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. Un integrante debe crear una organización en GitHub con un nombre cualquiera para este taller.
2. Agregar a los integrantes del grupo a la Organización enviándoles invitación. (Cada integrante debe abrir su correo y aceptar la invitación)
3. Dentro de la página de la Organización debe crear un nuevo repositorio.
4. Dentro de la configuración de la organización ir a “[**Member privileges**](https://github.com/organizations/TallerGIT/settings/member_privileges)” y en “**Organization members**” darles permiso “**Write**” a los miembros.
5. Después, debe clonar el repositorio en un directorio de la computadora local.
6. Descomprimir y copiar el código del proyecto Snake dentro del repositorio local.
7. 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 |
| GameModel  GoldModel  SnakeModel | GUIview  GameView | GameController  Main | Constants  GameFactory  GameOverException  GameTile  IGameFactory  Position  RectangularTile  RoundTile |

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

Si, ya que primero que todo existen las partes fundamentales y funcionales que son los modelos, los controladores y las vistas en donde en este caso los auxiliares son herramientas utilizadas por las vistas, por otro lado tanto la vista como los controladores dependen de los modelos y no en viceversa, donde en este caso se ve que los modelos Gold y Snake manejan en totalidad el comportamiento o lógica de los juegos, GameController maneja las acciones del usuario y las vistas correspondientes reflejan el cambio de los estados.

# 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:

1. **Integrante1**(creó el repositorio):
   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**:
   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**:
   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 (green).
      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.

Nota: 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.

# 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 ya que los métodos poseian su respectiva documentación y eso facilitaba mucho el trabajo de entender lo que hacía cada código.

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

**SnakeModel** ya que es la que contiene la mayoría de la lógica del juego, tales como crear las frutas, actualizar la posición de la serpiente, obtener la puntuación, entre otros.

**GameController** esta clase es la que permitirá correr y parar el juego, ademas contiene métodos que están relacionados con los eventos del teclado (escencial para el funcionamiento del juego).

**GoldModel**  Es importante ya que esta permite crear y cambiar las propiedades de las fichas y sobretodo porque contiene un método que es unos de los mas relevantes del juego(remover las fichas cuando la serpiente las toca y otorgar puntaje).

1. ¿Qué aspectos ayudaron a realizar los cambios?

Principalmente que los métodos contaran con su respectiva documentación.

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

Si, puesto que había una clara distinción en las clases, unas estaban totalmente enfocadas en crear la parte visual, otras en el comportamiento de los objetos y otras compuestas por métodos que interactuaban con la parte lógica y las vistas.

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

Se podría hacer una separación por paquetes para facilitar la identificación de las clases y comprensión de la lógica utilizada.

# 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.