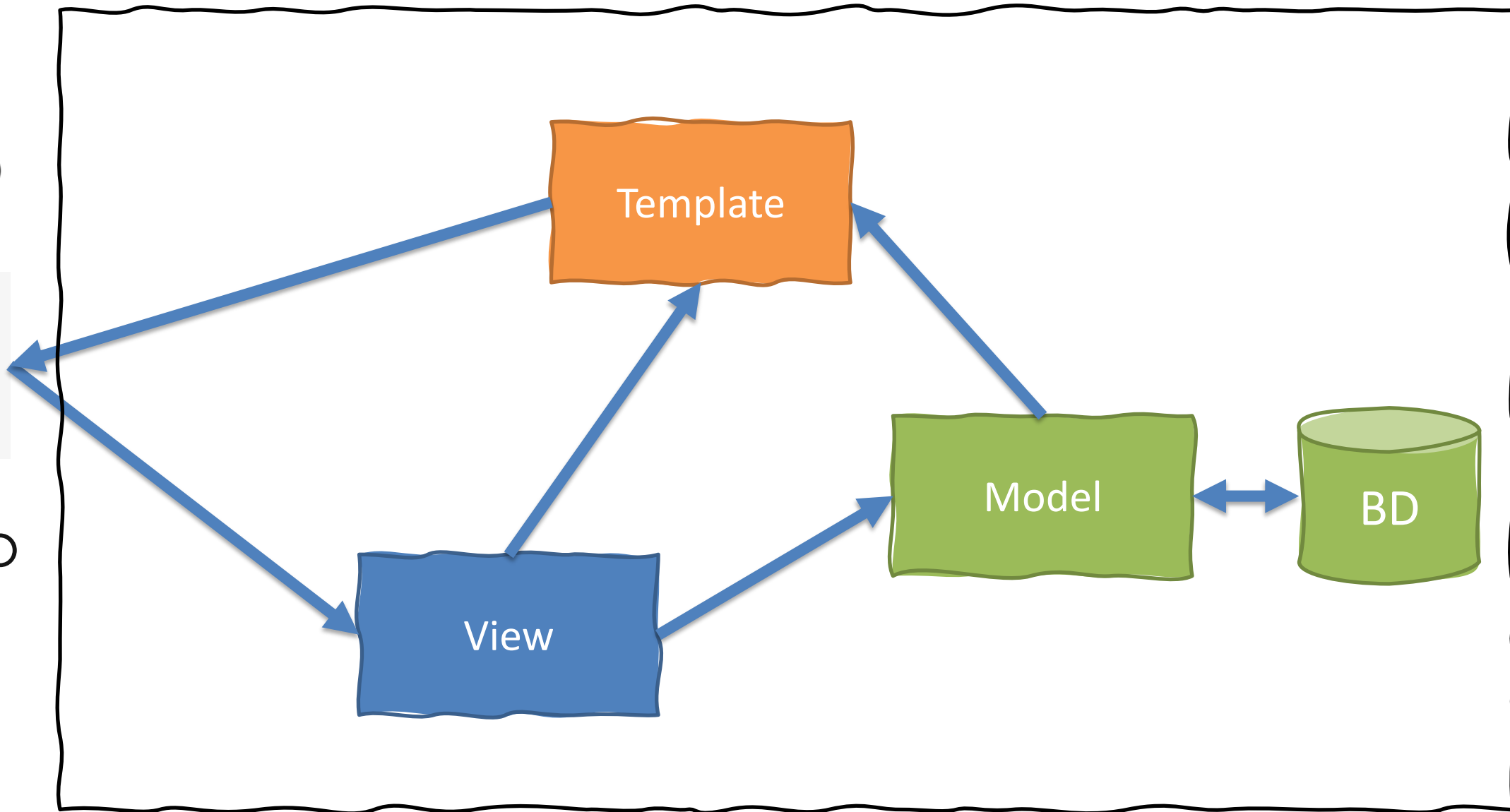
The PHP Framework for Web Artisans.

Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things.



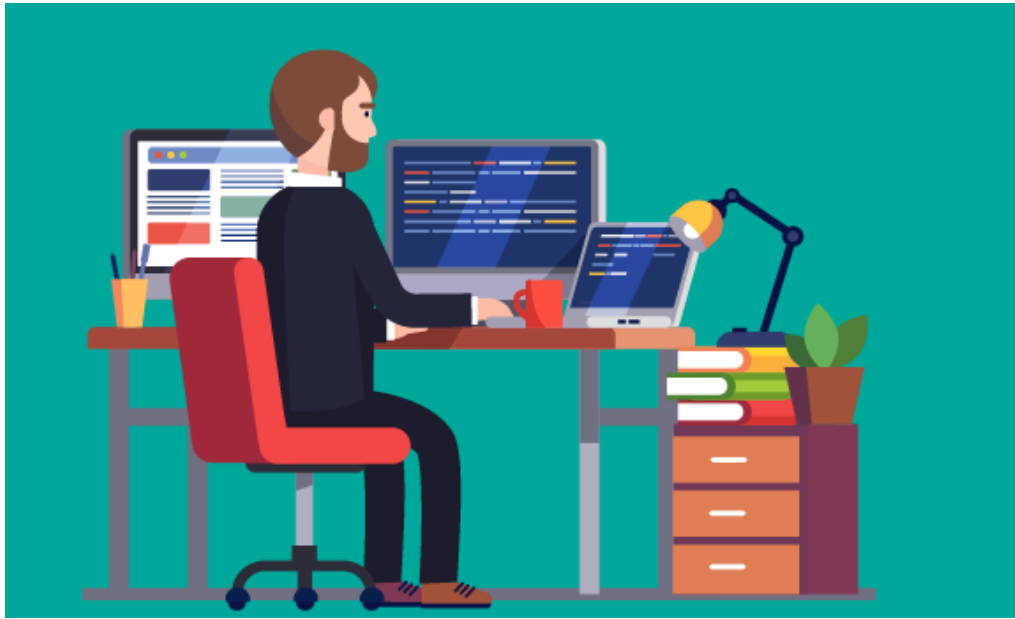


usuario



Framework MTV

# Ahora ¿cómo desarrollo un software en un esquema de trabajo en equipo?





# Opciones:

- Denme la directrices y yo programo todo.
- Tarreo virtual
- Frontend /Backend
- MVC
- colaboración



Es una plataforma de desarrollo colaborativo para alojar proyectos que utiliza el sistema de control de versiones Git.

Sus principales características son:

- Wiki para cada proyecto
- Página web para cada proyecto
- Indicadores para ver cómo los colaboradores trabajan en sus repositorios y bifurcaciones del proyecto
- Herramientas de trabajo colaborativo
- Gestor de proyectos
- etc



Es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

Creado por Linus Torvalds en 2005

Sus principales características son:

- Permite un desarrollo no lineal (gestión de ramas y fusión de versiones)
- Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales.



## Primeros pasos

- 0.- Crear una cuenta.
- 1.- Crear un repositorio
- 2.- Crear una división
- 3.- Realizar cambio (commit)
- 4.- Abre una solicitud de extracción
- 5.- Combina tu solicitud de extracción



# Flujo de Desarrollo

**Master** - rama (Branch) que tiene la última versión productiva del código.

**Release** - Es la rama que contiene los nuevos features terminados que se van desarrollando para el siguiente lanzamiento (release)



# Flujo de Desarrollo (cont)

**Develop** Es la rama que contiene las características (*features*) en desarrollo en una iteración, esta rama será posteriormente parte de Release mediante un *pull request*.

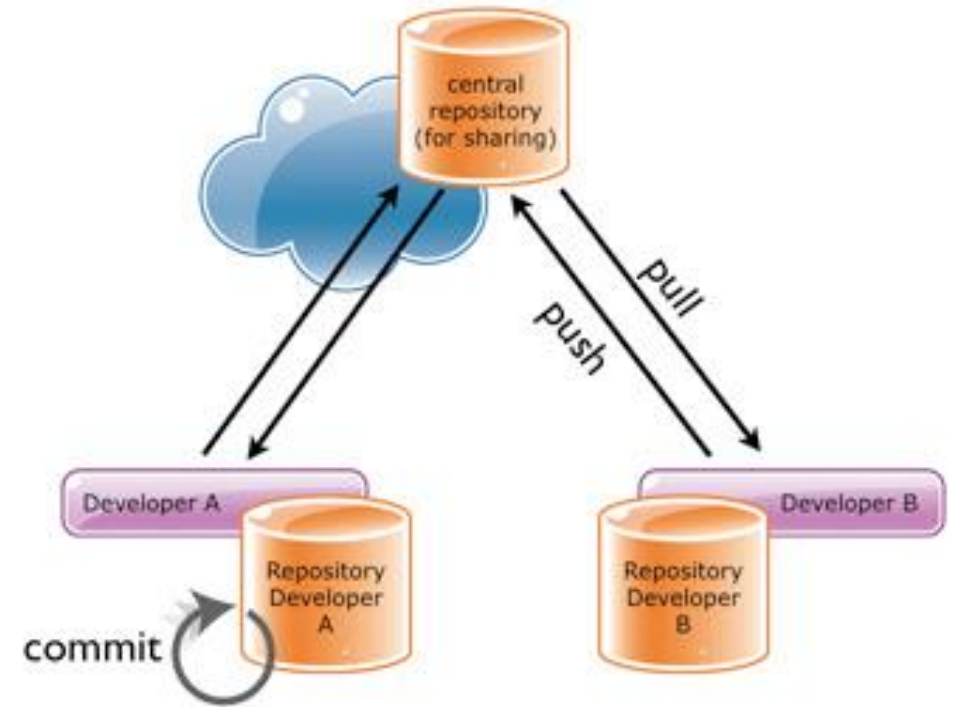
**Feature** Es la rama que contiene el *feature* en el que estás trabajando personalmente (varios desarrolladores pueden trabajar en un feature).



**Hotfix** Es la rama que contiene cambios urgentes sobre master que permiten corregir un bug o resolver un error

# Algunos Comandos

```
git add file.txt  
git commit -m "versión 0.0.1"  
git push origin MYBRANCH  
0  
git pull origin MYBRANCH  
0  
Git push origin master
```



# Herramientas

git-flow <http://danielkummer.github.io/git-flow-cheatsheet/>

```
git flow init
git flow feature start MYFEATURE
git flow feature finish MYFEATURE
git flow feature publish MYFEATURE
git flow release start RELEASE [BASE]
git flow release publish RELEASE
git flow hotfix start VERSION [BASENAME] ...
```

# Primeros Pasos

## configuración

```
git config global user.name "Tu nombre aquí"  
git config global user.email "tu_email_aquí@example.com"
```

## inicializar

```
git init
```

# Guardar archivos en el repositorio

```
git init  
git add file.txt  
git status  
git commit -m "mensaje para el commit"
```

# Subir al repositorio

```
git remote add origin https://github.com/aqui-tu-repo.git
```

```
git push -u origin master
```

# Clonar un repositorio

```
git clone https://github.com/aqui-tu-repo.git .
```

# Log del repositorio

```
git log
```

# Borrar archivo del repositorio (pero no de mi equipo)

```
git rm --cached nombre_archive  
git commit -m 'Eliminados archivos no deseados'  
git push
```

## Crear un Tag

```
git tag v0.0.1 -m "Primera versión"
```

después hay que hacer el commit

## Ver estado de versiones

```
git tag  
git show v0.0.1
```

## Enviar las tags

```
git push --tags  
  
git push origin v0.0.4  
  
git push origin master --tags
```



## Crear una rama

```
git branch experimental  
git show-branch  
git checkout experimental
```

## Fusionar una rama con la master

```
git checkout master  
git merge experimental  
git merge experimental -m 'Esto es un merge con mensaje'
```

## Subir la rama al repositorio

```
git push -u origin experimental
```



# Elige un flujo de trabajo

## Git Feature Branch Workflow:

Todos los features deben tener una rama dedicada en lugar de concentrar todo en master.

## Gitflow Workflow:

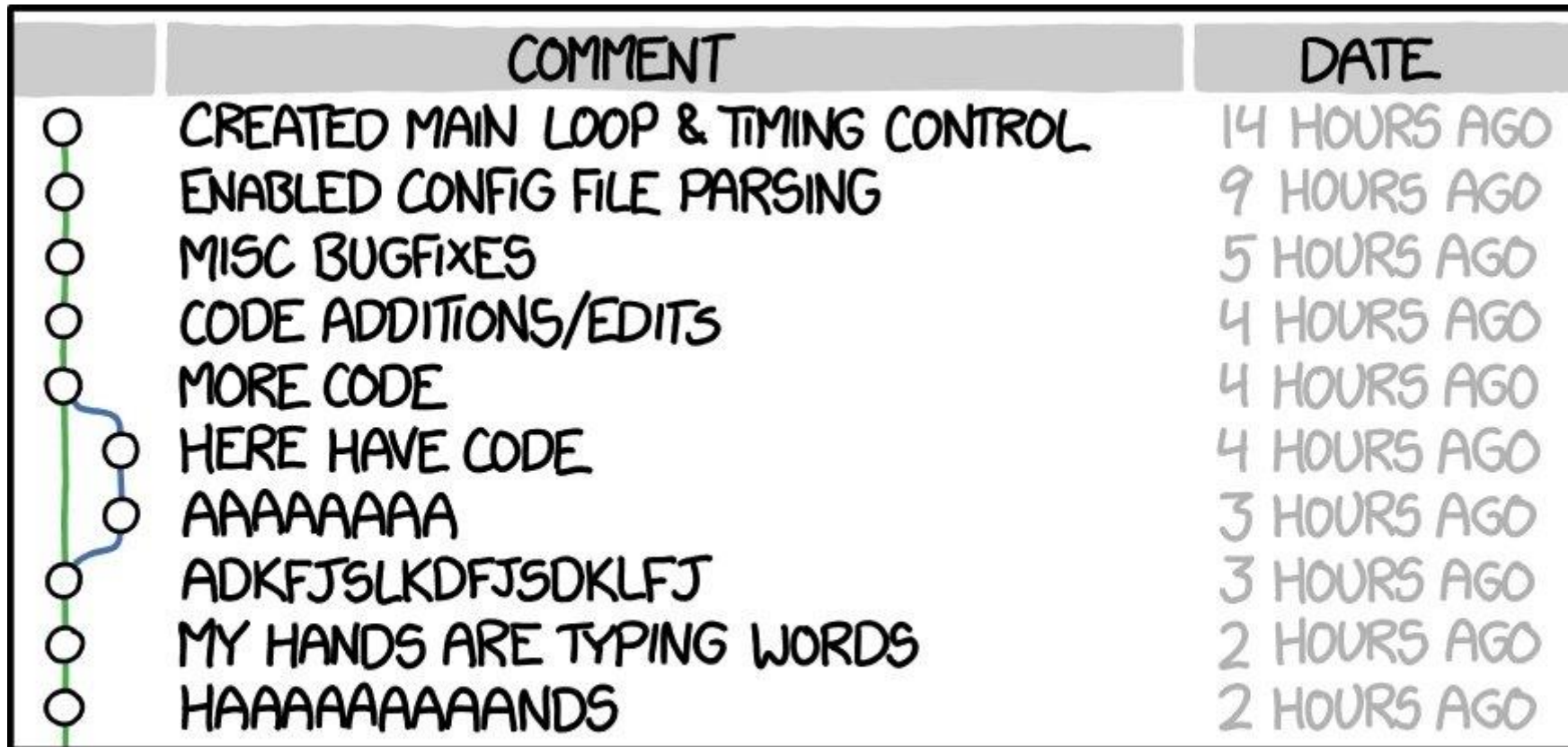
Es similar al anterior solo que la estructura de ramas está diseñado alrededor de las entregas del proyecto.

## Forking Workflow:

En lugar de usar un solo repositorio centralizado le da la opción a cada colaborador de tener uno propio.



Usa el mensaje de commit para aportar contexto y explicar el por qué detrás de un cambio



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Puedes emplear algunos sufijos como:

- feat:** Describe si trabajaste en un nuevo feature
- fix:** Describe si solucionaste un bug
- docs:** Dice si hiciste algún cambio en la documentación
- test:** Indica si añadiste un test
- refactor:** Nos muestra que se ejecutó algún refactor en el código

## Ejemplo:

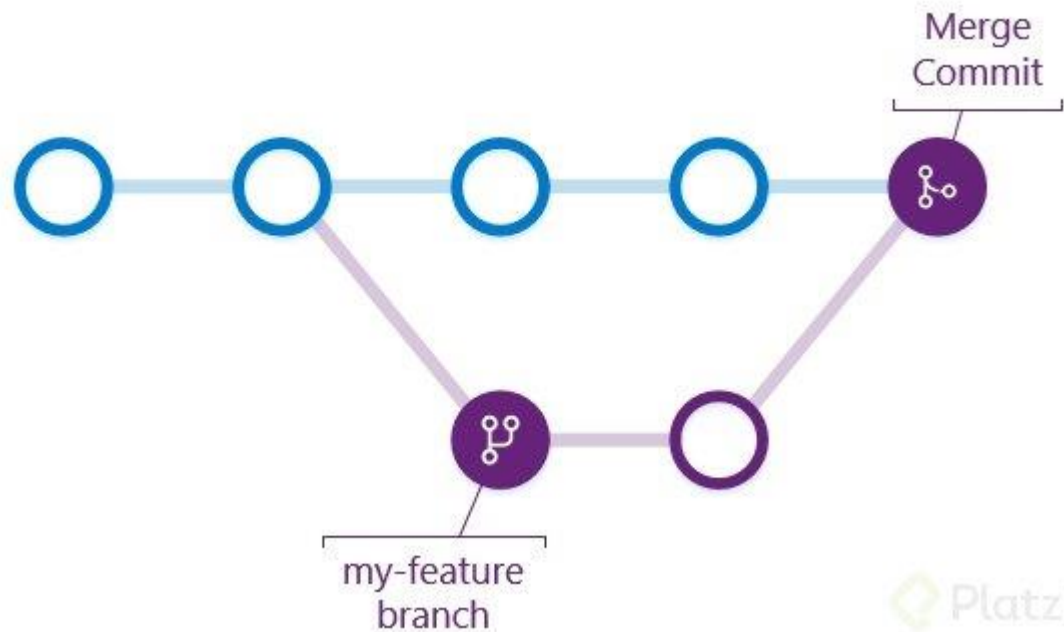
```
git commit -m "docs: new documentation" -m  
"Description....\nNew line....\nAnother new line"
```

Puedes configurar que se edite con vscode

```
git config --global core.editor "code --wait"
```

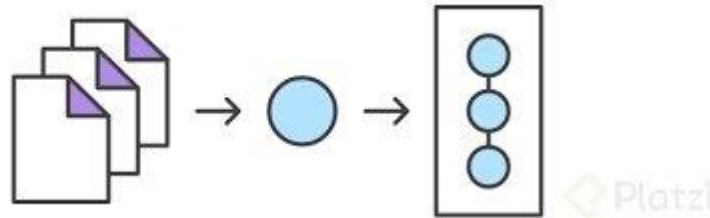
## Envía tus cambios en pull-request

Un pull-request es una solicitud que hacemos al repositorio cuando enviamos código con cambios, idealmente el propietario del repositorio revisa el código y decide si es apto para ser integrado.



## Añade solo los archivos en los que estuviste trabajando

```
git add controller/user.php  
git add model/user.php
```



## Código para el control de versiones (Semantic versioning )

Usa una secuencia  
de tres dígitos  
(Major.Minor.Patch)

**4 . 2 . 1**  
MAJOR Minor patch

**patch**: cambios menores. Mejoras en interfaz de usuarios, correcciones, cambios menores.

**Minor**: nuevas características o paginas, etc.

**Major**: cambios sustanciales,

El trabajo de un nuevo reléase se puede ver en la siguiente secuencia de versiones.

Alfa	1.2.0-a
Beta	1.2.0-b
Release Candidate	1.2.0-rc
Release	1.2.0
Post Release Fixes	1.2.2





Es una librería o framework que permite mejorar la interfaz de usuario.  
Facilita el diseño de sitios web ofreciéndonos herramientas para que nuestro sitio se vea en toda clase de dispositivos

Incluye un css y un js donde concentra sus funcionalidades y se puede comenzar a utilizar agregando este par de lineas

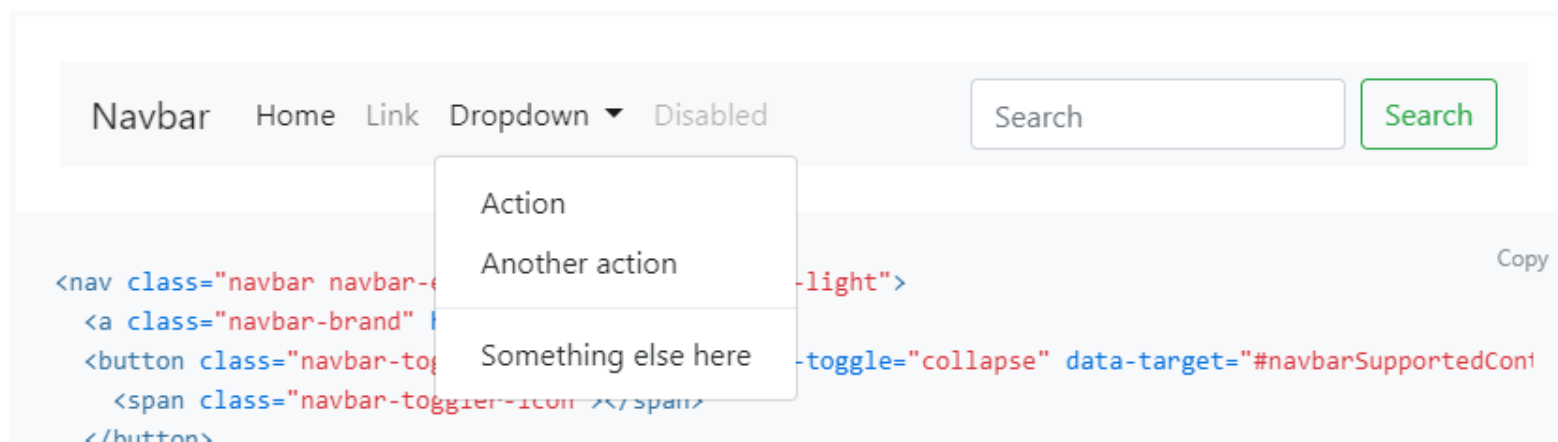
```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-B4gtl1jrGC7Jh4AgTPSdUtOBvf08shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
```

Utiliza una estructura de grilla de 12 columnas que permite adaptar las paginas fácilmente

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Tiene una navbar fácil de implementar



¿DUDAS?

