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 |
| Constantes  GameFactory  GameTile  GoldModel  SnakeMoel  GameModel  Position  RectangleTile  RoundTile | GuiView  GameView | GameController | GameOverException  GameFactory  Main |

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

El proyecto no se encuentra en su totalidad implementado de forma que siga los principios del patrón MVC a un 100%, ya que en clases como SnakeModel, podemos ver que además de describir las características y atributos del objeto, también se definen el controlador de movimiento en la misma clase.

# Parte 3

**Responda a las siguientes interrogantes:**

1. ¿Le resultó complicado realizar los cambios solicitados?

Al principio un poco, ya que se me hacia un poco difícil interpretar la estructura, pero luego cuando la analice pude encontrar en qué lado del código estaba lo que debía de modificar.

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

Creemos que las clases más importantes son Game controller, Game model, Gold model, Snake model ya que son las que manejan la lógica del juego en general.

1. ¿Qué aspectos ayudaron a realizar los cambios?

La organización de cada opción del juego en clases diferentes ayudo significativamente a identificar que sección del código debía ser modificada para realizar los cambios requeridos.

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

Sí, los datos, interfaz y lógica está separados en clases distintas(no en todos los casos) lo que hace fácil la modificación de cada uno de estos.

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

Culminando la aplicación de los principios MVC, por ejemplo en SnakeModel no solo se define el modelo sino también el controlador de movimientos del juego.