Proyecto DYDS

¿Cómo dividí el proyecto?

El proyecto estaba todo contenido en una sola clase enorme que no cumplía con ninguna practica de “Buena programación”, lo primero que hice fue entender que cosas hacia y como las hacia la clase dada y dividí las funcionalidades en 3 carpetas respetando MVC

Carpetas del Proyecto

Controller

En el controlador hay 3 clases entre ellas la clase main que se encarga de crear los objetos necesarios para ejecutar el programa, además se encuentra la interface “GameControler” y su respectiva implementación “GameControleImpl”.

La clase controlador va a ser intermediario entre la vista y el modelo, creando los hilos y también es el encargado de manejar las excepciones y de avisar en la vista al usuario cuando se genera algún error

Model

La carpeta model es la mas cargada de todas contiene 5 clases y 4, las cuales se encargan de distintas cosas, aplique SOLID en este punto y me cerciore que las clases tengan la menor cantidad de responsabilidades posibles, dichas clases son:

DataBase: Es la encargada de manejar la base de datos, dicha clase va a guardar y rescatar los datos de la Base de Datos siempre que sea necesario.

GsonLogic: Es la encarga de gestionar los datos que se extraen de Wikipedia y también de manejar las API dadas por la catedra (**WikipediaPageAPI** y **WikipediaSearchAPI**) para la conexión con Wikipedia.

Game: Es una clase que cree para encapsular los datos de un juego en específico, solo tiene campos para guardar el nombre y el texto a mostrar mas sus setters y getters

GameModel y GameModelImpl: GameModel es la interface la cual implementa GameModelImpl, Esta clase es a la que llamar el controlador para avisarle que tiene que hacer una tarea y es la clase donde la vista va a buscar los resultados de la operación que el controlador le pidió, además gestiona las ordenes que se le dan a la base de datos y a la API de Wikipedia

View

La carpeta view es la que contiene los forms los cuales se van a mostrar al usuario, en este punto también aplique SOLID dividiendo lo que antes estaba en un solo formulario en 4 distintos que solo interactúan con el controlador para ejecutar las acciones y con el modelo para recibir el resultado de las mismas, además que el controlador trabaja con las interfaces y no con las clases

GameSearchView y GameSearchViewImpl: GameSearchView es la interface la cual implementa GameSearchViewImpl, esta clase es la encargada de iniciar las búsquedas y de mostrarle el resultado al usuario, como tal no calcula nada toda la lógica la hacen entre el controlador y el modelo

GameStoredView y GameStoredViewImpl: GameStoredView es la interface la cual implementa GameStoredViewImpl, esta clase es la encargada de mostrar los datos que están guardados en la base de datos, de la misma manera esta clase no se encarga de la lógica, la lógica es delegada al controlador y al modelo

GameHistoryView y GameHistoryViewImpl: GameHistoryView es la interface la cual implementa GameHistoryViewImpl, esta clase es la encargada de mostrar el historial por pantalla, también toda la lógica recae en el controlador y el modelo

GameSearchMenu: Esta clase solo se encarga de mostrar la barra de navegación en la aplicación, además de manejar los errores que se van a mostrar por pantalla

Test

El test esta organizado en 2 carpetas integrationTest y UnitTest, en dichas carpetas están las clases necesarias para hacer los testeos al programa, utilice clases dummies para facilitar el trabajo

Aclaraciones / decisiones de diseño

- En el historial decidí que si buscas la misma cosa mas de una vez el mismo dia, se guardara una vez en el historial, por ejemplo: si buscamos 5 veces god of war mientras el programa este en ejecución se mostrara las 5 veces que lo busco pero cuando lo cierre y lo vuelta a abrir solo aparecerá una.
- La clase GameSearchMenu decidí que no tenga una interface ya que si uso es muy básico y solo sirve para mostrar y navegar en la vista
- Los métodos usados solo para el testing decidí ponerlos en la clase y no en la interface, ya que son solo métodos que se usan para testear y nunca van a ser utilizados para el funcionamiento del programa.

Cosas a mejorar

- Principalmente lo que más me gustaría mejorar es la parte de testing, no quedo como a mí me gustaría me faltaron varias cosas por testear en el testing de integración

Diagrama de clases

El diagrama de clases esta guardado en la carpeta donde envió el proyecto, ya que al ponerlo en el documento no se veía de una manera correcta