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 projecto.

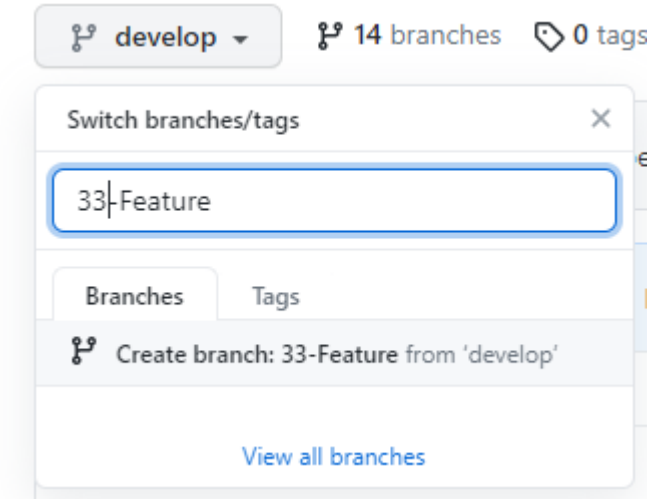
-Develop: Es la rama donde se encuentra los últimos cambios realizados por el DT (es la rama donde se hace “merge” de las distintas features finalizadas).

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

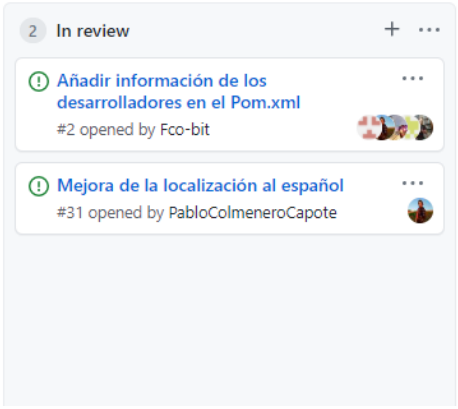
2.1 Como desarrollar las “features branches”

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).

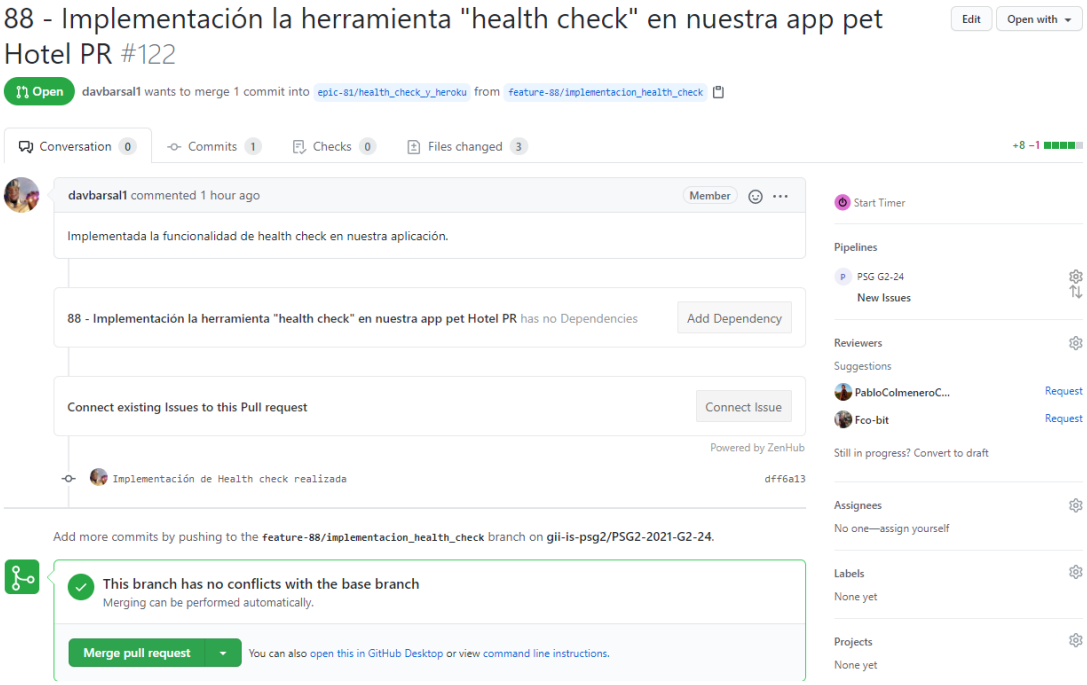


Ejemplo de creación de Rama 1

Una vez la tarea de una “feature branch” se da por finalizada tarea se lleva a “In review” se crea un “pull request” con la rama de su épica correspondiente, esta pull request no se realizará hasta que dicha tarea sea revisada por el ST (si la revisión encuentra algun fallo y/o posible mejora dicha tarea volverá a “In progress” para aplicarle dichos cambios).



Tareas Finalizadas por sus autores.



Ejemplo de Pull request de feature con su rama épica.

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

2.2 Como desarrollar las “Epic branches”

Las ramas épicas son ramas que se representan cada una de las épicas del sprint, a medida que las tareas de una épica se vayan realizando la rama épica se irá actualizando mediante las pull request de las ramas features.

Una vez todas las features de la épica han sido realizadas y mergeadas en esta rama épica, se hare un pull request de la rama epica a develop actualizando así develop con dicha épica terminada/implementada.

2.3 Como preparar las “releases” de las distintas versiones

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

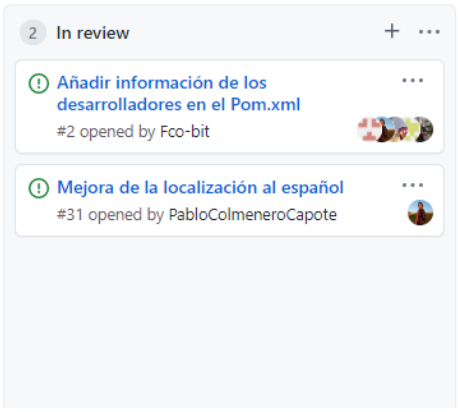
2.4 Como realizar los “hot fix” una vez el projecto está en producción

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 projecto 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 projecto a todo los miembro del DT y promueve el uso de estándares en el grupo.

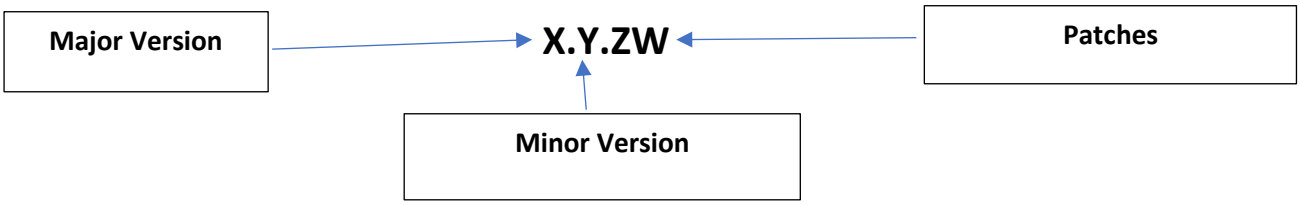
Para asegurar el cumplimiento de esta práctica, tenemos la restricción de que la tareas nunca podrán ser desplazadas desde “In-Review” a “Done” por el mismo autor de dicha tareaa, al igual que las pull request de dicha tarea con su rama épica tampoco podrá ser realizado por el mismo autor de la tarea (excepto claro que la tarea sea una tarea común de grupo).



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

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: Solo para cambios mayores. 0.Y.ZW para el desarrollo inicial. 1.0.0 defime 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 componenetes del versionado: 1.02.34 → 2.0.0.

-Cuando la "Minor version" se incrementa la version de los parches se reinicia:

1.0.67 → 1.1.0.

Versionado de documentos:

El versionado de documentos se hará de la siguiente forma:

-Se creará una tabala de versionado en el documento la cual contará con tres columnas: versión, fecha y autor (título autoexplicativo del contenido que se escribe en cada columna).

-La política de versionado es parecida a aquella explicada para el código:

+**major version**: reforma practicamente todo el documento.

+**Minor version:** addición de un/varios apartados nuevos en el documento.

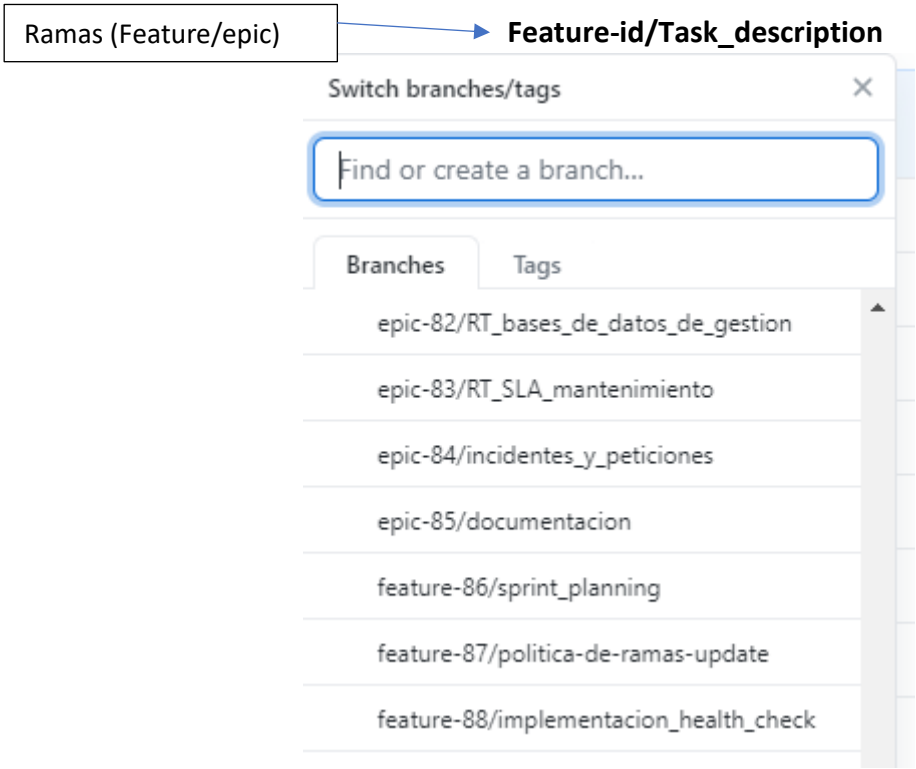
+**patches:** Cambios de ortografía y/o arreglos ene el leguaje formal.

Nota: Véase la tabla de versionado de este documentos como ejemplo.

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).



Nota: para las ramas épicas la nomenclatura será igual que solo que la palabra ”feature” se substituye por “epic”.

5. Asignación de tareas

Hemos llegado al conseso de que el reparto de tareas se hará de forma semanal, de esta forma conseguimos un método de asignación más flexible (e.g: en el caso de encontrar descompensaciones de trabajo a lo largo del sprint podemos arreglar dicha descompensacion de trabajo a la semana siguiente).

Además este modo de asignación es completamente compatible con nuestra idea de ir proponiendo objetivos semanales para incitar/ayudar a trabajar de forma más continua y progresiva.

Autor	Versión	Fecha
Francisco Rodríguez Pérez	**1.0.0**	**19/03**
Francisco Rodríguez Pérez	**1.1.0**	**05/04**
Francisco Rodríguez Pérez	**1.1.1**	**06/04**

Francisco Rodríguez Pérez	**1.2.0**	**15/05**