Proyecto 2: proyecto

Equipo: equipo

Integrantes:

- Arcos Morales Ramón. No: 319541478
- Casarrubias Casarrubias Victor Manuel. No: 421003581
- López Asano Miguel Akira. No: 320219089

Problemática:

Un estudiante cuenta con una computadora que ya no utiliza, casualmente, dicha computadora cuenta con gran almacenamiento y jdk. Por lo que planea utilizarla como servidor para guardar sus archivos. Quiere que desde su computadora personal, sea posible conectarse al servidor.
"Quiero una interfaz gráfica en la cual pueda subir mis archivos de texto al servidor, ver y editar estos mismos. Además, tener una bitácora con los cambios hechos a dichos archivos." El servidor está pensado para estar siempre prendido. Por último, el estudiante es políglota pero hay veces que solo puede pensar en un lenguaje en específico. Es por eso que requiere la opción de cambiar a alguno de los lenguajes que conoce.

Información del proyecto:

Para el correcto funcionamiento de nuestra aplicación, es necesario que el servidor sea ejecutado en una maquina basada en **Unix**. Este programa fue testeado con:

- Una computadora corriendo ambos, el cliente y el servidor.
- Dos computadoras en la misma red local sin ningún tipo de firewall o restricción. En donde una maquina ejecutaba el servidor y la otra el cliente.

Se hace uso de la herramienta **Maven** pues instala la paquetería necesaria para utilizar *Javafx*. Además, es requisito,tener organizado los archivos en distintos directorios para su correcto funcionamiento.
Para compilar el proyecto con jdk 20 y Maven 3.8.6:

- **mvn compile**
- **mvn install**

Para ejecutar el servidor:

- **./bin/server**
- Utilizar un puerto entre **1024** y **65535**.

Para ejecutar el cliente:

- **./bin/client**

En caso de probar el programa en una sola maquina:

- Address: **localhost**.
- Port: (Puerto seleccionado al ejecutar el servidor)

Clases correspondiente a Patrones:

- **MVC:**
  1. **Vista:** language.fxml, primary.fxml, textEditor.fxml, connect.fxml, list.fxml.
  2. **Controlador:** Main, App, PrimaryController, Client, ConnectController, ConnectWindow, LanguageController, LanguageWindow, TextEditorController, TextEditorWindow, ListController, ListWindow, Language, Spanish, English, Japanese.
  3. **Modelo:** MainServer, LogBuilderInterface, Log, EventLogBuilder, ServerLogBuilder, ChangeLogBuilder, DirectoryInterface, ProxyDirectroy, DirectoryInterface, Directory, Iterable, Protocol, HostServer, FileDirectory, ModifiedFile, VersionFile, Serializable, CoreFile.
- **Decorator:** Serializable, CoreFile, VersionFile, FileDirectory, ModifiedFile.
- **Iterator:** Directory, Iterable.
- **Proxy:** DirectoryInterface, Directory, ProxyDirectory.
- **Builder:** Log, LogBuilderInterface, EventLogBuilder, ChangeLogBuilder, ServerLogBuilder.
- **Strategy:** Language, Spanish, Japanese, English.

Justificación

- La aplicación es regida por el patrón **MVC**, ya que además de ser imperativo su uso, se quiere tener completa independencia entre el funcionamiento de nuestro modelo, que en este caso es un Servidor que maneja información, y la Vista, la cuál será la forma en la que el usuario podrá interactuar con el Sistema. Este patrón nos ayuda a prevenir múltiples problemas y cerciora el correcto funcionamiento de nuestro servidor y cliente. Pues no siempre estarán en un mismo sistema.

- Se decidió usar el patrón de diseño **Decorator** en nuestros archivos. Esto con la finalidad de poder preservar un historial de los cambios que tiene cada documento sin tener que sobreescribir las versiones anteriores.
- Dentro del Directory, se hace uso del patrón **Iterator**. Pues nos permite recorrer nuestra estructura de datos que contiene los archivos con sus respectivos nombres. Esto facilita el proceso, además de hacer el código más legible.
- Debido a que se quiere tener un sistema más seguro, se hizo la adición del patrón de diseño **Proxy**. Pues además de ser necesario que el cliente no modifique directamente al Directory dentro de HostServer. Es necesario un representante que maneje todas las peticiones hechas por el cliente.
- En la creación de los Logs, que guardan información relacionada a los cambios realizados por el cliente, se hizo uso del patrón **Builder**, esto, con la finalidad de crear un código más legible y menos engorroso. Pues nuestro Log puede recibir distinta información gracias a sus múltiples constructores.
- Dentro de nuestro controlador, para la opción de cambiar el lenguaje deseado de la vista, se hizo el uso del patrón de diseño **Strategy**. Esto, con la finalidad de permitir el cambio de comportamiento en tiempo de ejecución, que en este caso es el cambio de los mensajes mostrados en pantalla.