

Reporte técnico sobre la política de ramas

1. Estructura de repositorios y de ramas por defecto

Para nuestra política de ramas seguiremos la directrices impuestas por Git Flow, por lo tanto, tendremos dos ramas principales:

-Master: Es la rama fundamental donde se encuentran las distintas versiones “preparadas para la producción” del proyecto.

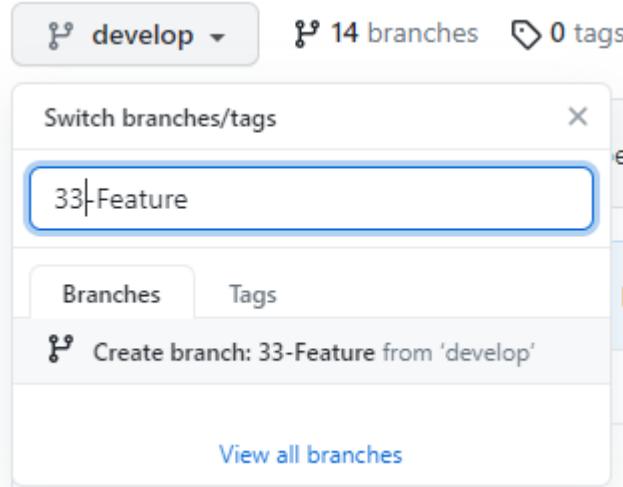
-Develop: Es la rama donde se encuentra los últimos cambios realizados por el DT (es la rama donde se hace “merge” de las distintas features finalizadas).

También, contamos con las ramas “feature branches” donde se realizarán las distintas tareas para posteriormente hacer “merge” con la “develop branch”, las ramas de “hot-fix” donde se arreglarán los bugs imprevistos y la rama de “releases” la cual se encarga de la preparación y consolidación de las “release”.

2. Estrategia de ramas

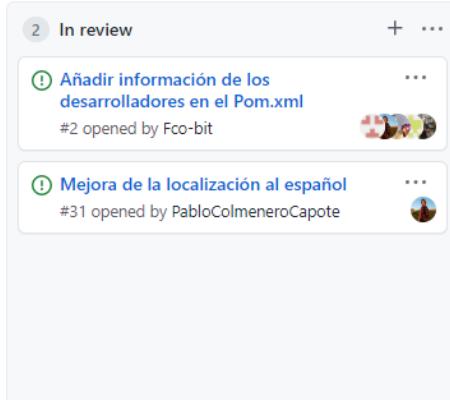
2.1 Como desarrollar las “features branches”

Las “features branches” se van creando a medida que se van desarrollando las tareas (es decir, cuando se va a realizar una nueva tarea se crea una rama de feature para dicha tarea).



Ejemplo de creación de Rama 1

Una vez la tarea de una “feature branch” se da por finalizada tarea se lleva a “In review” se realiza un “pull request” con la rama “Develop” (El encargado de hacer dichas pull requests es un miembro concreto del Scrum Team denominado “manager”, en nuestro caso es: David Barragán Salazar).



Tareas Finalizadas por sus autores.

A screenshot of a GitHub pull request creation interface. At the top, it says 'base: develop' and 'compare: 37-Feature'. It also indicates that the branches are 'Able to merge'. The title of the pull request is '37 - Service Test, Controller Test fixed and functionality fixed PR'. The body of the pull request contains the text: 'Almost every controller were wrong due to the fact that the new Services and security, Now are fixed. Problems with the find owner functionality and add visit are fixed.' There is a note at the bottom: 'Attach files by dragging & dropping, selecting or pasting them.' A green button at the bottom right says 'Create pull request'.

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Ejemplo de Pull request con develop.

Pd: No se borrará la rama feature una vez terminada la tarea y realizado el pull request.

2.2 Como preparar las “releases” de las distintas versiones

Para la preparación de las releases haremos uso de la rama de “releases” ya que esta rama es la correspondiente para realizar la preparación las releases. En esta rama se irán arreglando los bugs de la release antes de su “merge con master”, para intentar así que la versión “definitiva” que estará en master esté libre de bugs.

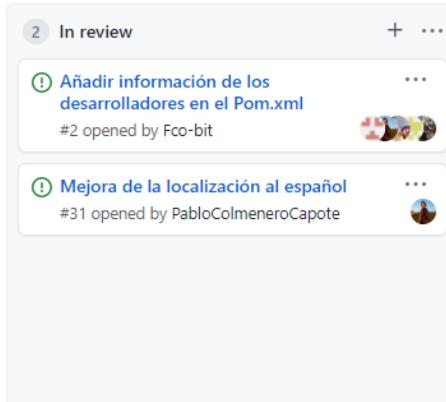
2.3 Como realizar los “hot fix” una vez el proyecto está en producción

Los hot-fix son aquellos que se realizan para arreglar bugs que existen en la versión que ya está en master y se daba como terminada. En el caso de encontrar un bug en una versión definitiva master se hará uso de la rama “hotfix” ya que esta es la que especifica Git Flow para arreglar estos problemas. Se arreglará el bug de la versión en dicha rama y una vez arreglado dicho bug, se hará un merge de dicho arreglo a la rama “development” y a la rama “master”.

2.4 Peer reviews

Cada tarea del proyecto debe ser revisada por otro/s miembro/s del proyecto antes de hacer “commit” de los cambios. Ya que se esta forma hay menos posibilidades de errores , facilita una comprensión general del proyecto a todo los miembro del DT y promueve el uso de estándares en el grupo.

Para asegurar el cumplimiento de esta práctica, tenemos la restricción de que la tareas nunca podrán ser desplazadas desde “In-Review” a “Done” por el mismo autor de dicha Tarea.



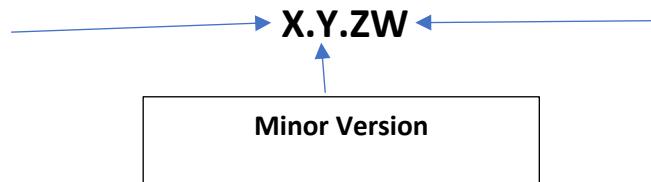
Tareas Finalizadas por sus autores 1 (y no revisadas).

3. Políticas de versionado

Para explicar nuestra política de versionado primero debemos explicar la semántica de versionado que vamos a usar durante el proyecto. La semántica que vamos a usar va ser la siguiente:

Major Version

Patches



Major Version: Solo para cambios mayores. 0.Y.ZW para el desarrollo inicial. 1.0.0 define la primera API pública.

Minor Version: Cambios menores. Incluye mejoras y nueva funcionalidad. Puede (y suele) incluir parches.

Patches: Arreglos de bugs. Arreglos internos para arreglar posible bugs/ comportamientos anómalos.

Reglas del versionado:

-Cuando la “Major version” se incrementa se reinician el resto de componentes del versionado: 1.02.34 → 2.0.0.

-Cuando la “Minor version” se incrementa la versión de los parches se reinicia:

1.0.67 → 1.1.0.

4. Estándar nomenclatura para el nombrado de ramas, pull request.

Nuestra nomenclatura estándar para el nombrado de estos elementos seguirá estas directrices:



Ejemplo de la nomenclatura PR estándar (PR son siglas de Pull Request).

Ramas (Feature)

► **Id-Feature**

Switch branches/tags X

Find or create a branch...

Branches Tags

2-Feature

2_Feature

10-feature

11-Feature

12-Feature

13-Feature

14-Feature

23-Feature

24-feature

27-Feature

Versión	Fecha
1.0	19/03
1.1	05/04