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) y 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".

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 las rama de "releases" ya que esta rama es la ccorrepondiente 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.

2.3 Como realizar los "hot fix" una vez el projecto está en producción

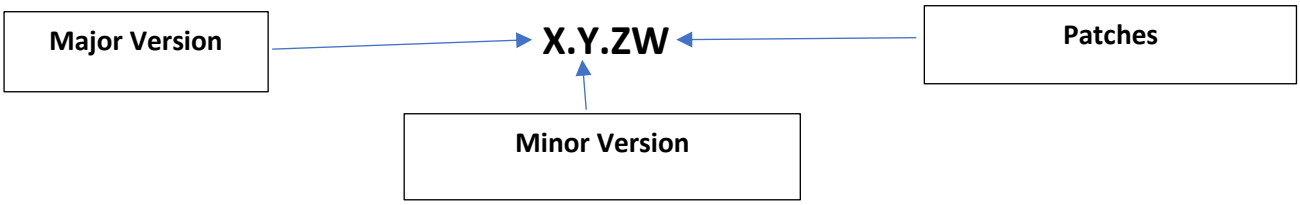
Los hot-fix son aquellos que se realizan para arreglar bugs que existen en la versión que ya está en master y se daba como terminada. En el caso de encontrar un bug en una versión definitiva master se hará uso de la rama "hotfix" ya que esta es la que especifica Git Flow para arreglar estos problemas. 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.

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.

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

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