Universidad ORT Uruguay
Facultad de Ingeniería

Ingeniería de Software Ágil 2

Guía de “Creación y posterior mantenimiento del repositorio: elementos que contiene y cómo los van a versionar”

Clavijo, Tomás (235426)

Barreto, Sofía (258216)

Facello, Bruno (260694)

Docente: Alvaro Ortas

2023

Índice

Introducción

Este documento reúne la información correspondiente a la creación y posterior mantenimiento del repositorio, elementos que contiene y cómo los vamos a versionar. Brindando información sobre los conocimientos en los que nos basamos para la toma de decisiones y las tecnologías que utilizaremos para lograr los objetivos.

Repositorio

El proyecto estará respaldado en un repositorio de GitHub, permitiéndonos trabajar de manera colaborativa y versionada.
El mismo no tendrá únicamente código ya que será el lugar donde tendremos tanto lo aportado por terceros (el proyecto previamente realizado), como nuestro trabajo. Por lo que en el mismo se podrá encontrar:

- Documentación
- Código fuente
- Scripts
- Bases de Datos

En cuanto a la estructura que llevaremos en el repositorio, quedará determinada de la siguiente manera:
Dividiremos en tres grandes carpetas y un *README.md*. Las carpetas serán las siguientes: *Código*, *Bases de datos* y *Documentación*. A su vez, subdividiremos a algunas de ellas en nuevas carpetas.
Documentación: en ella se encontrarán dos carpetas, una con la documentación brindada por los miembros que llevaron a cabo el proyecto y otra con la documentación propia de nuestros avances en el transcurso del obligatorio. Denominándolas: "*Documentación DA2*" y "*Documentación ISA2*", haciendo referencia a las asignaturas donde fueron elaboradas.
Código: se dividirá en dos carpetas, *Frontend* y *Backend*, basándonos en las ventajas que ello conlleva: desarrollo más rápido, protección de los servicios y evitar configuraciones.
Bases de datos: No se subdividirá, almacenará la base de datos brindada por quienes realizaron el proyecto, y posibles modificaciones.

Manejo de versiones

Para el manejo de versionado utilizaremos el modelo de integración *Trunk-based*. Este modelo consiste en tener una única rama principal desde la cual se crean feature branches caracterizadas por ser pequeñas y de corta duración.

Optamos por el mismo ya que facilita la integración continua, característica que permite acelerar el flujo de deployments y es importante en los procesos DevOps.

Además, utilizaremos la herramienta de Pull-requests para mergear código. Esto consiste en requerir que un segundo miembro del equipo (desarrollador) revise nuestro código antes de permitir que el mismo se integre a la rama trunk, en caso de ser aprobado, se mergea el código a trunk. Este proceso permite reducir la posibilidad de introducir errores, ya que el desarrollador encargado detectaría posibles bugs; permitiendo mitigar riesgos.

Por último, destacar que al momento de programar, utilizaremos TDD (pruebas unitarias), asegurándonos que previo a mergear todas se encuentren en un estado green, es decir, que hayan pasado.

Estándares aplicados

Para una mayor organización decidimos determinar algunos estándares a utilizar a partir de esta etapa en los cambios que realicemos, las agregaciones tanto de nueva documentación como código, entre otros.
Cabe destacar que en el momento de armar el repositorio no teníamos fijados los estándares, por lo que algunos, como los nombres de los commits, no cumplirán en su totalidad con ellos. Los propusimos después con el objetivo de tener una mayor organización y entendimiento del flujo de trabajo.

Branches

Como se mencionó anteriormente, trabajaremos con el modelo Trunk-based. Por lo que desde la rama principal saldrán ramas que deberán cumplir con algunos estándares en su nomenclatura:

- En caso de estar solucionando algún bug o error presente en el código se utilizará: *fix-NombreDeLaRama*
- Si es una nueva rama se nombrará como: *feature-NombreDeLaRama*

Destacar que en ambos casos se utiliza UpperCamelCase.

Issues

Para reportar los bugs y errores encontrados en las distintas funcionalidades del proyecto usaremos la sección de Issues que proporciona GitHub y la sección de Projects también de GitHub para plantear el tablero, colocar las issues y poder observar el flujo de trabajo. Utilizaremos algunos de los labels disponibles por defecto como: “bug”, pero también se crearán otros nuevos para poder indicar la severidad y la prioridad.

Según su severidad:

- Crítico: un defecto que obstaculice o bloquee completamente la prueba o el uso de un producto o función.
- Mayor: una función principal que no cumpla con los requisitos y se comporte de manera diferente a lo esperado. Cuando funciona muy lejos de las expectativas o no está haciendo lo que debería estar haciendo.

- Menor: función que no cumpla con sus requisitos y se comporte de manera diferente a lo esperado, pero su impacto es insignificante hasta cierto punto o no tiene un impacto importante en la aplicación.
- Leve: defecto cosmético.

Según su prioridad:

- Inmediata: plazo máximo 24 hs. (generalmente los defectos de severidad crítica se asocian a prioridad inmediata).
- Alta: plazo máximo 48 hs.
- Media: plazo máximo un par de semanas.
- Baja: sin plazo.

Commits

Para los commits definiremos un “label” que indique el tipo de commit que estamos realizando, y posteriormente su correspondiente descripción. Los tipos pueden variar, agregándole nuevas variedades a medida que el proyecto avance.
Por el momento definiremos los siguientes:

- Documentation: usado para agregar o modificar documentación del repositorio.
- Fix: se utilizará para indicar que se está solucionando un error en el código.

Documentación

En el caso particular de la documentación decidimos utilizar docs de Google Drive, pudiendo así llevar un control de versiones que nos permita ver las modificaciones que se han realizado, en qué momento y quién la realizó; incluso volver hacia atrás si se desea. Al momento de considerar que está finalizada, se convierte a formato PDF, y se sube en la carpeta correspondiente del repositorio; lugar donde estará todo lo relacionado al proyecto como ya se mencionó.
A modo de estándar definimos las siguientes características:

- Interlineado y espaciado entre párrafos: 1.5
- Letra: Arial 12, los títulos y subtítulos llevarán negrita.
- Cada documento tendrá una introducción e índice.
- Alineación: Justificado, a excepción de títulos que estarán centrados.