Taller Git - MVC

# Objetivos:

* Analizar la estructura de un proyecto implementado a partir del patrón arquitectónico MVC.
* Realizar modificaciones al proyecto de forma distribuida utilizando GitHub.
* Resolver conflictos de integración de código utilizando diferentes estrategias.

# Antecedentes

Cada grupo debe crear una organización dentro de Github y crear un nuevo repositorio en donde todos los integrantes puedan tener permisos para realizar cambios (rw). La primera parte de este taller la deben realizar en conjunto, mientras que la segunda parte debe ser dividida entre los integrantes y deben subir sus avances al repositorio remoto.

## Pasos previos

1. El líder debe crear el repositorio en GitHub con un nombre cualquiera para este taller.
2. Agregar a los integrantes del grupo al repositorio. (Cada integrante debe abrir su correo y aceptar la invitación)
3. Dentro de la configuración del repositorio debe darles permiso “**Write**” a los miembros.
4. Después, debe clonar el repositorio en un directorio de la computadora local.
5. Descomprimir y copiar el código del proyecto Snake dentro del repositorio local.
6. Finalmente, debe agregar los nuevos archivos al repositorio remoto.

## Comandos importantes

* Conocer el estado de su repositorio: **git status**
* Clonar localmente un repositorio remoto: **git clone <repositorio\_remoto>**
* Agregar todos los cambios realizados al stage: **git add .**
* Guardar todos los cambios agregados al stage: **git commit -m “Comentarios de los cambios realizados”**
* Enviar al repositorio remoto todos los commit sin enviar: **git push origin master**
* Descargar los nuevos cambios desde el repositorio remoto: **git pull**
* Si hay un usuario grabado en su computadora y desea utilizar otro: **git config --local credential.helper ""**

# Parte 1

1. Abrir su correo asociado a Github y aceptar la invitación para colaborar en el proyecto.
2. Clone en un directorio de su computadora el repositorio remoto. (Esto deben realizarlo todos los integrantes en cada computadora)
3. Importe el proyecto al IDE de su preferencia (Se recomienda usar Eclipse, pero se puede importar desde Netbeans).
4. Observe y analice la estructura del proyecto, complete lo siguiente.

* Según su criterio, ¿cómo clasificaría a las clases/archivos que conforman el proyecto?:

|  |  |  |  |
| --- | --- | --- | --- |
| Modelos | Vistas | Controladores | Auxiliares |
| SnakeModel  GoldModel  GameModel | GUIView  GameView  GameFactory | GameController | Constants  Position  GameTile  RoundTile  RectangularTile  IGameFactory  GameOverException |

* ¿Considera usted que el proyecto fue implementado siguiendo los principios del patrón MVC?

En una vista general del proyecto, si cumple pero hay ciertas clases que tienen métodos que deberían estar en otra clase para cumplir los principios al 100%.

# Parte 2

Cada integrante debe clonar en su computadora el repositorio remoto y luego todos deben trabajar paralelamente con esta versión. Luego, el orden para enviar los cambios realizados debe ser el siguiente:

Nota1: Cada integrante debe resolver un conflicto de integración.

Nota2: Tengan en cuenta que cada integrante debe realizar los cambios que le corresponde, pero al finalizar deben subir en el orden (integrante 1, integrante 2, integrante 3, integrante 1). Esto es para tratar de generar un conflicto de integración de código con cada integrante.

1. **Integrante1**(creó el repositorio) Marlon Eddie Lindao Varas:
   1. Ventana principal:
      1. Cambiar el texto del botón de “Start Game” a “Let’s Go!!!”.
   2. Juego Gold:
      1. Por cada ficha recolectada, asigne 2 puntos en lugar de 1.
2. **Integrante2 Kevin Briones**:
   1. Ventana principal:
      1. Cambiar las dimensiones de la ventana a 15x15.
      2. Cambiar el texto del botón “Start Game” a “Let’s Play”
      3. Cambiar el color de fondo de la pantalla principal de blanco a gris (lightGray)
   2. Juego Snake:
      1. Para el juego ‘Snake’ cambie el color de la serpiente a verde (GREEN).
3. **Integrante3 David Chang**:
   1. Ventana principal:
      1. Cambiar el texto del botón “Start Game” a “Empezar”
      2. Cambiar el color de fondo de la pantalla principal de blanco a celeste.
   2. Juego Gold:
      1. Cambie el color de relleno de las fichas a verde (green) y el color del borde a azul (blue).
      2. Por cada ficha recolectada, asigne 3 puntos en lugar de 1.
4. **Integrante1( otra vez )**:
   1. Juego Gold:
      1. Cambie el color de relleno de las fichas a verde (CYAN).
      2. Por cada ficha recolectada, asigne 4 puntos en lugar de 2.
   2. Juego Snake:
      1. Aumente el número de frutas de 1 a 3.

# Parte 3

Responda a las siguientes interrogantes (**Solo uno del grupo debe subir este archivo al GitHub con las respuestas grupales**):

1. ¿Le resultó complicado realizar los cambios solicitados?

No, porque cada integrante teníamos respaldo de los cambios que habíamos hecho, lo cual no teníamos problemas si se nos eliminaba nuestros cambios.

1. ¿Cuáles considera usted que son los archivos/clases ‘más importantes?

Las clases gameview, snakemodel y goldmodel.

1. ¿Qué aspectos ayudaron a realizar los cambios?

Que cada integrante no trabajo mucho en una misma clase del otro integrante.

1. Luego de haber explorado el código, ¿considera usted que se respetan los principios de MVC?

No, porque en ciertas clases como goldmodel, que es un modelo, tiene funciones que deberían estar en una clase que sirva de control, por ejemplo, tiene el método que mueve al punto rojo.

1. ¿Qué cambios haría para mejorar la arquitectura de la aplicación?

Ubicar las funciones que no pertenecen a una clase, en la clase correspondiente.

# Entregables

1. En el repositorio de GitHub debe estar el código con los cambios requeridos.
2. En el repositorio de GitHub debe estar subido este archivo con las respuestas de todo el grupo.
3. En Sidweb debe subir el enlace del último commit del repositorio remoto.