Proceso Git Flow

Proceso de Git Flow

1) Por cada uno de los tickets, el desarrollador deberá crear un nuevo branch tomando como base el branch **Develop**, en cual trabajará toda la resolución del ticket. **Paso del equipo de desarrollo.**

2) Una vez este Ok desde desarrollo, el desarrollador deberá publicar un PR, para ser evaluado por el resto del equipo. Aquí inicia un proceso de revisión,donde el equipo deberá plasmar todas sus dudas y/o sugerencias de corrección.
   Para ellos tener en cuenta las siguientes consideraciones:

   - Siempre intentar hacer PRs los más chicos posible, en general si hay capas en el proyecto, un PR por capa suele ser un buen balance. La idea de esto es poder hacer más amenos los reviews, y que no lleve tanto tiempo el ida y vuelta de feedbacks. También ayuda a tener una revisión anticipada de un approach, y poder ajustar sobre la marcha sin necesidad de que todo esté completo y después haya que cambiarlo.

   - Tratar de tener una buena cobertura de tests en cada Pull requestque se crea: se debe tener en cuenta que al crear un pull request, no hayamos olvidado de cubrir casos importantes, y cuando se revise códigos asegurarnos de hacer seguimiento de ese punto también, y de dar puntos de vista sobre casos que sería interesante testear.

   - Hacer uso del self code review (antes o después de crear el PR), puede ayudar ver el código desde una perspectiva más general, como reviewer, más allá de ser el autor del código. Ejemplo: veo en los cambios que hay código que se repite, ahí puedo considerar armar un private method y reutilizarlo en los lugares donde repetí código. Releo el código y descubro un typo. **Esto es simplemente una validación más, el review recibido debe ser por parte del resto del equipo.**

   - Más allá de aprobar un pull request que revisé, tratar de participarde una manera más activa, ya sea *sugiriendo algún cambio*, o *preguntando algo que no está tan claro en el código*, o *bien marcando algo que no conocía y que me parece interesante cómo se aplicó en esa oportunidad*. Esto ayuda a tener una noción más amplia del codebase del proyecto en el que se está trabajando, ir incorporando prácticas o incluso para tener en cuenta porciones de código que podrían ser útiles en una task que toque implementar a futuro. Tener la confianza de tener la oportunidad de comentar y sacarse dudas.

   - Respecto al seguimiento de comments en pull requests, siempre está bueno responder a todos los comments que nos hacen en los reviews, tanto si vamos a aplicar el cambio (indicando en cuál commit se aplicó, ayuda mucho pegar el link del commit) o si no lo vamos a

aplicar, explicar las razones, para que se genere un ida y vuelta más

rico. Siempre revisar antes de mergear que no haya quedado algún comment sin tener en cuenta, además de los approvals.

3) Una vez el PR es aprobado por al menos 2 devs (depende el numero de integrantes del equipo), inicia la etapa de prueba funcional por parte del cliente. El mismo deberá poder acceder al PR y aprobarlo.

ACLARACION: El step anterior del proceso es realizado exclusivamente por el cliente.

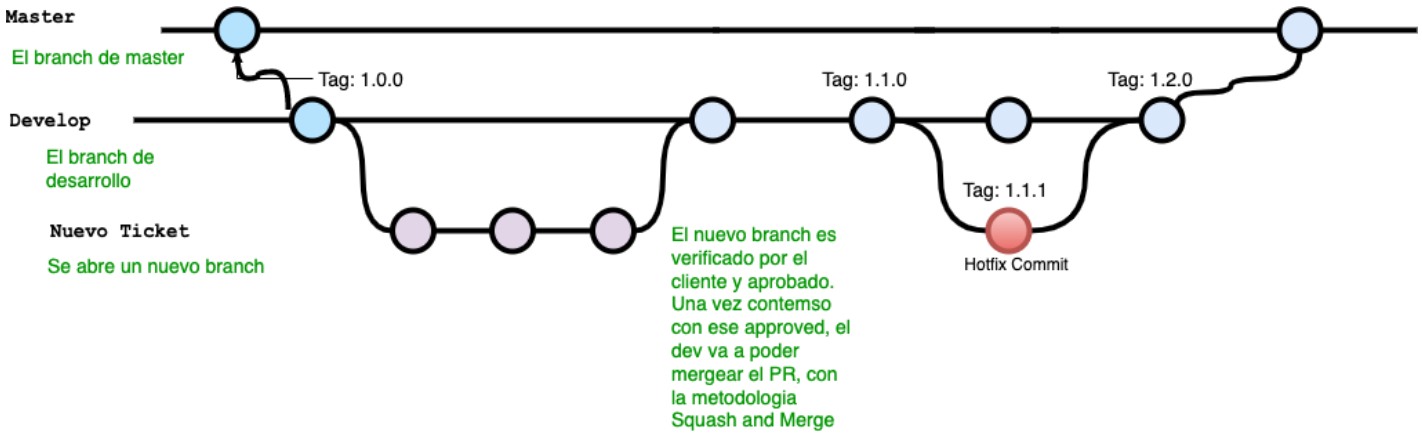
4) Una vez recibido el **Ok funcional del cliente**, se procederá a mergear el PR en forma de **SquashAndMerge (Dicha opción permite encapsulas todos los commits en uno solo, y así mantener un historial más limpio del repositorio).**

Consideraciones:

a. Antes de mergear, asegurarse de que sobre el branch creado se tiene la última versión de **Develop, para ello deberán rebasear el branch nuevamente.**

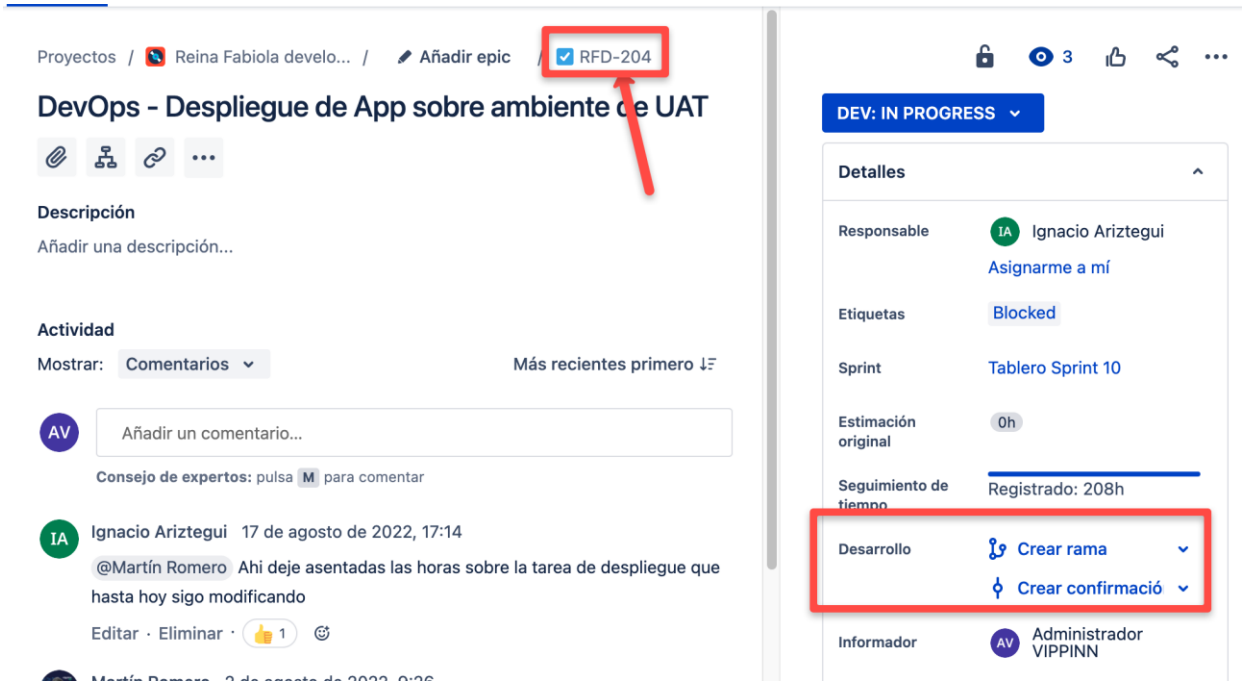
b. Comunicación con el equipo para realizar estas acciones de forma coordinada.

c. Resolución de conflictos: por más que la va a realizar el mismo que escribió el PR, en caso de conflictos a rebasear, involucrar a los demás Dev para evitar errores de merge.



Nomemclatura de los Branchs y commits

Este aspecto a la hora de crear los PRs es clave para asegurar la trazabilidad de Jira con GitHub, dado que de esta forma Jira interpreta que se ha creado un PR o commit en relación a un ticket en específico.



El identificador del ticket (RFD-204, en el ejemplo), es el id que vamos a utilizar para nombrar brachs que vamos a crear por cada uno de ellos. La nomenclatura a seguir es la siguiente:

ID-#-Descripción del ticket

Por ejemplo, para el ticket anterior, se debería crear un branch nombrado como ***"RFD-204-Despliegue-de-App-sobre-UAT".***

Las descripciones de los commit tiene una nomenclatura similar:

[ID-#] - Descripción de los cambios realizados

Por ejemplo, para el ticket anterior, se debería crear un commit nombrado como ***"[RFD-204] - Generacion de un nuevo Build v1".***