DPOO Taller 5

Santiago Casasbuenas 202214932

El proyecto sobre el cual se realizará el análisis consiste en el desarrollo de una aplicación de correo electrónico que le permite al usuario agregar un correo electrónico existente o ingresar como usuario ya registrado. Adicionalmente, la aplicación cuenta con funcionalidades que les permiten a los usuarios mandar correos electrónicos a otros correos que se encuentren registrados dentro de la aplicación.

La estructura general del diseño se centra en un patrón de desarrollo de tipo modelo vista controlador. En este, la vista es la encargada de conformar completamente el front-end de la aplicación, mientras que el modelo y el controlador son los encargados del back-end.

Los retos de diseño que presenta el proyecto son principalmente la forma en la que se controlan las instancias de la aplicación. Esto debido a que, al registrar un usuario, a menos de que este borre su cuenta, el perfil de este no debería desaparecer una vez el usuario termine de utilizar la aplicación. Es por eso por lo que en este proyecto se utiliza el patrón “Singleton”, el cual les permite a los desarrolladores construir clases excepcionalmente grandes las cuales solo necesitan ser utilizadas una vez y luego esa misma instancia pueda ser reutilizada la cantidad de veces que sea necesaria.

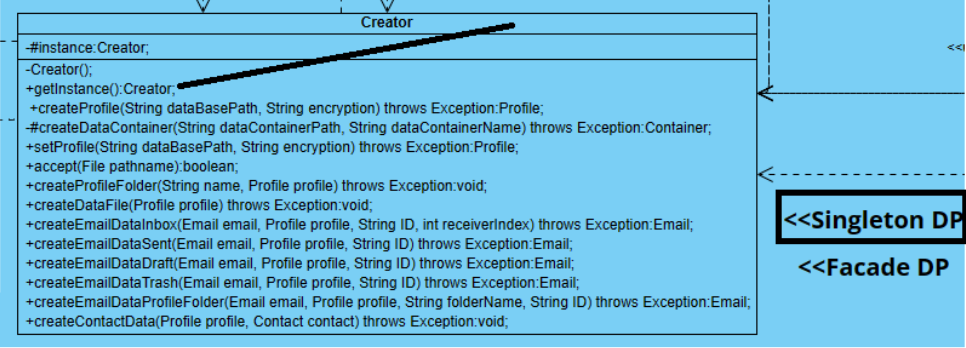
* Enlace a UML del proyecto:

<https://drive.google.com/file/d/1lc6Dt8ateEJRGqP8FJWiq-h6v7QSehwY/view>

* Enlace al repositorio en GitHub:

<https://github.com/mohamedhassan279/Email-Server.git>

Siendo que este es un proyecto grande el cual consta de una gran cantidad de clases, se ha dejado un enlace al UML del proyecto por si es necesario revisar la estructura completa de este. Para el desarrollo de este taller, solo nos interesan tres clases principales que manejan el patrón Singleton.



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Como podemos observar en las imágenes previas, tanto en la clase “Creator” como en la clase “Delete” existe un atributo público llamado instancia. Más adelante también se observa que ambas clases utilizan un método “get” para obtener el estado de la instancia, lo cual le permite a la base de datos tomar una decisión sobre que hacer con el perfil del usuario en cuestión. Principalmente, el patrón se usa sobre estas dos clases debido a que, son aquellas que condensan las acciones que un usuario puede ejecutar dentro de la aplicación y que afectan el perfil de este. Es decir, que, si un usuario quiere eliminar a un contacto de su lista, este debe poder hacerlo en la misma instancia en la que se eliminaría o agregaría cualquier otro contacto o cualquier otro atributo que los desarrolladores le hayan permitido al usuario. Adicionalmente, se tiene una clase llamada “Database” la cual se conecta directamente con el Paquete del Modelo y guarda la información del perfil del usuario basado en la instancia.

El patrón Singleton se utiliza para garantizar que una clase tenga una única instancia y proporciona lo que denominamos un “punto global de acceso” a esa instancia.

Una de las principales ventajas que tiene usar el patrón Singleton es que, como mencionado anteriormente, les permite a los desarrolladores hacer uso de clases que pueden ser más grandes y complejas de lo normal utilizando solo una instancia de estas, lo cual además cuenta con la ventaja de que esta instancia es reutilizable las veces que sea necesario, por lo cual, al cerrar la aplicación y hacer uso de esta en una nueva instancia, no se comprometen datos que ya fueron guardados anteriormente.

Una desventaja que podemos observar en el uso del patrón Singleton es que puede generar dificultades entre las subclases. Esto debido a que la instancia única se comparte entre la clase base y sus subclases, y el hecho de modificar el comportamiento de la instancia única en una subclase puede afectar a otras subclases y a la clase principal.

Teniendo en cuenta que el patrón Singleton genera una dependencia de instancias únicas que se acceden de manera global, otro enfoque mediante el cual se puede resolver el problema en cuestión sería la Inyección de Dependencias, ya que a través de esta se pasan las dependencias necesarias por medio de la construcción o métodos de una clase especifica. Todo esto con el fin de generar dependencias explícitas.