Administración de Votaciones

Grupo 1 - Evaluación y Gestión de la Configuración

El módulo de Administración de Votaciones, por decisión conjunta de sus miembros, establece las siguientes normas para la gestión de incidencias y de versiones de código:

GESTIÓN DE INCIDENCIAS

Cada incidencia tratará un problema relacionado con el proyecto, ya sea organizacional o de código, o bien de una tarea. Se tendrá una incidencia única para cada problema. Si una incidencia se repitiese o tratase varios temas a la vez, se etiquetaría como “descartado” y se desecharía. Además, se intentará redactar la incidencia de manera concisa y detallada, evitando ambigüedades en la descripción.

Para cada incidencia se podrán asociar los labels o etiquetas que se definen a continuación:

- **API**
- **Base de Datos**
- **Bugs**
- **Descartado**
- **Documentación**
- **Duda**
- **Failure**
- **Investigación**
- **No Tocar**
- **Nuevo**
- **Requisitos**
- **Reunión**
- **Urgente**
- **Visual**
- **Accepted**
- **Código**

Por último, deberá asociarse al Milestone del proyecto correspondiente (Del 2 al 4).

GESTIÓN DE VERSIONES

La gestión de versiones está predeterminada por el grupo de integración. Según este modelo, las versiones serán de la forma X.X, dependiendo de las funcionalidades y los bugfix realizados. Un ejemplo claro en nuestro caso sería:

Versión 0.0 - Código inicial heredado.

Versión 1.0 - Código con la BBDD MariaDB implementada.

Versión 1.1 - Bug de la aplicación detectado y corregido

GESTIÓN DE CÓDIGO

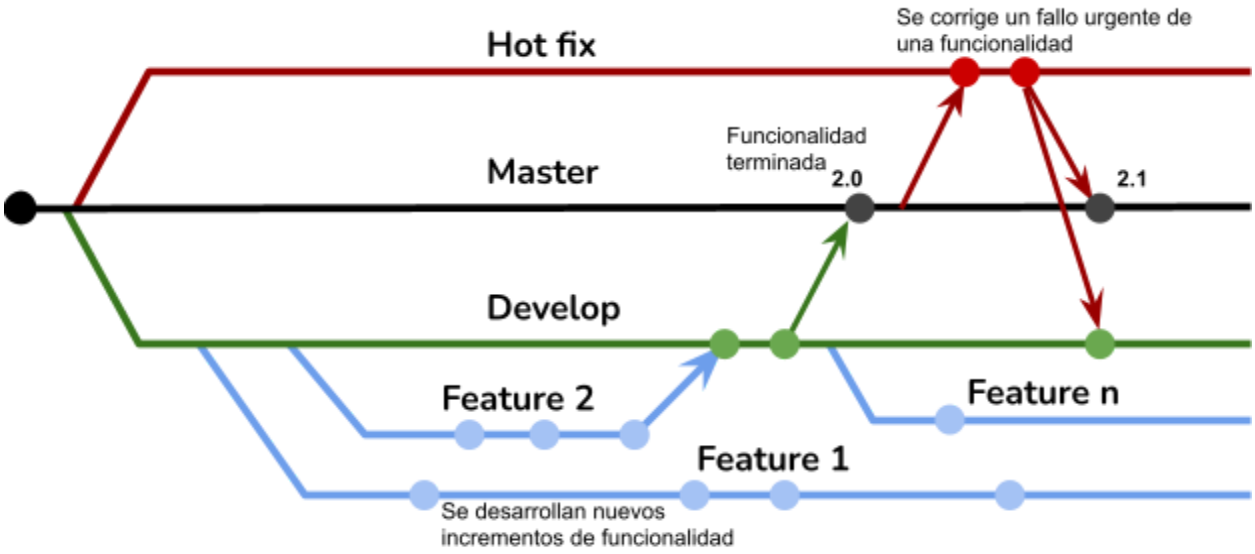
El código se gestionará en todos los módulos mediante la herramienta Git. Para ello, hemos de definir distintos parámetros: Qué modelo de trabajo se seguirá en el repositorio, qué es un commit y cuándo se realiza.

Modelo de uso

El modelo de uso de este módulo intentará ser una versión simple del método GitFlow. Mientras que en este conocido método se tiene tanto una rama máster como una release, una develop y una de hotfix, en nuestra versión, ajustada a un proyecto tan pequeño como el nuestro, la rama de release queda eliminada y sólo se distingue separación entre desarrollo, master, y hotfix.

Si fuese necesario arreglar un fallo urgente en una versión completa, se usará la rama hotfix y se añadirá un tercer dígito X.1, que irá aumentando por cada “hotfix” que se deba realizar.

No existe un número concreto de ramas asociadas a estos incrementos, pero se pretende que cada incremento esté asignado principalmente a una sola persona (aunque no sea exclusivo y cualquier otro miembro del equipo pueda hacer checkout de esa rama para ayudar y/o completar ese incremento de forma puntual)



Commits

El trabajo de desarrollo en local se realizará en las ramas de incremento (o feature). En ellas se guardará nuestro trabajo mediante el uso de commits **atómicos**. Los commits son un conjunto de "changesets" o cambios en el repositorio. Que sean atómicos quiere decir que cada commit se refiere explícitamente a una tarea en el código. Algunos ejemplos de tareas unitarias son: Creación de una clase, corrección de un bug o un issue, añadir un conjunto de métodos relacionados entre sí etc.

Como mensaje asociado al commit usaremos una estructura común:

Plantilla:

<Tema>
<LÍNEA EN BLANCO>
<Cuerpo>
<LÍNEA EN BLANCO>
<Pie de mensaje>

Ejemplo:

Creación de la vista A

La vista A incluye un dropdown y un input de datos.

Closes #5

Nótese que en el pie de mensaje hay mensajes con palabras reservadas como "Closes #X" que tienen un efecto directo en las incidencias del repositorio. En el proyecto se usará tanto "Closes #X" como "Issue #X" para hacer referencia a la issue a la que nos referimos.

Los títulos comenzarán con un **sustantivo** que defina la acción que se realiza.

INTEGRACIÓN CONTINUA

Para la integración usaremos TravisCI en nuestro repositorio de GitHub.

IMPORTANTE: Todos los commits relativos a los ficheros de configuración podrán ser actualizados directamente en la rama Máster:

- Dockerfile
- travis.yml