Configuración del Entorno de Trabajo

Plataforma de comunicación:

Slack:

Ninguno de nosotros tiene mucha experiencia utilizando Slack, pero sabemos que en ambiente profesional se utiliza demasiado para tener una comunicación efectiva entre el equipo. Por lo tanto, lo utilizaremos para aprender a usarlo de forma efectiva y conocer sus herramientas disponibles.
https://lqsolutionsldaw.slack.com

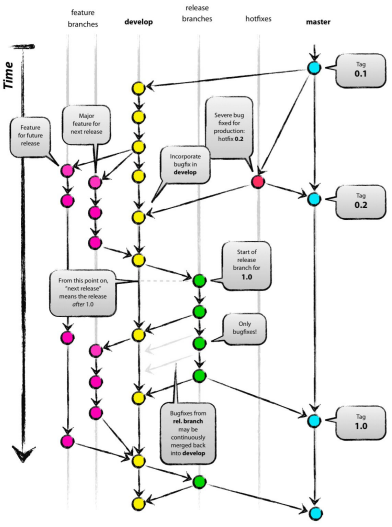
Software para control de versiones:

Github:

Decidimos utilizar esta plataforma debido a que todos los integrantes del equipo ya tienen experiencia en su uso. Además, Github ofrece repositorio privado para maximo 4 integrantes lo cual nos servira en este caso.
https://github.com/albertcastaned/LDAW

Branching Model:

El modelo de ramas que se utilizará es el modelo de ramas de Driessen. El repositorio central tiene 2 ramas principales: la rama master; que es la rama que contiene las versiones “listas para producción” y la rama develop; qué es la rama de integración donde se juntan los trabajos en progreso para la próxima versión de producción. Existen más ramas con un tiempo de vida mucho menor en donde se trabaja una sola función, un “hotfix” o liberación. La rama más común: la de función surgen de develop son usadas para desarrollar una nueva característica o función y mueren cuando ésta termina.



Referencia:

http://taylorlopes.com/um-modelo-bem-sucedido-de-branches-no-git/?lang=es

Justificación: Reducir o eliminar los conflictos que pueden surgir a partir de la unión de ramas y al trabajar simultáneamente. Es un modelo que nos permite a todos los integrantes acceder a la versión de trabajo más reciente y poder modificar o agregar cosas de manera rápida y efectiva, además si surgen errores o conflictos es fácil regresar a una versión estable sin perder mucho trabajo.

Herramienta para seguimiento de proyecto:

Trello: Utilizaremos Trello para administrar las tareas por hacer, las etapas en donde se encuentran, la fecha límite para terminarlas, y el encargado de esa tarea. Creemos que utilizar una hoja de Sheets o Excel es más lento que una aplicación como Trello.

- https://trello.com/invite/b/KyKjL1Vc/cb4d3c21362f449fa7bade6fe699f3ed/ldaw

Estándar de codificación:

Utilizando Python para el framework Flask, seguiremos el estilo de documentación y formato PEP8 para mejorar la legibilidad de nuestro código. Se instalará un parser de PEP8 en nuestros editores de texto para corregir automáticamente a este formato y así ahorrarnos tiempo. Sin embargo, aquí describimos los puntos principales que seguiremos para garantizar que todos sigamos el mismo formato:

- **Organización de archivos:**
  - Dividir archivos en carpetas dependiendo de los módulos
  - Separar el sistema y la documentación

- **Nombre de archivos:**
  - Simples y significativos.
  - En formato lowerCamelCase
  - Evitar abreviaciones

- **Nombre de variables:**
  - Evitar abreviaciones
  - Simples y significativos.
  - Separar palabras con piso: mi_variable
  - Variables constantes deberán estar en UPPERCASE

- **Espacios en operadores:**
  - Para mejorar la legibilidad, se utilizaran solo un espacio entre operadores como en los siguientes ejemplos:
    - **Correcto:**
      - x = y + x
      - total = (x + y) / (2 - 4)
      - valores = [“arg1”, “arg2”, “arg3”]
      - cadena = “ab” + “am” + “@” + “a” + “+” + var + “var”

    - **Incorrecto:**
      - x=y+x
      - total=(x+ y) / (2-4)
      - valores=[“arg1”, ” arg2,”arg3 “]
      - cadena =”ab” +”am” +”@” +’’a”+”+”+var+”var”

- **Funciones:**
    - Sus nombres son verbos en infinitivo
    - Solo deberan tener una tarea y deberán evitar llamar a otras funciones para evitar dependencias.

- **Comentarios:**
    - Solo escribir comentarios cuando sea difícil explicar lo que el código hace, o para escribir recordatorios.
        - **Recordatorios:**
            - #TODO: Descripción de error, implementación, o cualquier cosa que haga falta revisar después.
- **Espacios:**
    - Separar funciones y bloques de codigo relacionados con espacios.
        - Ejemplo:
            - class calculadora():

```
def sumar(self, numero_1, numero_2):
        print(numero_1 + numero_2)

def restar(self, numero_1, numero_2):
        print(numero_1 - numero_2)
```

- **Longitud de Líneas:**
    - Para evitar scroll horizontal, cada línea debe tener máximo 80 caracteres.

- **Principio DRY**
    - Don't Repeat Yourself, reutilizar código cuando sea posible

- **Casos de Prueba**
    - Diseñar y ejecutar pruebas automatizadas que logren demostrar la funcionalidad de la gran mayoría de las funciones del sistema. Las pruebas deberán ser diseñadas y ejecutadas lo más pronto posible para considerar el código entregado como terminado. Para lograr las pruebas automatizadas se utilizará la librería de python: unittest. Además, una vez que se dé inicio el desarrollo de código se instalarán hooks para correr las pruebas automáticamente antes de hacer un commit para poder así validar que lo que se realizó funciona correctamente.