Universidad Autónoma de Querétaro

Facultad de informática

Patrones de diseño

Proyecto final – Taller

Enrique Aguilar

315342

Jeffrey Castillo Pérez

Ingeniería en Software

Grupo 34

Descripción

El proyecto final consiste en el entendimiento y asociamento de los conceptos vistos en clase (Principios de POO, Principios SOLID y algunos patrones de diseño) en una serie de archivos que se nos fue asignado

Carpetas

- ***Common***
  - ***Auth***

    Auth-service
    Authservice

    Dentro de la carpeta de **Auth** se encuentran dos archivos, los cuales podrían asociarse con el principio de Single Responsibility ya que un tienen métodos que tienen solo una tarea encargada, ya sea encriptar la contraseña con hash y la verificación del inicio de sesión.

  - ***Cache***

    Cacheservice
    Index
    Types
    Until

    Dentro de los otros archivos, se encuentran interfaces y exportaciones de los archivos de la carpeta **Auth**, pero dentro del archivo Until se encuentra algo interesante, que es el CamelCase, el cual es tener una palabra con mayúsculas intermedio, lo cual se asemeja a la figura de un camello.
    El mismo archivo puede ser considerado que utiliza el patron de diseño ADAPTER, ya que proporciona funciones para recibir parámetros de todo tipo (versiones, cadenas de texto, hashes y URLs)

- ***Config***
  - ***Db***

    Index
    Redis
    Sql

    Dentro de los archivos, contiene funciones para la conexión con base de datos y comprobar la misma con REDIS y POSTGRE

- ***Inversify***
  - *Index*

    Dentro del archivo, se utiliza el patrón de diseño SINGLETON, ya que cuenta con una única instancia reutilizable para la aplicación

- ***Constants***

  Code-errors
  Default
  Game
  Index
  Redis

  Dentro de esta carpeta solo se declaran funciones y variables que son útiles para la prevención de errores, variables de tiempo y la definición de mensaje de victoria o derrota

- ***Controllers***
  - ***User***

    Sign-in
    Sign-up

    Dentro de la carpeta, los dos archivos que contiene son similares entre sí, ya que son para el inicio y registro de sesión por lo que podría estar usando el patrón de diseño, PROXY, ya que es útil para controlar el acceso a la aplicación y retrasar la acción para verificar o subir la información

  Base-controller
  Index

  Dentro del archivo de **BASE-CONTROLLER** se utiliza un principio de POO ABSTRACCION, ya que la clase BaseController es una clase abstracta, lo que significa que no puede ser utilizada instanciando directamente y debe ser extendida por otras clases.

- ***Data-access***
  - ***Cache***

    Redis-impl

- ***User***
  - *User-repository*

    Dentro del archivo **REDIS-IMPL** se puede apreciar otros dos principio de POO que es la herencia y el encapsulamiento, la herencia se aprecia con la implementación de la interfaz CacheService, mientras que el encapsulamiento con la creación de métodos get y set para obtener o cambiar varios parámetros pero centrados en el de cliente. Este ultimo de igual manera se implementa en el archivo **USER-REPOSITORY**

- ***Domain***
  - ***User***
    - *Sign-in*
    - *Sign-ip*
    - *User-repository*
    - *User*
  - *Index*
  - *UseCase*

    Dentro de los archivos de **SIGN-IN** y **SIGN-UP,** se puede apreciar el uso de la herencia al usar la implementación de la interface SigninUseCase

- ***Interfaces***
  - *Custom-error*
  - *Index*

    Dentro de la carpeta, los archivos exportan interfaces, así como otros archivos

- ***Plataform***
  - ***Config***
    - *Index*
    - *Types*

      Dentro de los archivos se crean variables que sirven para la configuración final para la aplicación, asi como la base de datos, los puertos, la comprobación del inicio de sesión.

- ***Lib***

  General-error

  Dentro del archivo, se puede volver a apreciar la herencia ya que la clase CustomError extiende la clase base Error, heredando su propiedades y métodos.

- ***Middlewares***

  Error-handler
  Validate-token

  Dentro del primer archivo se definen constantes útiles para el manejo de errores dentro de la aplicación. Dentro del segundo archivo se valida y autoriza el acceso

- ***Server***

  Express
  Index
  Types

  Dentro de los archivos nuevamente se aprecia el principio de la herencia, entendiendo la interface CustomRequest de la interface base Request
  De igual manera se utiliza un poco de lo que es el ADAPTER, ya que se utilizan un método para transformar un parámetro de entrada y así poder usarlo

- ***Routes***

  index
  user

  Dentro de la carpeta, se puede apreciar nuevamente el principio de encapsulamiento, ya que se utiliza un método get para acceder a la clase container

- ***Until***

  Index
  Promises

  Dentro de la carpeta contiene archivos con el uso de funciones y promises, las cuales son un mecanismo para manejar operaciones asincrónicas, ya que pueden representar un valor que puede estar disponible ahora, en el futuro o nunca.

A lo largo de los archivos he podido distinguir algunos de los conceptos vistos en clase, pero otros no solo se encuentran en un archivo, asi como lo es el principio de Interface Segregation, ya que muchas de las interfaces base que se creaban, mayormente eran implementadas en otras interfaces con mas información

Conclusión

A lo largo del taller he podido aprender cosas útiles para la programación de las cuales ni siquiera era consiente, además de reforzar algunos de los temas que vi en algún momento en mis clases, pero que sé que podre aprovechar e intentar implementar en mis siguientes trabajos y proyectos a futuro