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 |
| GoldModel  GameModel  SnakeModel | GameView  GUIView | GameController | Constants  GameFactory  GameOverException  GameTitle  IGameFactory  Main  Position  RectangularTile  RoundTile |

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

Yo consideraría que sí, ya que estaba organizado de tal manera que la comprensión y entendimiento del funcionamiento de cada uno de los módulos que constituyen este sistema se volvió mucho más sencilla de comprender.

# 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 (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.

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?
2. ¿Cuáles considera usted que son los archivos/clases ‘más importantes?
3. ¿Qué aspectos ayudaron a realizar los cambios?
4. Luego de haber explorado el código, ¿considera usted que se respetan los principios de MVC?
5. ¿Qué cambios haría para mejorar la arquitectura de la aplicación?

# 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.

Nombre: Javier de Jesús Anchundia Rosado

1. ¿Le resultó complicado realizar los cambios solicitados?

Si comparo esta situación con otras en las cuales he tenido que cambiar o modificar el código de otra persona, pues la verdad es que resulto mucho más sencillo de identificar en que parte se encontraba el código que debía modificar.

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

* Snake Model
* Gold Model
* GameController

1. ¿Qué aspectos ayudaron a realizar los cambios?

El hecho de que cada una de las clases esté bien etiquetadas, permitiendo facilitar el entendimiento de terceros

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

Yo consideraría que sí, ya que estaba organizado de tal manera que la comprensión y entendimiento del funcionamiento de cada uno de los módulos que constituyen este sistema se volvió mucho más sencilla de comprender.

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

La arquitectura de la aplicación me parece excelente y para ser sincero aun no tengo mucho conocimiento sobre la materia de Diseño de Software para atreverme a hacer algún cambio de la arquitectura ya podría caber la posibilidad de que en realidad empeore en vez de mejorarlo

DIEGO YANCE ALVAREZ

1. ¿Le resultó complicado realizar los cambios solicitados?

No me pareció tan complicado antes de haber hecho los *merges* debido a que la documentación del proyecto facilitaba las cosas, pero después del merge era un poco caótico debido a que se generaban líneas de código basura e inclusive en un método se repetía lo que ocasoniaba un poco de confusión

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

* Snake Model
* Gold Model
* GameController
* Main
* Las librerías jar

1. ¿Qué aspectos ayudaron a realizar los cambios?

Creo que ayudó el software *Eclipse* y lo bien comentado que se encontraba el documento, ya que en *Eclipse* venia etiquetado los métodos y variables que hay en las clases por lo que uno podría intuir donde estaba la ubicación para hacer los cambios y la documentación ayudaba a verificar si lo que se asumía era correcto.

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

Creo que sí debido a que el programa está estructurado de una manera que respeta los principios que haya una parte vista que era la clase *GUI* que se encargaba de mostrar los datos al usuario, la parte modelo era controlado por las clases modelo de cada parte del juego como *SnakeModel* o *GoldModel* y la parte controlador que se encargaba de actualizar la información como la posición de los objetos y la actualización del marcador.

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

Considero que para mejorar la aplicación sería ideal realizar una mejor implementación de cómo cambiar de un modo de juego a otro, ya que como esta implementado es muy tedioso para el usuario tener que abrir un menú desplegable y estar seleccionando en cada momento del uno al otro; considero que una mejor implementación sería tener dos botones donde al hacer clic en uno vaya al modo de juego que describa el texto del botón.

ALBIN ISMAEL ARIAS TIGSI

1. ¿Le resultó complicado realizar los cambios solicitados?

Un poco, debido a que se me complico encontrar donde se encontraba cada línea de código a modificar para realizar los cambios que se pide a cada integrante.

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

-GUIView

-Main

1. ¿Qué aspectos ayudaron a realizar los cambios?

Los nombres de los archivos/clases, además de los comentarios en cada línea de código permitiendo hallar de manera fácil cada línea que se tenía que modificar

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

Sí, porque cada parte estaba dividido y agrupado donde correspondía separando los así de manera correcta los datos y la lógica de la aplicación de su representación (Interface)

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

Le pondría más diversión, tal vez cambiaria los círculos por otras figuras a medidas que pasan los niveles, para pasar los niveles que consiga cierta cantidad de puntos, haría más llamativo y más interactivo.