Programación III

Trabajo Final - 2ª parte

Grupo de trabajo 11

integrantes:
Joaquin Tomas Esposito
Criado Facundo
iavicoli nicolas
Eugenia Toledo

Año de realización: 2024

Índice:

1.Introducción:

El trabajo siguiente tendrá por objetivo lograr construir un programa completo que pueda ser ejecutado por un cliente vía interfaz gráfica y elaborado con prácticas de la programación orientada a objetos, concurrencia de hilos de ejecución, en la que se ejecutarán distintos procesos de forma sincrónica, separación de las capas de trabajo donde la interfaz gráfica, la lógica de ejecución, y la comunicación entre estas últimas exista de forma independiente. El programa permitirá a un usuario registrarse, iniciar sesion y simular su paso por un proceso de contratación de choferes para realizar un viaje y calcular el precio en base a la ruta, el equipaje y el acompañamiento de mascotas en caso de que estos lo permitan, y que estos procesos puedan persistir en el disco y no rehacerse cada vez que se ejecute el programa.

El motivo está en poder construir un programa más cercano a una experiencia real de producto así como el aprendizaje de la concurrencia de procesos, patrones como dao/dto que permiten persistir datos de forma independiente a la programación de los mismos, observer-observable que organiza los cambios en base al cambio de estado de los datos en ejecución, y el MVC que organiza y gestiona la inclusión de una interfaz gráfica en un programa.

2.Desarrollo:

2.1.Metodología:

Los pasos clave que se planearon durante la implementación del sistema:

Definición de Requisitos:
Elaboración de clases y jerarquías en base a las especificaciones provistas por la cátedra y la retroalimentación en las consultas, con el objetivo de organizar la codificación posterior de las mismas.

Implementación de las Funcionalidades:
Se aplicó una refactorización a la primera versión del trabajo final con el objetivo de que las capas de la arquitectura de 3 capas se conviertan en la capa de modelo del patrón de arquitectura MVC, y posteriormente, desarrollar las capas de controlador y vista al usuario.

Pruebas y Depuración:
Fase previa a la entrega del trabajo, donde se realizan las pruebas de los componentes del sistema y se corrigen los fallos, de ser necesario volviendo a rediseñar dichos componentes

Rediseño:
De ser necesario, replantear enteramente la función de ciertos componentes en caso de haber encontrado demasiadas complicaciones en la fase de depuración, aunque siempre dentro de pautas de los patrones de diseño que imperan sobre la estructura del sistema.

2.2.Diseño e implementación:

(Diagrama UML del sistema completo)

2.2.1.Diseño MVC

Modelo: Sección lógica del sistema, que toma la entrada de datos, los procesa, almacena, y luego ejecuta la salida de datos que serán comunicados por la vista.

Vista: Sección que contiene la entrada y validación de datos por parte del usuario, también ejecuta la presentación del programa, que en este caso se da por una interfaz gráfica, compuesta por ventanas, campos de texto, botones, entre otras cosas.

Controlador: Capa del sistema que comunica los datos ingresados por la vista hacia el modelo y los cambios del modelo para actualizar los componentes de la vista.

2.2.2.Patrones utilizados

Patrón DTO:
En este caso empleado para transferir y posteriormente persistir a los choferes, vehículos, y usuarios que emplean el sistema.

Patrón Observer-Observable:

Patrón MVC:

2.2.3.Diseño de clases

3.Conclusiones:

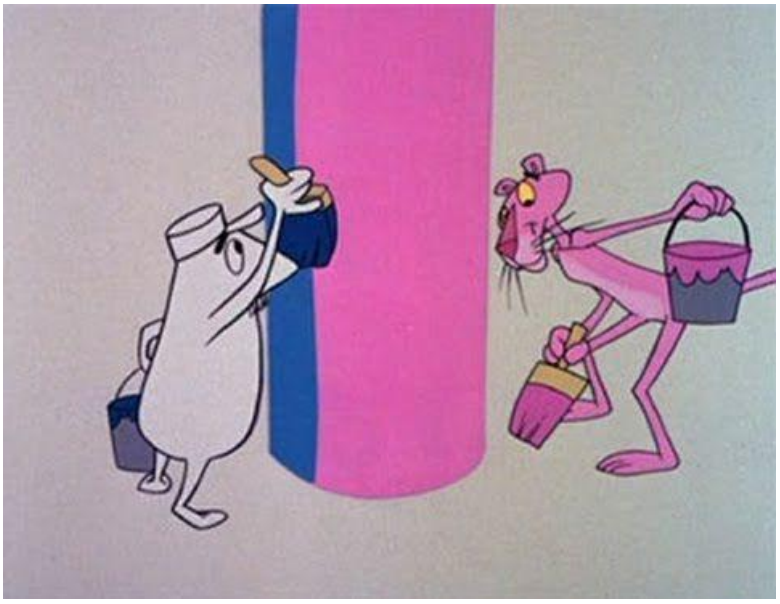
3.1.Expectativas

Al iniciar este proyecto, se tenían diversas expectativas y motivaciones que abarcaban varios aspectos, en el Académico:, aplicar los conocimientos teóricos adquiridos en el aula en un contexto práctico, profundizar en conceptos como patrones de diseño, arquitectura de software y buenas prácticas de programación. En lo colaborativo aprender a colaborar

eficientemente con otros miembros del equipo via herramientas de control de versiones. En desarrollo técnico la meta era mejorar las habilidades de programación en Java y comprender mejor los patrones de diseño utilizados en el proyecto.
Aspiraba a enfrentar desafíos técnicos y aprender de ellos.
En retrospectiva, puedo afirmar que muchas de estas expectativas se cumplieron, aunque encontramos problemas en el camino.

3.2.Dificultades

La dificultad más clara es la resolución de errores que requieran inquirir en las partes del código a cargo de otro miembro del equipo, debiendo adaptarse rápidamente a codigo ajeno, en el transcurso de un proyecto en continuo cambio los diagramas de clases pueden no permanecer de acuerdo a lo pactado en un inicio y dar lugar a situación donde el trabajo de uno pisa el del otro.



(Una sencilla ilustración de lo que puede convertirse la experiencia si hay comunicación deficiente entre 2 o más miembros del grupo.)

Otro gran problema fueron los hilos de ejecución, a primera vista sencillos pero que resultaron complicados de ejecutar y representar en la vista, tras estudiar cuidadosamente el patrón observer-observable se llegó a un diagrama de clases que era más claro de implementar en el sistema.

3.3.Soluciones

Desarrollar componentes como los eventos y los observadores casi que por separado del modelo para después introducirlos al mismo una vez está más clara la comunicación con las otras capas de la arquitectura resultó ser una metodología más apropiada para producir mejores resultados a la hora de ejecutar el programa.

3.4.Aprendizaje

Lo más importante parece ser como la arquitectura termina comportándose como un cliente-servidor y las capas y jerarquías de clases sólo son formas de organizar este comportamiento, a la hora de dividir el trabajo resulta más productivo aproximar la carga de trabajo por ese lado y no caer en querer hacer capas de forma individual, puesto que puede derivarse en los problemas de comunicación mencionados anteriormente. Además la mayor carga de trabajo aparece en la fase de depuración y una línea de trabajo lineal puede tornarse exponencial hacia el final del desarrollo del sistema, por lo cual es recomendable llegar lo antes posible a esta instancia de depuración.