



INFORME TÉCNICO

PSG2 G6-62

Índice de contenidos

1. INTRODUCCIÓN.....	2
2. ESTRUCTURA DE REPOSITORIOS Y RAMAS POR DEFECTO	3
3. ESTRATEGIA DE RAMIFICACIÓN, BASADA EN GIT FLOW.....	4
4. POLÍTICA DE VERSIONES	5
5. CONCLUSIONES	7

1. INTRODUCCIÓN

En esta segunda parte del proyecto vamos a describir la metodología de gestión utilizada por el grupo. Usaremos GitHub para la gestión de tareas. En los siguientes puntos se explicará de forma detallada su estructura y la evolución que hemos tenido hasta ahora.



Figura 1: Icono GitHub

2. ESTRUCTURA DE REPOSITORIOS Y RAMAS POR DEFECTO

La primera estructura que utilizamos para nuestro repositorio fue la más simple, pero a su vez, la que puede dar origen a más problemas. Cada uno de los integrantes del grupo trabajamos sobre la misma rama 'master', lo que pudo ocasionar serios problemas de conflictos. Gracias a que algunos miembros del grupo ya usaron GitHub, pudimos resolver estos problemas de antemano. Para la prevención de estos conflictos decidimos no trabajar en paralelo. De esta forma, pudimos evitar que dos o más trabajos se pisen entre ellos.

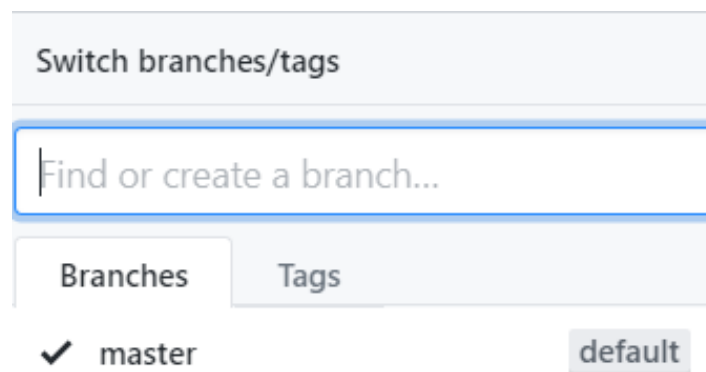


Figura 2: Rama master

3. ESTRATEGIA DE RAMIFICACIÓN, BASADA EN GIT FLOW

La segunda estructura o metodología que utilizamos fue Git Flow. Esto es una estrategia de ramificación que permite una mejor organización del trabajo y la prevención de conflictos. Esta estructura parte de una rama principal 'master' en la que sale una rama secundaria 'develop'. Cada tarea del proyecto corresponde una rama 'task-#xx'. Estas ramas son creadas a partir de la rama secundaria. Cuando una tarea es acabada se hace un merge hacia la rama secundaria. Esto es, el trabajo de la task-#xx se sube a la rama develop. Así podemos ir uniendo todo el trabajo. Cuando todas las tareas están subidas a develop se comprueba que todo funcione a la perfección. Si todo va bien, entonces podemos proceder al último paso: hacemos un merge de develop a la master.

Con esta estructura podemos trabajar simultáneamente desde diversas ramas. Además, podemos evitar que se nos cuele algún error en la rama principal, el cual puede ser muy engorroso de resolver o que, en el peor de los casos, no se pueda ni resolver.

Repositorio del grupo G6-62.

Edit

[Manage topics](#)

19 commits

11 branches

0 packages

0 releases

3 contributors

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

Switch branches/tags

Find or create a branch...

Branches

Tags

default

master

Develop

Task-#10

Task-#11

Task-#12

Task-#13y14

Task-#18

Task-#21

Task-#22

Task#13

Task#15

Latest commit 549fa33 8 days ago

commit	22 days ago
commit	22 days ago
branch 'master' into Task-#18	10 days ago
commit	22 days ago
e) 'Add a translation of the messages in Spanish' finished	19 days ago
commit	22 days ago
Problemas	8 days ago
commit	22 days ago
commit	22 days ago
commit	22 days ago
commit	22 days ago
...	...

Figura 3: Conjunto de ramas

Nota: Podríamos haber creado una rama por componente del grupo, pero al final nos decantamos por una rama por tarea. Creemos que a mayor número de ramas menor número de conflictos.

4. POLÍTICA DE VERSIONES

La primera versión del proyecto correspondería a la primera estructura del proyecto, en la cual usamos una única rama master. En cambio, para la segunda versión ya usaríamos la metodología Git Flow.

La siguiente gráfica muestra el número de commits realizados a la rama master. Como se puede apreciar, hay un descenso después del cambio de metodología usada por el grupo. Recordemos que para Git Flow la mayoría de los commits se hacen a las task-#xx y posteriormente al develop. Así que, dentro de un conjunto de actividades, sólo sería necesario realizar un commit a la master.

Feb 16, 2020 – Mar 16, 2020

Contributions: Commits ▼

Contributions to master, excluding merge commits

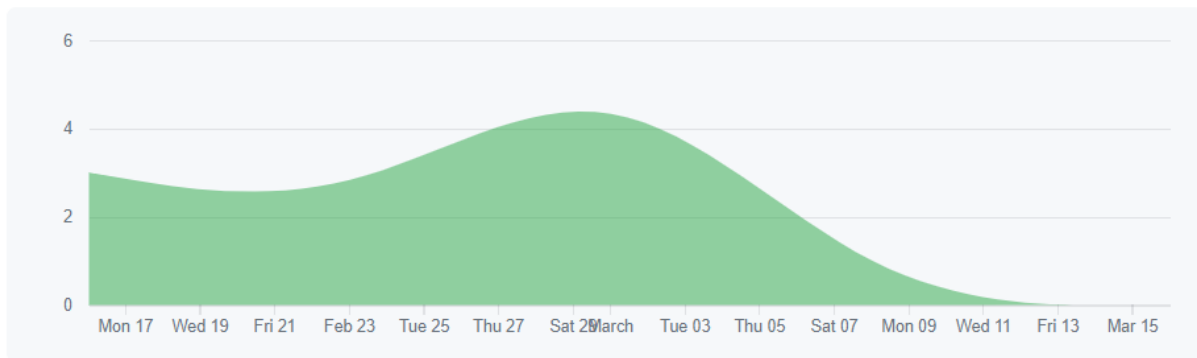


Figura 4: Commits a la master

Añadiremos una segunda gráfica que haría referencia a la estructura de ramas seguida en el proyecto. Como podemos observar, los cuatro primeros días estuvimos trabajando sólo con la rama **master** (1ª estructura). A partir del 23 de febrero, pasamos a usar Git Flow (2ª estructura). Se puede ir apreciando como se van creando **task-#xx** que finalmente acaban en la rama **develop**:

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

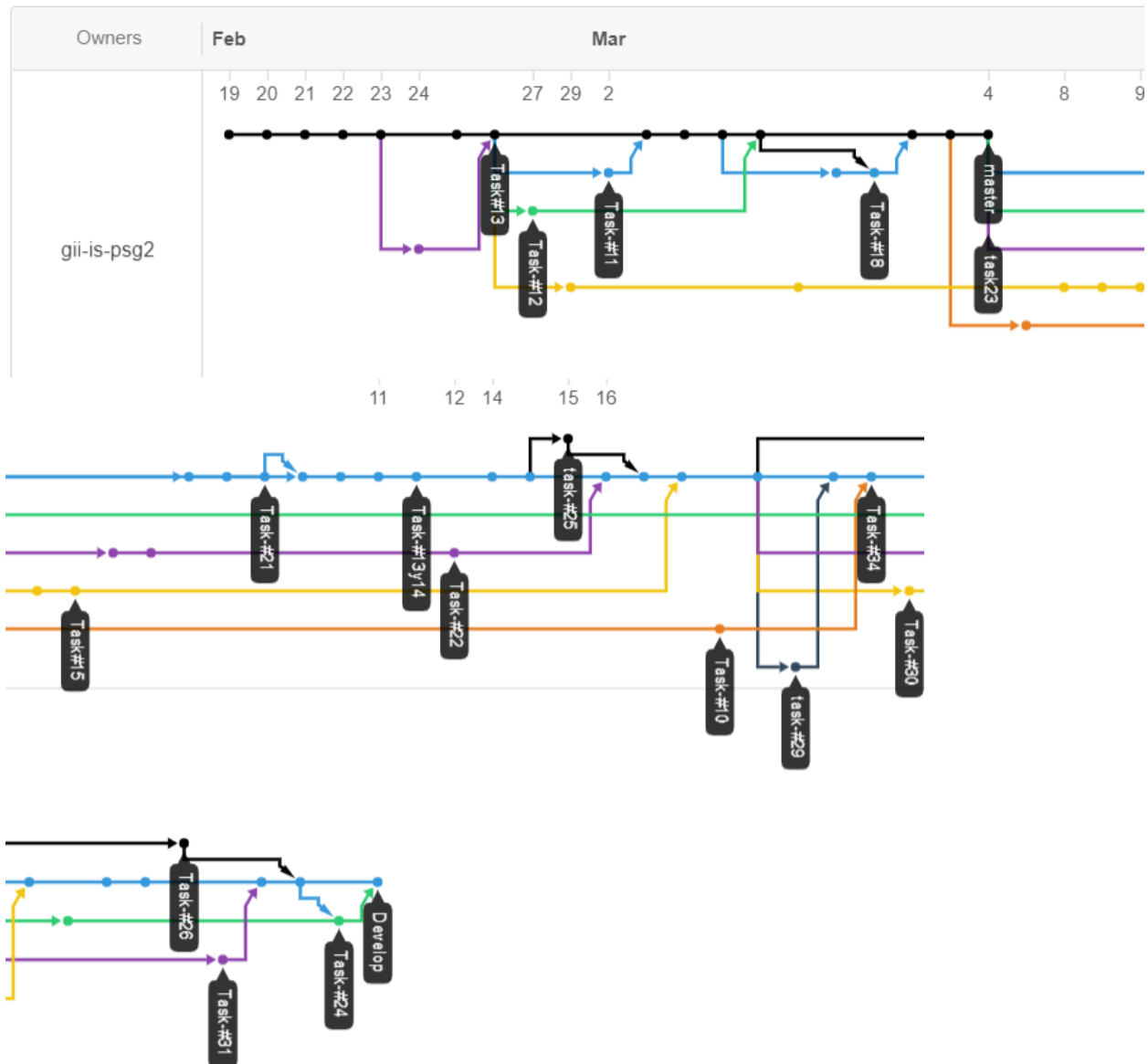


Figura 5: Estructura de las ramas

Nota: Cabe mencionar que la rama develop será unida a la master. Como todavía estamos trabajando, este enlace no se puede apreciar en la figura 5.

5. CONCLUSIONES

En vista a la metodología de gestión, el grupo pudo prevenir los primeros problemas comentados en el segundo punto. Otros problemas que surgieron es que algunos compañeros crearon ramas a partir de la master, en vez de develop. Estos problemas pudimos solventarlos creando otra rama a partir de develop y pasando el trabajo de la primera rama a ésta última. En general, ha habido una buena organización y madurez en la resolución de problemas.