Universidad de Barcelona

Arthur Font / 20222613

Cristian Rodríguez / 20222381

Victor Llinares / 20222160

Proyecto de Prácticas

Disseny de Software

Práctica 2 – 03/11/2019

Barcelona

2019

Index

1. Introducción

El objetivo de la práctica es diseñar la aplicación utilizando el método TDD (Test Driven Development) creando la estructura específica para que cada test de aceptación se ejecute de manera independiente y que facilite la estructura del código, aplicando los padrones de diseño.

La técnica de desarrollo ágil es una metodología muy común en los proyectos de software, ya que se basa en el desarrollo de manera iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.

2. Modelo de Classes

(definición de todas las clases)

…..

…..

Diagrama de clases

ub.edu.controller.Controller o-- ub.edu.model.CarteraClients : carteraClients

ub.edu.controller.Controller o-- ub.edu.model.CatalegSeries : catalegSeries

ub.edu.controller.Controller o-- ub.edu.resources.service.DataService : dataService

ub.edu.controller.Controller o-- ub.edu.resources.service.AbstractFactoryData : factory

ub.edu.model.Actor -up-|> ub.edu.model.Artist

ub.edu.model.Director -up-|> ub.edu.model.Artist

ub.edu.model.Episodi o-- ub.edu.model.DataEpisodi : data

ub.edu.model.Episodi o-- ub.edu.model.DurationEpisodi : duration

ub.edu.model.EstatEnReproduccio -up-|> ub.edu.model.Estat

ub.edu.model.EstatEnReproduccio o-- ub.edu.model.DurationEpisodi : tempsVist

ub.edu.model.EstatVist -up-|> ub.edu.model.Estat

ub.edu.model.Serie o-- ub.edu.model.Productora : productora

ub.edu.resources.dao.MOCK.DAOArtistMOCK .up.|> ub.edu.resources.dao.DAOArtist

ub.edu.resources.dao.MOCK.DAOClientMOCK .up.|> ub.edu.resources.dao.DAOClient

ub.edu.resources.dao.MOCK.DAOEpisodiMOCK .up.|> ub.edu.resources.dao.DAOEpisodi

ub.edu.resources.dao.MOCK.DAOSerieMOCK .up.|> ub.edu.resources.dao.DAOSerie

ub.edu.resources.dao.MOCK.DAOTemporadaMOCK .up.|> ub.edu.resources.dao.DAOTemporada

ub.edu.resources.dao.MOCK.DAOUsuariMOCK .up.|> ub.edu.resources.dao.DAOUsuari

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOArtist : artistDAO

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOClient : clientDAO

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOEpisodi : episodiDAO

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOSerie : serieDAO

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOTemporada : temporadaDAO

ub.edu.resources.service.DataService o-- ub.edu.resources.dao.DAOUsuari : usariDAO

ub.edu.resources.service.FactoryMOCK .up.|> ub.edu.resources.service.AbstractFactoryData

4. Repartición del trabajo

Hemos discutido las ideas y el diseño más general en grupo. El resto del trabajo ha sido delegado de la siguiente forma:

Victor: Llistar catàleg de series, Detalls d'una sèrie

Cristian:Visualitzar Whatched, ContinueWatching y MyList

Arthur:Marcar la serie en MyList

El resto del trabajo no indicado ha sido de forma grupal donde todos aportamos ideas y las discutimos para hacerlo de una u otra forma.

5. Cambios

La clase valoración la hemos cambiado ahora es una clase que implementa un hashmap <usuario, entero>,suponiendo que la valoración va de 1 a 5, que guardará para cada usuario la valoración que ha guardado. Siendo valoración un atributo de Episodio, para acceder al usuario, ya que puede haber usuarios iguales, hemos hecho que el primer parámetro sea una string con el nombre del cliente más el nombre del usuario. Al no haber 2 clientes con el mismo nombre no habrá el error de 2 usuarios iguales en el hashmap.

Eliminamos la clase enlace que hacía de unión entre Episodio, Usuario y Visualización de Episodio.

Hemos quitado la clase de estado No_Visto, ya que hemos llegado a la conclusión de que al tener esa clase, cada vez que añadiéramos un usuario tendríamos que recorrer todos los episodios de todas las temporadas de todas las series y marcarlos como no vistos, y hemos dejado únicamente las clases Visto y EnReproducción que derivan de Estado,

6. Conclusiones

Realizar este trabajo nos da la idea de como diseñar un problema, en este caso, la plataforma para visualizar series, para tenerlo todo más estructurado.

Y así poder dividir el trabajo en problemas menores, tener menos fallos de compatibilidad entre las partes de código realizado por las diferentes partes del grupo y en general conseguir una plataforma más sólida, clara y eficaz.