



UD 03.CONTROL Y GESTIÓN DE VERSIONES

PARTE 2 DE 5: RAMAS EN GIT

Entornos de desarrollo (ED)

Raúl Palao

UD 03.CONTROL Y GESTIÓN DE VERSIONES

3.CONTROL Y GESTIÓN DE VERSIONES

3.2. RAMAS EN GIT

3.2.1 ¿QUÉ SON LAS RAMAS?

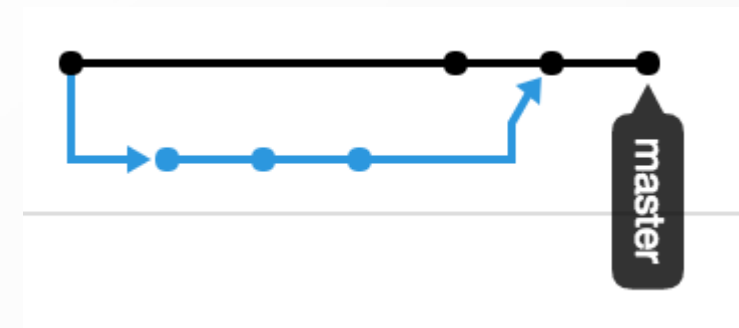
3.2.2 COMANDOS BÁSICOS

3.2.3 CONFLICTOS

3.2.4 EJEMPLO DE RAMAS EN GIT

3.2.1 ¿Qué son las ramas?

- La ramificación nos permite aislar nueva funcionalidad a la hora del desarrollo para que no impacte de una forma directa al proyecto principal.



3.2.1 ¿Qué son las ramas?

- En todo proyecto tendremos como mínimo las siguiente ramas:
 - **MASTER:**
 - Es la rama principal del proyecto con el código en PRODUCCIÓN.
 - ¡NUNCA SE TOCA CÓDIGO DIRECTAMENTE DE ESTA RAMA!
 - SE INTEGRA SIEMPRE CON EL CÓDIGO DE DEVELOP.
 - **DEVELOP:**
 - Es una de las ramas secundarias para el código en PREPRODUCCIÓN.
 - Siempre parte de MASTER.
 - **FEATURE** (habrá varias):
 - Cada programador que vaya a realizar un cambio desde crear una NUEVA RAMA DESDE **DEVELOP**.
 - Después integra desde develop.



UD 03.CONTROL Y GESTIÓN DE VERSIONES

3.CONTROL Y GESTIÓN DE VERSIONES

3.2. RAMAS EN GIT

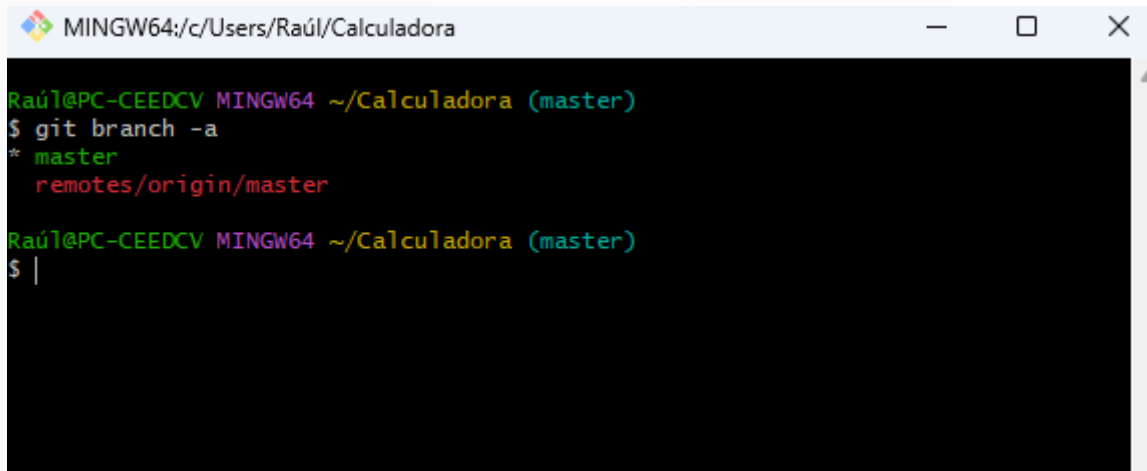
3.2.1 ¿QUÉ SON LAS RAMAS?

3.2.2 COMANDOS BÁSICOS

3.2.3 CONFLICTOS

3.2.4 EJEMPLO DE RAMAS EN GIT

3.2.2 Comandos básicos



```
MINGW64:/c/Users/Raúl/Calculadora
Raúl@PC-CEEDCV MINGW64 ~/Calculadora (master)
$ git branch -a
* master
remotes/origin/master
Raúl@PC-CEEDCV MINGW64 ~/Calculadora (master)
$ |
```

- Podemos ver todas las ramas con el comando:
git branch -a
- En este caso podemos ver que hay dos ramas
 - master local
 - master remoto

3.2.2 Comandos básicos

```
MINGW64:/c/Users/Raúl/HolaMundo

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git branch -a
* master

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git branch develop

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git checkout develop
Switched to branch 'develop'

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$
```

```
MINGW64:/c/Users/Raúl/HolaMundo

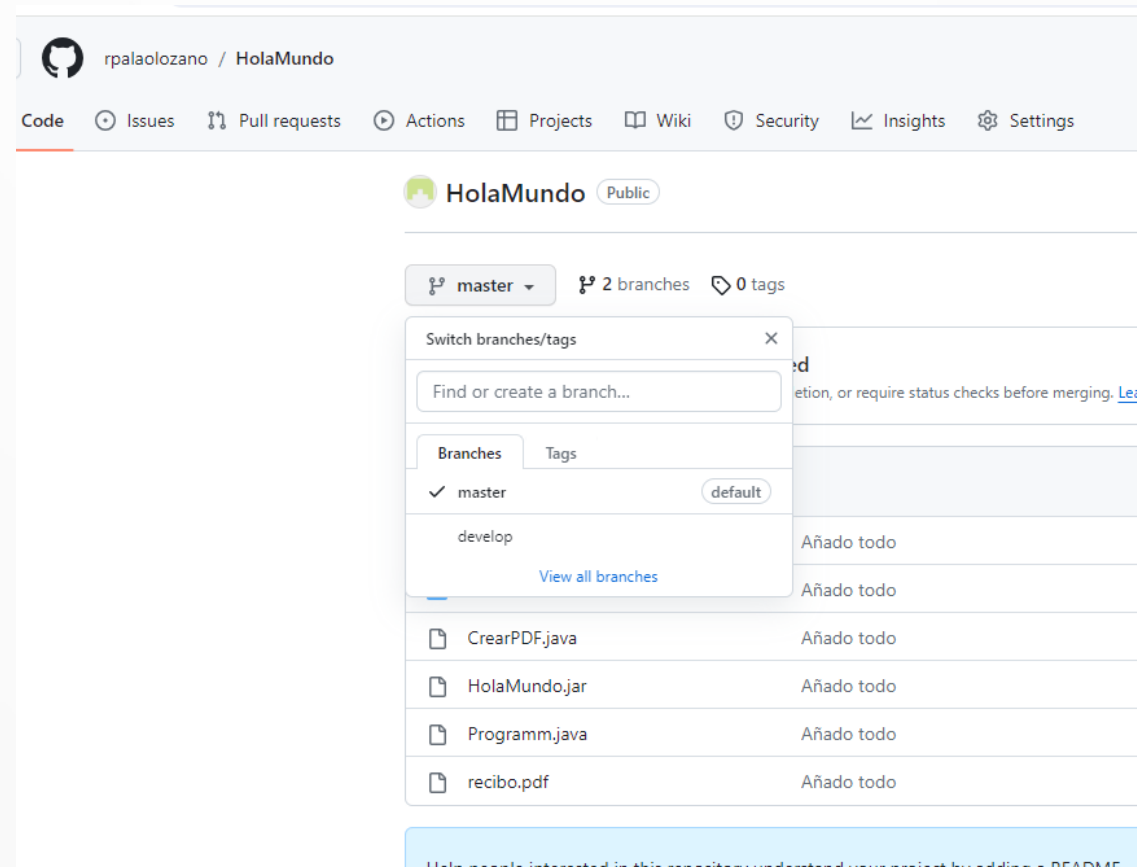
Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/rpalaozano/HolaMundo/pull/new/develop
remote:
To https://github.com/rpalaozano/HolaMundo.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$
```

- Para crear una nueva rama usamos el comando:
git branch NOMBRE
- La rama siempre se crea con el código de la rama DESDE LA QUE CREAMOS.
- Para movemos a la rama creada con el comando:
git checkout NOMBRE
- Para subir la rama creada a GitHub:
git push -u origin develop

3.2.2 Comandos básicos

- En GitHub podemos ver las ramas remotas en el desplegable:



3.2.2 Comandos básicos

- Para realizar una nueva funcionalidad siempre debemos crear una nueva rama llamada **FEATURE** desde la rama **DEVELOP**.
- Informamos en GitHub que esa rama está creada subiéndola (con el mismo código).

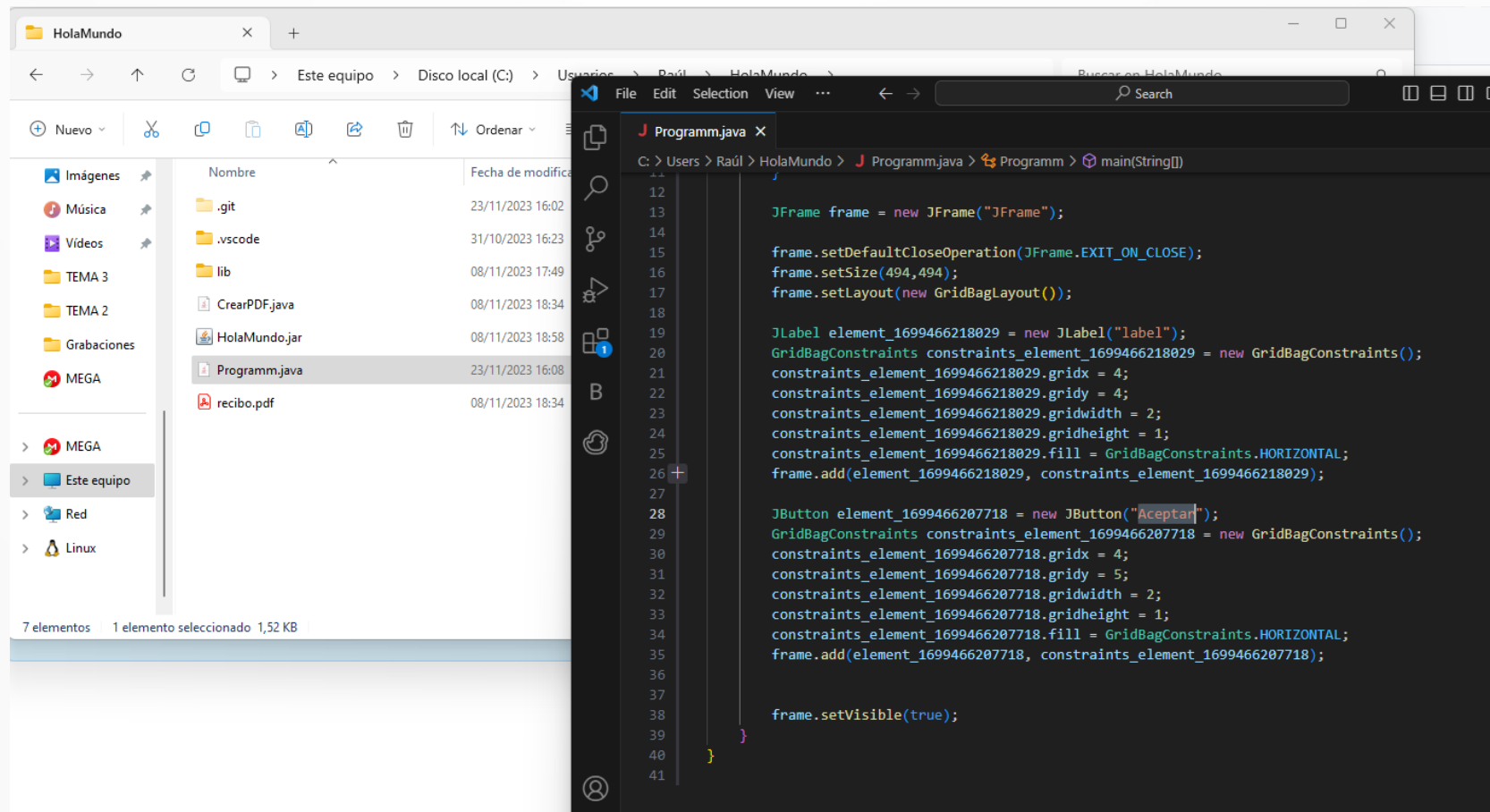
```
Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git branch feature1

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git checkout feature1
Switched to branch 'feature1'

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git push -u origin feature1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:   https://github.com/rpalaoizano/HolaMundo/pull/new/feature1
remote:
To https://github.com/rpalaoizano/HolaMundo.git
 * [new branch]      feature1 -> feature1
branch 'feature1' set up to track 'origin/feature1'.
```

3.2.2 Comandos básicos

- En este paso cambiaríamos el código añadiendo nueva funcionalidad:



3.2.2 Comandos básicos

- Ahora al hacer el git status aparece el fichero modificado:

```
MINGW64/c/Users/Raúl/HolaMundo
$ git push -u origin feature1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature1' on GitHub by visiting:
remote:   https://github.com/rpalaoizano/HolaMundo/pull/new/feature1
remote:
To https://github.com/rpalaoizano/HolaMundo.git
 * [new branch]   feature1 -> feature1
branch 'feature1' set up to track 'origin/feature1'.

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git status
On branch feature1
Your branch is up to date with 'origin/feature1'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Programm.java

no changes added to commit (use "git add" and/or "git commit -a")

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$
```

- Tras esto ya podemos hacer el add y el commit. Así como el push (sin el -u ya que la rama remota ya está creada):

```
MINGW64/c/Users/Raúl/HolaMundo
Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git add .
warning: in the working copy of 'Programm.java', LF will be replaced by CRLF the
next time Git touches it

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git commit -m "Cambio el botón por Aceptar"
[feature1 4a06e2f] Cambio el botón por Aceptar
1 file changed, 1 insertion(+), 1 deletion(-)

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git push origin feature1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 319 bytes | 319.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/rpalaoizano/HolaMundo.git
 ffc7c57..4a06e2f feature1 -> feature1
```

3.2.2 Comandos básicos

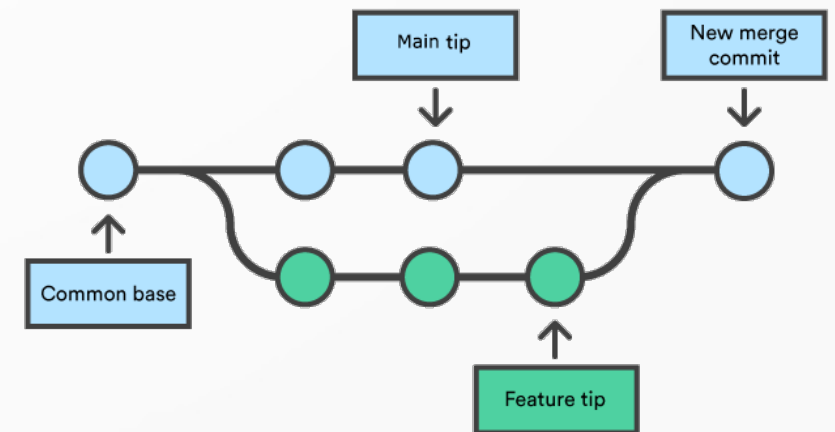
- Ahora debemos incorporar los cambios de FEATURE a DEVELOP:
 - Nos movemos a DEVELOP: `git checkout develop`
 - Hacemos un pull: `git pull origin develop`
 - Hacemos la incorporación de los cambios: `git merge feature1`
 - Comprobamos que todo es correcto y hacemos el push: `git push origin develop`

```
MINGW64:/c/Users/Raúl/HolaMundo
Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (feature1)
$ git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git pull origin develop
From https://github.com/rpalaoizano/HolaMundo
* branch      develop    -> FETCH_HEAD
Already up to date.

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git merge feature1
Updating ffc7c57..4a06e2f
Fast-forward
 Programm.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rpalaoizano/HolaMundo.git
 ffc7c57..4a06e2f develop -> develop
```



3.2.2 Comandos básicos

- Ahora integramos todos los cambios de DEVELOP a MASTER:
 - Nos movemos a MASTER: `git checkout master`
 - Hacemos un pull: `git pull origin master`
 - Hacemos la incorporación de los cambios: `git merge develop`
 - Comprobamos que todo es correcto y hacemos push: `git push origin master`

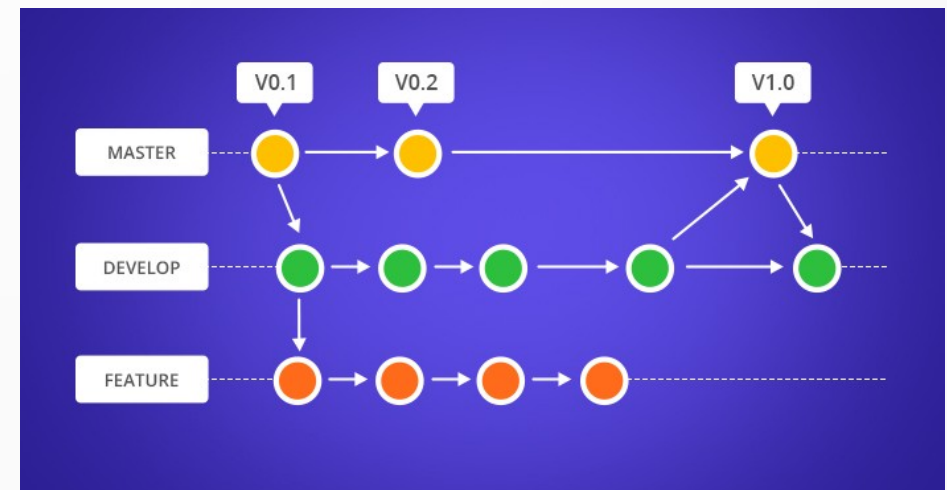
```
MINGW64/c/Users/Raúl/HolaMundo

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (develop)
$ git checkout master
Switched to branch 'master'

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git pull origin master
From https://github.com/rpalaozano/HolaMundo
* branch      master      -> FETCH_HEAD
Already up to date.

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git merge develop
Updating ffc7c57..4a06e2f
Fast-forward
 Programm.java | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git push origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rpalaozano/HolaMundo.git
 ffc7c57..4a06e2f  master -> master
```



3.2.2 Comandos básicos

- Tras esto podríamos eliminar la rama FEATURE:

- En la terminal:

git branch -d feature1

```
Raúl@PC-CEEDCV MINGW64 ~/HolaMundo (master)
$ git branch -d feature1
Deleted branch feature1 (was 4a06e2f).
```

- En la nube:

The screenshot shows the GitHub web interface for the 'HolaMundo' repository. The 'Your branches' section lists two branches: 'develop' and 'feature1'. Next to 'feature1', there is a 'Delete feature1' button. A blue arrow points to this button. On the left, a 'Switch branches/tags' dropdown is open, showing a list of branches including 'master', 'develop', and 'feature1'. The 'feature1' branch is currently selected.

UD 03.CONTROL Y GESTIÓN DE VERSIONES

3.CONTROL Y GESTIÓN DE VERSIONES

3.2. RAMAS EN GIT

3.2.1 ¿QUÉ SON LAS RAMAS?

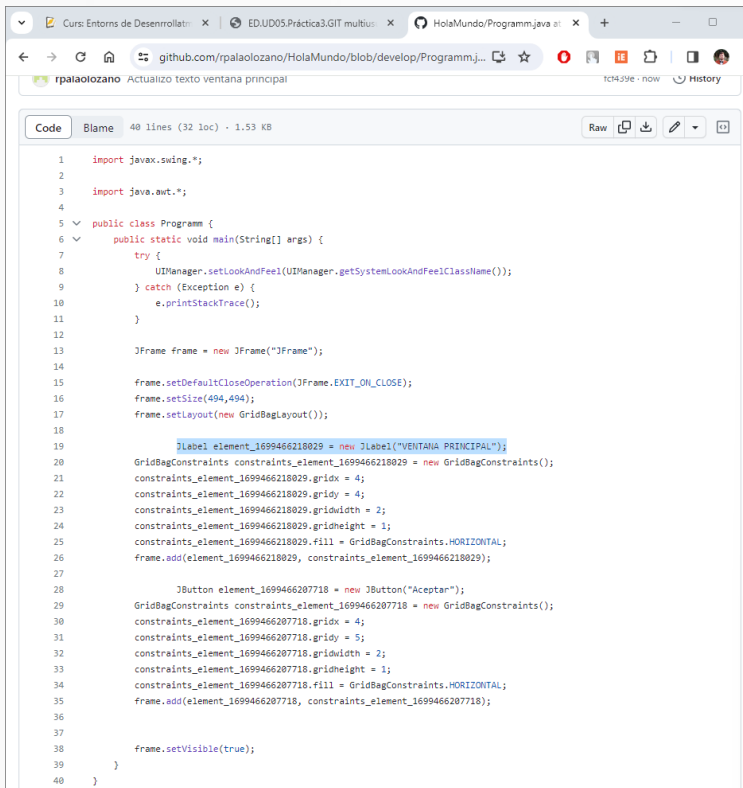
3.2.2 COMANDOS BÁSICOS

3.2.3 CONFLICTOS

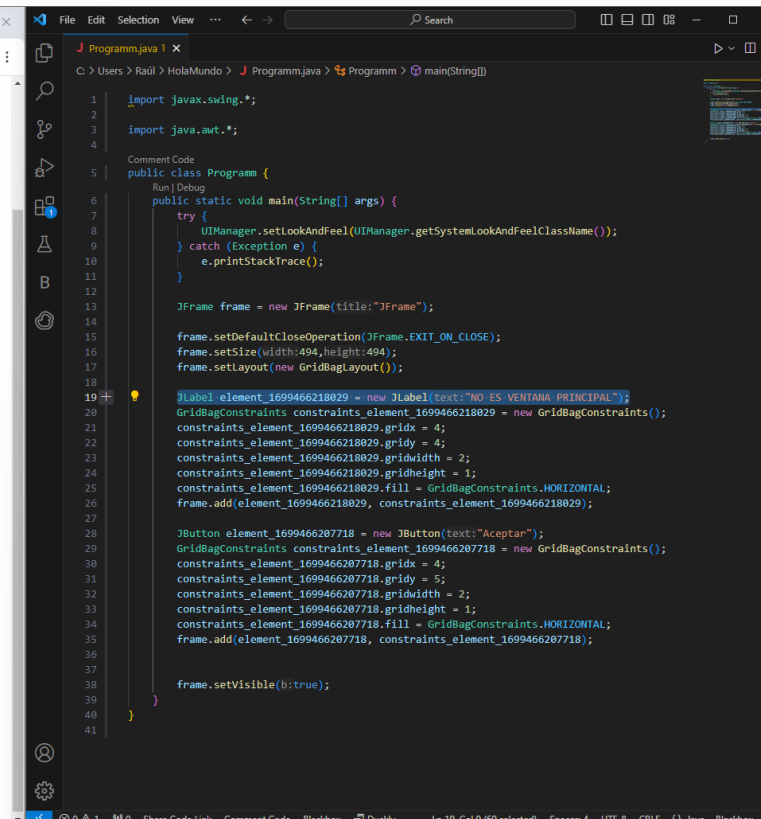
3.2.4 EJEMPLO DE RAMAS EN GIT

3.2.3 Conflictos

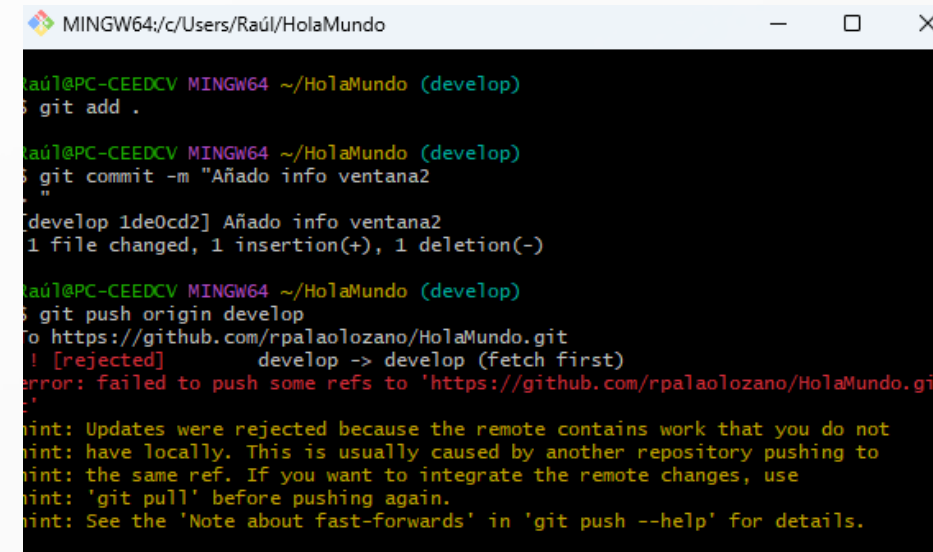
- Los conflictos aparecen cuando dos o más usuarios están manipulando el mismo fichero DE LA MISMA RAMA.
- Suele ocurrir cuando hay cambios que están en la nube pero no en local y hacemos un PUSH.



A screenshot of a web browser displaying a GitHub repository file view for Program.java. The file is 40 lines long, with a size of 1.53 KB. The code is in Java and shows a Swing application with a JFrame, GridBagLayout, and several JLabel and JButton components. The file is named Program.java and is located in the HolaMundo/Programm.java file.



A screenshot of an IDE showing the same Program.java file. The code is identical to the one in the browser, but it includes a comment at the top: "Comment Code". The IDE also shows a "Run" button and a "Debug" button. The file is named Program.java and is located in the HolaMundo/Programm.java file.



A screenshot of a terminal window showing a git push command and its output. The command is "git push origin develop". The output shows that the push was rejected because the remote contains work that you do not have locally. The message says: "Updates were rejected because the remote contains work that you do not have locally. This is usually caused by another repository pushing to the same ref. If you want to integrate the remote changes, use 'git pull' before pushing again. See the 'Note about fast-forwards' in 'git push --help' for details."

3.2.3 Conflictos

- Al hacer el pull olvidado nos aparecerá algo como lo siguiente en el fichero.
- En HEAD aparece la versión de nuestro local y debajo del ===== la versión de la nube.
- Borramos las líneas que no nos interesan y haríamos el add, commit y push.

```
C: > Users > Raúl > HolaMundo > J Programm.java > ...
Click here to ask Blackbox to help you code faster |
1 import javax.swing.*;
2
3 import java.awt.*;
4
5 public class Programm {
6     public static void main(String[] args) {
7         try {
8             UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
9         } catch (Exception e) {
10             e.printStackTrace();
11         }
12
13         JFrame frame = new JFrame(title:"JFrame");
14
15         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         frame.setSize(width:494,height:494);
17         frame.setLayout(new GridBagLayout());
18
19 <<<<<<< HEAD (Current Change)
20 JLabel element_1699466218029 = new JLabel(text:"NO ES VENTANA PRINCIPAL");
21
22 =====
23 >>>>>> fcf439ea372d8fcae034a46114fc81aa39b708a0 (Incoming Change)
24 GridBagConstraints constraints_element_1699466218029 = new GridBagConstraints();
25 constraints_element_1699466218029.gridx = 4;
26 constraints_element_1699466218029.gridy = 4;
27 constraints_element_1699466218029.gridwidth = 2;
28 constraints_element_1699466218029.gridheight = 1;
```

```
C: > Users > Raúl > HolaMundo > J Programm.java > Programm > main(String[])
Click here to ask Blackbox to help you code faster |
1 import javax.swing.*;
2
3 import java.awt.*;
4
5 public class Programm {
6     public static void main(String[] args) {
7         try {
8             UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
9         } catch (Exception e) {
10             e.printStackTrace();
11         }
12
13         JFrame frame = new JFrame(title:"JFrame");
14
15         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16         frame.setSize(width:494,height:494);
17         frame.setLayout(new GridBagLayout());
18
19         JLabel element_1699466218029 = new JLabel(text:"NO ES VENTANA PRINCIPAL");
20         GridBagConstraints constraints_element_1699466218029 = new GridBagConstraints();
21         constraints_element_1699466218029.gridx = 4;
22         constraints_element_1699466218029.gridy = 4;
23         constraints_element_1699466218029.gridwidth = 2;
```

UD 03.CONTROL Y GESTIÓN DE VERSIONES

3.CONTROL Y GESTIÓN DE VERSIONES

3.2. RAMAS EN GIT

3.2.1 ¿QUÉ SON LAS RAMAS?

3.2.2 COMANDOS BÁSICOS

3.2.3 CONFLICTOS

3.2.4 EJEMPLO DE RAMAS EN GIT

- SHAKIRA
- PIQUÉ
- CLARA

han montado una empresa de desarrollo web y están haciendo una web para irse juntos de viaje.

¿EMPEZAMOS?





En este ejemplo tendremos que tener tres terminales, una por usuario.
También podemos tener nuestro equipo y dos máquinas virtuales para evitar confusiones:

TERMINAL DE SHAKIRA

Paso 1.1: Crear un repositorio con una carpeta

- 1) **Crea una carpeta**, dentro de ED.Practicas.GIT, llamada **03_REP_HIELO**.
- 2) Dentro de esa nueva carpeta, descarga y descomprime el archivo **hielo.zip** que encontrarás en el Aula Virtual.
- 3) Asegúrate de que no se crea una carpeta dentro de esa carpeta y de que **NO INCLUYES EL ARCHIVO ZIP** en el repositorio.
- 4) Abre una terminal/console
- 5) Luego, desde la consola:
 - 1) Crea el Repositorio **[git init]**
 - 2) **Identifícate con tu email y con el nombre de usuario:**
[git config --global user.name "xxx"]
[git config --global user.email xxx]
 - 3) Mira el estado del repositorio **[git status]**
 - 4) Añade al staging TODOS LOS ARCHIVOS **[git add .]**
 - 5) Grábalos en repositorio **[git commit -m "plantilla inicial"]**



- **Paso 1.2: Sigue los pasos que ya conoces para conseguir estos hitos:**

- 1) Crea en la nube un nuevo repositorio llamado **ED_03_REP_HIELO**

Recuerda la orden:

[git remote add origin https://github.com/USUARIO/REP]

- 2) Sube (a la nube) los archivos.

Recuerda la orden:

[git push origin master]

- **Paso 1.3: Crea también la rama develop y sube a GitHub:**

[git branch develop]

[git checkout develop]

[git push -u origin develop]



TERMINAL DE PIQUÉ

• Paso 2.2: REALIZA CAMBIOS

- 1) Crea una carpeta, dentro de ED.Practicas.GIT, llamada **03_REP_HIELO**.

- 2) Dentro de esa nueva carpeta, crea un nuevo repositorio
Recuerda la orden: **[git init]**

- 3) Identificate con tu nombre y correo

Recuerda la orden: **[git config --global user.name "xxx"]**

[git config --global user.email "xxx"]

- 4) Baja el repositorio desde la nube (**ED_03_REP_HIELO**).

Recuerda: **[git remote add origin https://github.com/USUARIO/REP]**

[git pull origin master]

- 5) Crea una nueva rama feature1 y situate en ella **[git branch feature1]**

[git checkout feature1]

- 6) En local, haz los cambios que ves a la derecha **EN EL ARCHIVO INDEX.HTML**

- 7) Haz add y commit con el texto siguiente:

Recuerda: **[git add .]**

Recuerda: **[git commit -m "cambios básicos en index.html"]**

- 8) Sube a la nube los cambios que acabas de realizar en la nueva rama.

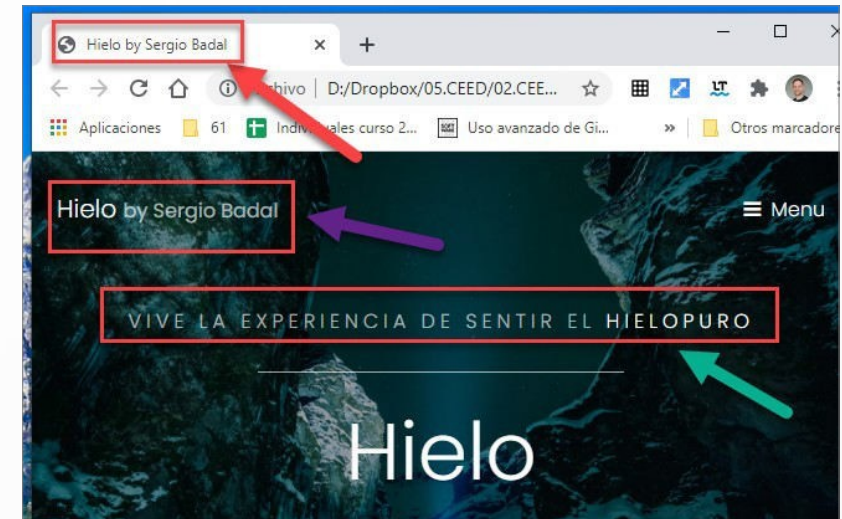
Recuerda la orden: **[git push -u origin feature1]**

- 9) Cambia en local a la rama develop, realiza un merge y sube los cambios. Después haz lo mismo con master.

[git checkout develop] **[git checkout master]**

[git merge feature1] **[git merge develop]**

[git push origin develop] **[git push origin master]**



```
<head>
<title>Hielo by Sergio Badal</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link rel="stylesheet" href="assets/css/main.css" />
</head>
<!-- Header -->
<div id="header" class="alt">
<div class="logo"><a href="index.html">Hielo <span>by Sergio Badal</span></a></div>
<a href="#menu">Menu</a>
</div>
<!-- Nav -->
<nav id="menu">
<ul class="links">
<li><a href="index.html">Home</a></li>
<li><a href="generic.html">Generic</a></li>
<li><a href="elements.html">Elements</a></li>
</ul>
</nav>
<!-- Banner -->
<section class="banner full">
<article>

<div class="inner">
<header>
<p>Vive la experiencia de sentir el <a href="https://hielopuro.com">HIELOPURO</a>
</p>
<h2>Hielo</h2>
</header>
</div>
</article>
</section>
```

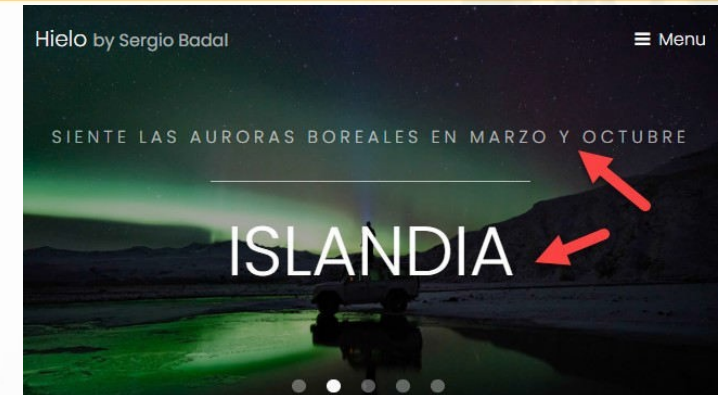




TERMINAL DE CLARA

- **Paso 2.3: REALIZA MÁS CAMBIOS**

- 1) Crea una carpeta, dentro de ED.Practicas.GIT, llamada **03_REP_HIELO**.
- 2) Ahora haz LO MISMO QUE EL PASO ANTERIOR. Dentro de esa nueva carpeta:
 - 1) Crea un nuevo repositorio.
 - 2) **Identificate.**
 - 3) Baja (**PULL**) el repositorio desde la nube (**ED_03_REP_HIELO**)
 - 4) Crea una nueva rama feature2 y posiciónate en ella
 - 5) En local, haz los cambios que te indicamos **EN EL ARCHIVO INDEX.HTML**
 - 6) Haz **add/commit** con el texto "**nuevos cambios tras reunión con el cliente**"
 - 7) Sube (**PUSH**) a la nube los cambios en la nueva rama.
 - 8) Cambia a la rama develop, master, realiza el merge y sube (**PUSH**).



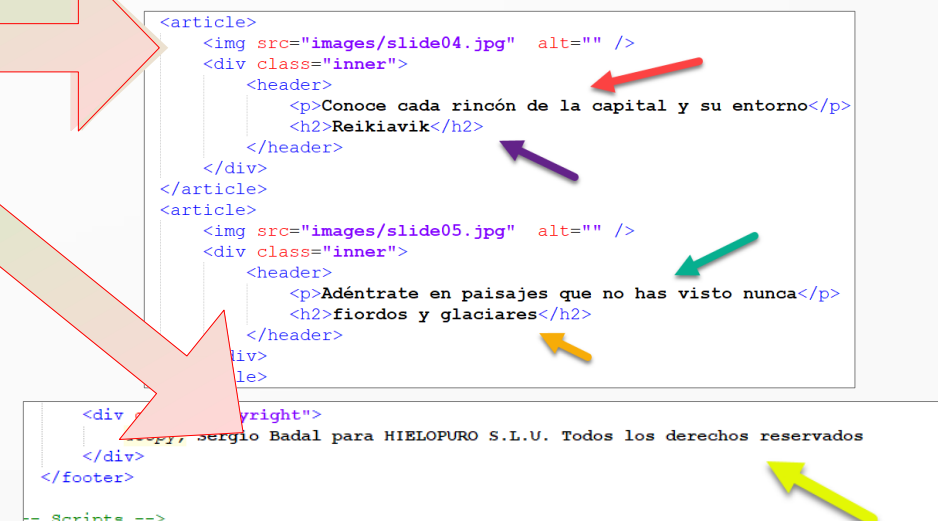
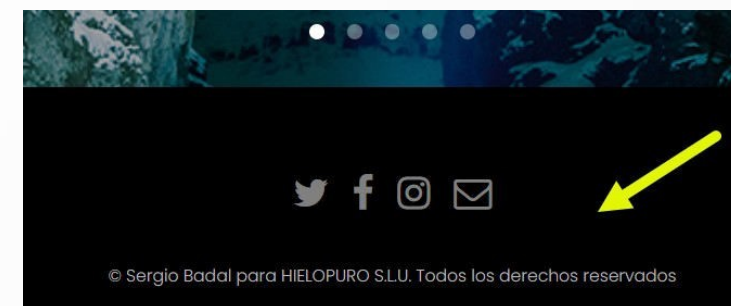
```
</article>
<article>
  
  <div class="inner">
    <header>
      <p>Siente las auroras boreales en marzo y octubre</p>
      <h2>ISLANDIA</h2>
    </header>
  </div>
</article>
<article>
  
  <div class="inner">
    <header>
      <p>Descubre paisajes increíbles</p>
      <h2>Vesturland</h2>
    </header>
  </div>
</article>
</articles>
```




TERMINAL DE SHAKIRA

• Paso 2.4: CAMBIOS DE SHAKIRA

- 1) Vuelve a la carpeta **03_REP_HIELO**. **NO ES NECESARIO QUE TE IDENTIFIQUES OTRA VEZ.**
- 2) Baja (**PULL**) el repositorio desde la nube (**ED_03_REP_HIELO**).
- 3) Crea una rama feature3 y posíciónate en ella.
- 4) En local, haz los cambios que te indicamos abajo **EN EL ARCHIVO INDEX.HTML**
- 5) Haz **add/commit** con el texto "**textos finales de Shakira**" (NO HAGAS PUSH).
- 6) En local, cambia el texto de copyright como indica la captura **EN EL ARCHIVO INDEX.HTML**
- 7) En local, añade esta línea al final del **archivo assets/css/main.css** para eliminar el faldón inferior: **#one, #two, #three {display:none}**
- 8) Haz un nuevo **add/commit** con el texto "**oculto faldón inferior y cambio copyright**"
- 9) Sube (**PUSH**) a la nube los cambios a la rama feature3.
- 10) Cambia en local a la rama develop, master, realiza el merge y sube los cambios (PUSH).





• Paso 2.5: Vamos ahora a simular un conflicto

Un conflicto ocurre cuando un usuario intenta escribir en un repositorio sobre un dato que ha sido escrito por otro usuario.

- 1) CLARA CHÍA va a cambiar el LEMA de la web:
PULL DESDE DEVELOP + CREAR RAMA feature4 + *EDITAR EL TEXTO* + ADD + COMMIT (LEMA1) + PUSH RAMA feature4 + MERGE DESDE DEVELOP + PUSH DEVELOP + PULL DESDE MASTER + MERGE DESDE MASTER + PUSH MASTER
- 2) SHAKIRA también va a cambiar el LEMA de la web, pero NO HARÁ UN PULL PREVIO:
CREA RAMA feature5 DESDE DEVELOP + *EDITAR EL TEXTO* + ADD + COMMIT (LEMA2) + PUSH RAMA feature5 + MERGE DESDE DEVELOP y ... **PUSH que será RECHAZADO POR CONFLICTO.**
- 3) Resolveremos el conflicto de la siguiente manera.

SHAKIRA:

- 1) Descargará la versión de la nube (PULL) situándose en DEVELOP antes.
- 2) Resolverá los conflictos a mano (*EDITAR EL TEXTO*) Hará ADD/COMMIT (LEMA3)
- 3) Volverá a actualizar la nube (PUSH)



```
<header>
  <p>Vive la experiencia de sentir <a href="https://hielopu
  <h2>Hielo</h2>
</header>
```

```
<header>
  <p>Siente la mirada del <a href="https://
  <h2>Hielo</h2>
</header>
```

```
<<<<<< HEAD
      <header>
      <p>Vive la experiencia de sentir <a href="https://
      HIELOPURO</a></p>
      =====
      <p>Siente la mirada del <a href="https://hielopur
      </a></p>
>>>>>> 9a4ebdecba589b12371ba169defaf3aee521a1fb
      <h2>Hielo</h2>
      </header>
```

```
<header>
  <p>Vive la experiencia y siente la mirada del <a href="http
  <h2>Hielo</h2>
</header>
```