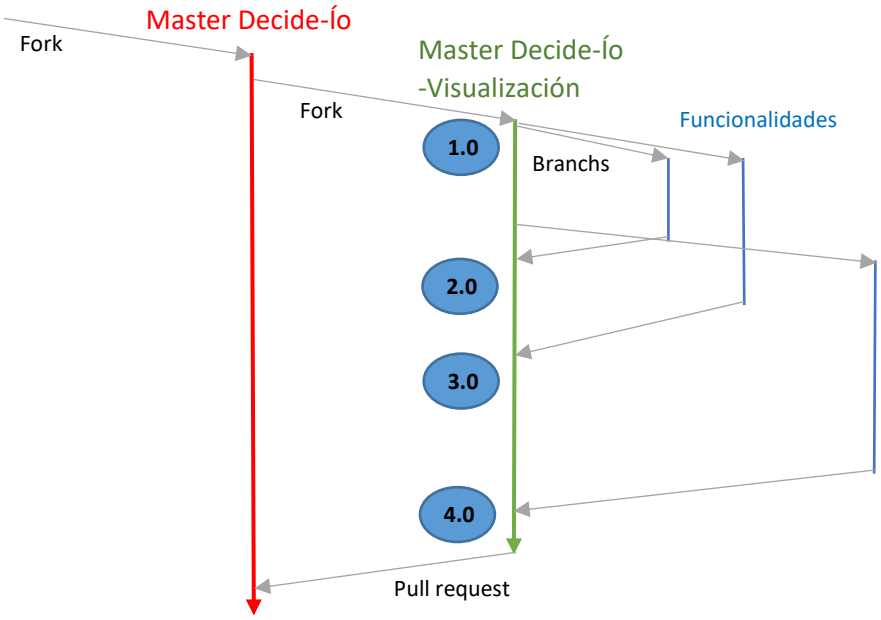
Proceso de gestión del código

Git flow



Del fork de Decide de la asignatura EGC se ha hecho otro fork para el grupo Decide-Ío. Ese fork, común para los 7 subgrupos será gestionado por el coordinador de cada subgrupo, es decir, por 7 coordinadores.

Del fork de Decide-Ío se hará un fork para cada subgrupo, por lo que habrá 7 forks en total. Cada uno de esos fork será gestionado internamente por el grupo y será el coordinador del grupo el encargado de realizar el pull request con el fork Decide-Ío cuando sea oportuno y de gestionar dicha integración.

A partir del fork Decide-Ío-Visualización vamos a crear ramas para cada una de las funcionalidades, es decir, si decidimos ampliar los tipos de gráficas del módulo de visualización, se creará una rama para dicho desarrollo. Una vez terminado el desarrollo y una vez realizadas las pruebas oportunas, el desarrollador será el encargado de gestionar la integración de su rama con la rama master del repositorio.

Puesto que la integración de ramas implica nuevas funcionalidades, el versionado seguirá el patrón 1.0, 2.0, 3.0, etc.

En el caso de que se añadan desarrollos en la rama master que no impliquen nuevas funcionalidades, el versionado solo afectará a la parte decimal, es decir, 1.1, 1.3, 1.35, etc. Dependiendo de si el cambio es significativo o no, la numeración afectará a la décima o a la centésima.

Commits

Únicamente se realizarán commits cuando se cumpla alguna de las siguientes condiciones:

- Se ha finalizado y testeado una nueva funcionalidad
- Se ha llegado a un punto en el desarrollo en el que el sistema es estable y puede funcionar, aunque la funcionalidad no esté finalizada
- Se ha solucionado alguna incidencia
- Se ha solucionado algún error, aunque no exista una incidencia relacionada
- Se ha refactorizado y testeado el código
- Subida de documentos o algún otro recurso

La base fundamental de los commits es que el código del repositorio debe estar siempre estable y puede desplegarse en cualquier momento, aunque algunas funcionalidades no estén terminadas.

Los commits deben seguir la siguiente plantilla (esta subida al repositorio para ser usada):

```
# <Tipo> Tipo de commit. <Asunto> Resumen del commit. (Max 50 caracteres)
# Ejemplo: FIX. Solución incidencia #21

# Explicar porque se ha realizado el cambio:
# |<---- Explicación de porqué el cambio realizado es necesario. ¿qué? ¿por qué? ¿dónde? ¿cuándo?
# Intenta crear párrafos cortos y concisos (80 ~ 120 caracteres por párrafo). ---->|

# Proporciona enlaces o referencias que puedan ayudar a contextualizar el commit.
# Ejemplo: Incidencia #21
# Ejemplo2: tarea con nombre "..." de Trello.

# --- FIN DEL COMMIT ---
# los Tipos pueden ser:
# FEAT (new feature)
# FIX (bug fix)
# REFACTOR (refactoring production code)
# STYLE (formatting, missing semi colons, etc; no code change)
# DOCS (changes to documentation)
# TEST (adding or refactoring tests; no production code change)
# --------------------
```