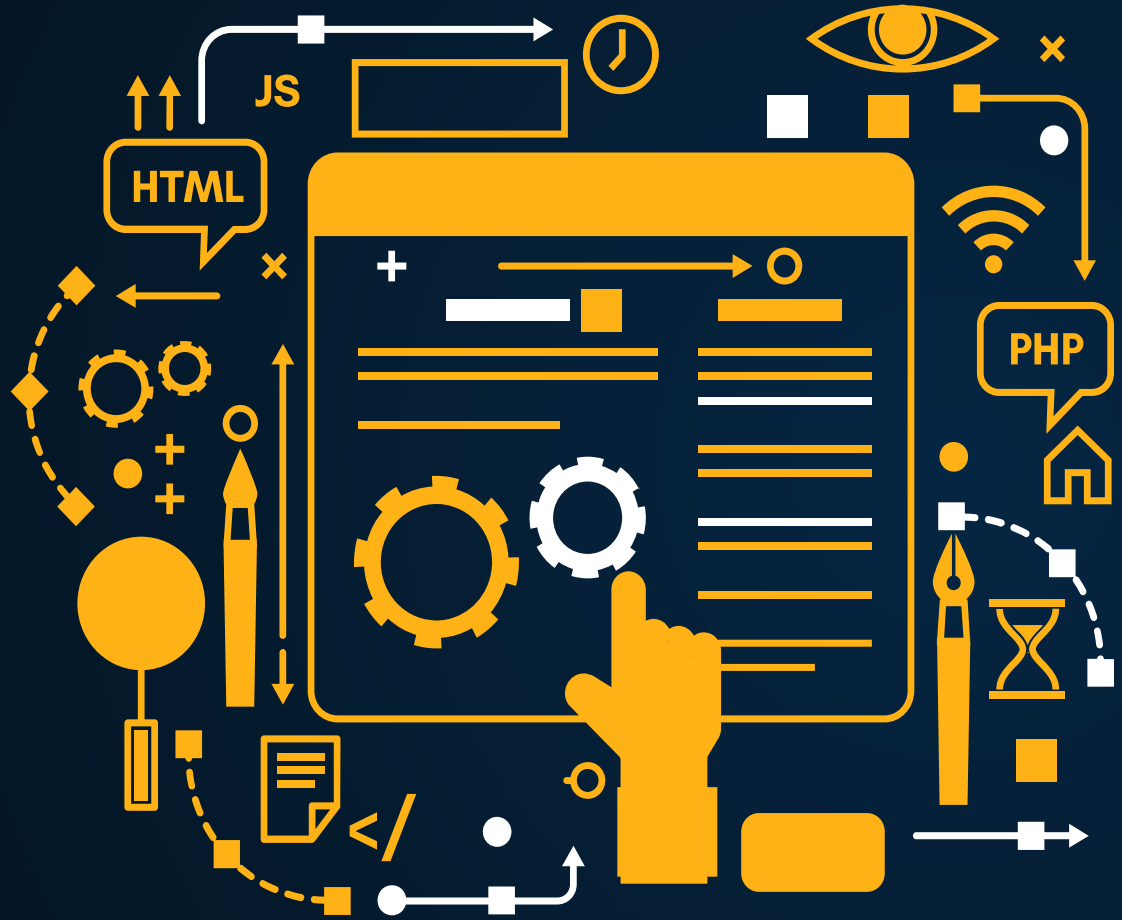

ENTORNOS DE DESARROLLO

UD4.3 “Branching y resolución de conflictos”

Contenidos

- Ramas en GIT
  - Comandos más usados
- Merge
- Rebase
- Resolución de conflictos
- Ejemplo práctico

Ramas en GIT

Las ramas son una parte fundamental del flujo de trabajo

Las ramas en Git son simplemente apuntadores a un commit específico. Son útiles para aislar cambios en el código mientras trabajas en diferentes características o funcionalidades. La rama por defecto en Git se llama `[master|main]`

Ramas - Comandos más usados

1. Crear una rama

```
git branch nombre_de_la_rama
```

Este comando crea una nueva rama llamada **`nombre_de_la_rama`**.

2. Cambiar a una rama

```
git checkout nombre_de_la_rama
```

Este comando te permite cambiar a la rama **`nombre_de_la_rama`**.

Ramas - Comandos más usados

3. Crear y cambiar a una rama en un solo comando

```
git checkout -b nombre_de_la_rama
```

Este comando crea una nueva rama llamada **`nombre_de_la_rama`** y luego cambia a esa rama.

4. Fusionar una rama

```
git merge nombre_de_la_rama
```

Este comando fusiona los cambios de la rama **`nombre_de_la_rama`** en la rama actual.

Ramas - Comandos más usados

5. Eliminar una rama

```
git branch -d nombre_de_la_rama
```

Este comando elimina la rama nombre_de_la_rama.

Ramas - Comandos más usados

6. Seguimiento de ramas remotas con ramas locales

Para trackear una rama de un repositorio clonado en Git, puedes usar el comando **`checkout`** con la opción **`-t`** (track)

```
git checkout -t origin/nombre_de_la_rama
```

Este comando crea una nueva rama local con el mismo nombre que **`nombre_de_la_rama`**, apunta a la rama **`nombre_de_la_rama`** en el repositorio remoto y cambia a la nueva rama local. Si la rama ya existe en tu repositorio local y solo quieres establecer el seguimiento, puedes usar el comando branch con la opción **`-u`** o **`--set-upstream-to`**

```
git branch -u origin/nombre_de_la_rama
```

Merge (I)

`git merge` es un comando que se utiliza para combinar los cambios de una rama a otra.

Imagina que tienes dos ramas: `master` y `feature`. Has hecho algunos cambios en la rama `feature` que te gustaría incorporar en `master`.

El historial de Git podría verse así:

```
A - B - C (master)
     \
      D - E (feature)
```

Merge (II)

Aquí es donde git `merge` entra en juego. Al ejecutar `git checkout master` seguido de `git merge feature`, Git combina los cambios de la rama `feature` en la rama `master`.

```
    A - B - C - F (master)
          \       /
           D - E (feature)
```

El commit `F` es un nuevo commit que Git crea automáticamente. Este commit `F` es un “commit de fusión” que tiene dos padres: `C` y `E`.

Es importante recalcar que `git merge` mantiene todo el historial de commits intacto.

Rebase (I)

`git rebase` es un comando poderoso que se utiliza para integrar cambios de una rama a otra.

Imagina que tienes dos ramas: `master` y `feature`. Mientras estabas trabajando en la rama `feature`, alguien más hizo algunos commits en la rama master, quedando el historial de esta forma:

```
A - B - C (master)
     \
      D - E (feature)
```

Rebase (II)

Ahora, quieres incorporar los cambios de `master` en tu rama `feature`. Aquí es donde `git rebase` entra en juego. Al ejecutar `git rebase master` mientras estás en la rama `feature`, Git “reaplica” tus cambios en `feature` sobre los de `master`. Dejando el historial del siguiente modo:

```
A - B - C (master)
         \
          D' - E' (feature)
```

Los commits `D'` y `E'` son versiones “rehechas” de los commits `D` y `E` que originalmente hiciste en la rama feature.

Es importante recalcar que `git rebase` altera el historial de commits, por lo que se debe usar con cuidado. Es una buena práctica no usar git rebase en ramas que otros puedan estar utilizando.

Resolución de conflictos

Un conflicto en Git ocurre cuando dos ramas modifican la **misma línea en el mismo archivo**, o cuando **un archivo ha sido modificado en una rama y eliminado en la otra**, y luego intentas fusionar estas dos ramas.

Git no puede resolver estos cambios automáticamente y necesita de nuestra intervención para decidir qué cambios conservar.

Cuando ocurre un conflicto, Git modifica el archivo para indicar los cambios de ambas ramas. Los conflictos se representan de la siguiente manera:

```
<<<<<<< HEAD
Cambios en la rama actual
=======
Cambios en la otra rama
>>>>>>> nombre_de_la_otra_rama
```

Resolución de conflictos

- `<<<<<<< HEAD` marca el inicio de los cambios que existen en la rama actual.
- `=======` separa tus cambios de los cambios en la otra rama.
- `>>>>>>> nombre_de_la_otra_rama` marca el final de los cambios de la otra rama.

Para resolver el conflicto, debes editar el archivo para decidir qué cambios quieres conservar (puedes **conservar los cambios de una rama**, **de la otra**, **de ambas** o **incluso puedes escribir algo completamente nuevo**), y **luego agregar el archivo y hacer un commit**.

Ejemplo práctico

Realicemos un ejercicio para poner en práctica rebase, merge y resolución de conflictos.

1. Creamos un directorio “`lista_compra`”
2. Inicializamos un repositorio en el directorio “`lista_compra`”
3. Creamos un fichero llamado “`lista.txt`”, lo añadimos y commit de los cambios
4. Ahora creamos una rama llamada “`pareja`”
5. Comprobamos las ramas que tenemos y en cuál estamos. Deberíamos estar en “`main`”
6. Abrimos el fichero “`lista.txt`” y escribimos dos elementos de una lista de la compra, añadimos los cambios y confirmarmos.
7. Repetimos la acción 6 sobre la rama “`pareja`”
8. Repetimos de nuevo la acción 6 sobre la rama “`main`”
9. Resolvamos el conflicto que se ha generado
    a. Primero “`rebase`” de “`pareja`” sobre “`main`”
    b. A continuación integramos la rama “`pareja`” en “`main`” con “`merge`”

Ejercicio

Pongamos en práctica los conceptos que hemos trabajado a través de un ejercicio tematizado en el universo de películas de Marvel.

https://aules.edu.gva.es/fp/pluginfile.php/5583527/mod_resource/content/1/DAM%20in%20the%20MULTIVERSE%20OF%20GITNESS.pdf

Bibliografía

- git-branch Documentation
- git-checkout Documentation
- git-merge Documentation
- git-rebase Documentation