Informe de la configuración de la metodología de administración

# Estructura de los repositorios y las ramas por defecto

Los documentos se van a encontrar en la carpeta src/main/resources/docs habiendo una carpeta de documentos por cada sprint a realizar. La aplicación irá también en la carpeta src/main/java

Vamos a seguir la metodología de GitFlow para el desarrollo, por lo tanto, vamos a tener dos ramas por defecto. La rama máster es la rama con el producto estable en producción, y la rama develop es la rama donde están las nuevas funcionalidades que aún no están en producción.

# Estrategia de ramas, basada en Git Flow e incluyendo la revisión por pares

## Cómo desarrollar las ramas de funcionalidades

Se va a crear una rama a partir de la rama develop, la convención de nombres será

feature/x-nombre-separado-por-guiones

Donde x es el número de la tarea y nombre es el nombre de dicha tarea. Una vez terminado de desarrollar dicha funcionalidad se hará un push de la rama, se creará una pull request de dicho push y se asignará un peer reviewer.

La tarea se moverá a la columna In review y se le asociará la pull request correspondiente. Una vez el peer reviewer dé el visto bueno se hará merge de dicha pull request y se moverá la tarea a done cerrándola.

## Cómo preparar la versión de producción

Una vez terminadas las funcionalidades y añadidas a la rama de develop, se va a crear una nueva rama con el nombre de release/x.y.z donde x.y.z es la versión del producto. En esa rama se va a preparar el repositorio para la puesta en producción, añadiendo información metadata necesaria y arreglos de bugs menores.

Una vez terminada la rama de release se va a hacer merge con la rama de máster.

## Cómo arreglar los bugs en producción

Si hay algún error crítico en producción que requiera reparación inmediata se va a crear una rama a partir de master con el nombre hotfix/x.y.z siendo x.y.z la versión siguiente a la versión de producción. Una vez que se han resuelto los errores se vuelve a hacer merge con la rama de master.

# Política de versionado

Para el versionado de nuestro proyecto seguiremos la nomenclatura recomendada de GitFlow:

Versión X.Y.Z-Label donde X representa cambios mayores en el proyecto y rompe con la compatibilidad de versiones anteriores, Y representa cambios menores, pero no provoca incompatibilidades de versiones anteriores, Z que denotará pequeños cambios para corrección de errores y *Label* que representará versiones previas a la publicación.

Cualquier cambio en una versión posterior provocará el reinicio de la numeración de sus posiciones anteriores, por ejemplo. Si pasamos de 1.9.2 a 1.10.0 o 2.0.0.