Adrián Torres Hernández A01173530
Javier Martínez Olivares A01401017
Luis Rodrigo Montúfar Pérez A01273078
Raziel Baruc Hernández Ramírez A01275036

Configuración del Entorno de Trabajo

Plataforma de comunicación: Utilizaremos Whatsapp como medio principal de comunicación entre nosotros en donde nos pondremos de acuerdo para realizar y entregar los laboratorios y zoom en caso de que la actividad demande una opinión del equipo.

Control de tareas: Con la ayuda de ***Trello*** tendremos control de tareas asignadas a cada uno de los integrantes.

https://trello.com/b/JPSPleYE/bloque-2

Control de versiones: El software de control de versiones que planeamos utilizar es GitHub.

Decidimos utilizar GitHub puesto que es una herramienta la cual nosotros ya estamos familiarizados por lo que sabemos como usarla y ya tenemos cuentas desarrolladas.

Adrián Torres Hernández A01173530
Javier Martínez Olivares A01401017
Luis Rodrigo Montúfar Pérez A01273078
Raziel Baruc Hernández Ramírez A01275036

Branching model: El modelo de trabajo que utilizaremos es “Git FLow” el cual está basado en dos ramas llamadas “*Develop* y *Master”*.

Según el artículo de 4 branching workflows for Git este flujo abarca las siguientes ramas.

Master: Contiene todo el contenido de producción.

Develop: Tiene código de preproducción y donde se pondrá el merge de los features.

Feature: Tienes las nuevas extensiones para el sistema.

Hotfix: Sirve como soporte para estatus no deseados en la rama de máster.

Release: Sirve para corrección de errores pequeños antes de un lanzamiento de una nueva producción.

Decidimos utilizar este modelo debido a la facilidad de comprender y sobre todo la limpieza en las ramas que se genera.

Adrián Torres Hernández A01173530
Javier Martínez Olivares A01401017
Luis Rodrigo Montúfar Pérez A01273078
Raziel Baruc Hernández Ramírez A01275036

El flujo general de Gitflow es el siguiente:

1. Se crea una rama `develop` a partir de `main`.
2. Se crea una rama `release` a partir de la `develop`.
3. Se crean ramas `feature` a partir de la `develop`.
4. Cuando se termina una rama `feature`, se fusiona en la rama `develop`.
5. Cuando la rama `release` está lista, se fusiona en las ramas `develop` y `main`.
6. Si se detecta un problema en `main`, se crea una rama `hotfix` a partir de `main`.
7. Una vez terminada la rama `hotfix`, esta se fusiona tanto en `develop` como en `main`.

Este es el link a nuestro repositorio:

https://github.com/JavierMtzO/Proyecto-CAMMI

Estándar de codificación: El código debe seguir las normas expuestas en el estándar PSR-1 que se encuentra en la página oficial de PHP-FIG, el cual es un estándar muy básico en PHP y fácil de utilizar el cual nos ayudará a darle un orden y un mejor entendimiento dentro de nuestros futuros trabajos.

Algunos ejemplos de estándares serían:

- Los archivos DEBEN usar solo etiquetas <?php y <?.
- Los nombres de clase DEBEN declararse en Studly Caps. Los nombres de clase, entre otras cosas, a veces se componen de varias palabras.
- Las constantes de clase DEBEN declararse en mayúsculas con separadores de subrayado.

Adrián Torres Hernández A01173530
Javier Martínez Olivares A01401017
Luis Rodrigo Montúfar Pérez A01273078
Raziel Baruc Hernández Ramírez A01275036

- Los nombres de los métodos DEBEN declararse en camelCase.

Referencias:

https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf

Flujo de trabajo de Gitflow | Atlassian Git Tutorial

PSR-1: Basic Coding Standard - PHP-FIG