Daniel Humberto Velez Rojas - 202221703
Eric Sebastian Alarcón - 202220287
Andres Botero Ruiz - 202223503
Manolo Hernández Rojas - 202224469

DOCUMENTO FINAL

En el presente documento se encuentra un resumen de los tres proyectos con sus respectivas descripciones y los diagramas de clase correspondientes en su última iteración. Adicionalmente, se hará el respectivo análisis de los factores positivos y negativos que consideramos para cada entrega final en los primeros dos proyectos. Con respecto al análisis anterior, se harán conclusiones sobre posibles mejoras y cambios.

Proyecto 1:

Al inicio del proyecto, cuando apenas estábamos teniendo un abrebocas del curso y empezabamos a conocer los diagramas de clases, no sabiamos ni que tan complejo ni dificil sería hacer uno para esta entrega tomando en cuenta la información suministrada e imaginandonos posibles futuras implementaciones para los siguientes trabajos. Al momento de iniciar decidimos hacer una primera lectura del pdf individualmente para crear nuestro propio entendimiento del trabajo. Luego, teniendo una perspectiva personal, hicimos una lectura grupal donde subrayamos, con distintos colores, 4 diferentes aspectos que encontramos en la guía: objetivos y atributos, requerimientos funcionales, restricciones y aspectos que no supiéramos cómo clasificar. Lo anterior nos ayudó a organizar nuestro primer UML que a pesar de tener varios cambios hasta el resultado final, gracias al proceso previo de la lectura del trabajo, fue un inicio clave para llegar a resultados -que como grupo- creemos valiosos.

A partir del primer diagrama, empezamos a añadir nuevas clases, atributos y requerimientos que se nos ocurrían durante reuniones grupales e incluso fuera de estas. Mientras añadimos y editamos, había llegado el momento de programar. Con respecto al diagrama sentimos que en este punto habíamos logrado un 80% del resultado final, sin embargo al terminar de programar, tuvimos que editar e incluso eliminar una clase hasta lograr el UML enviado para este proyecto. (imagen 1.0).

Dificultades:

Ahora bien, a la hora de programar tuvimos ciertas dificultades. En términos de la tecnología fue dificil entender como manejar la persistencia, es decir, guardar la información para que cuando se inicie la aplicación, no sea necesario cargar siempre datos para formalizar el alquiler o manejar requerimientos a los que les haga falta información previa. Adicionalmente, el tema de los usuarios (login), fue complicado ya que instanciar el objeto usuario dependiendo de las credenciales y permisos que cada usuario tenga se hizo un poco extenso tanto planear como creíamos que sería óptimo hacerlo e implementar esas ideas que

tuvimos. La última dificultad que afrontamos en este proyecto fue entender cómo

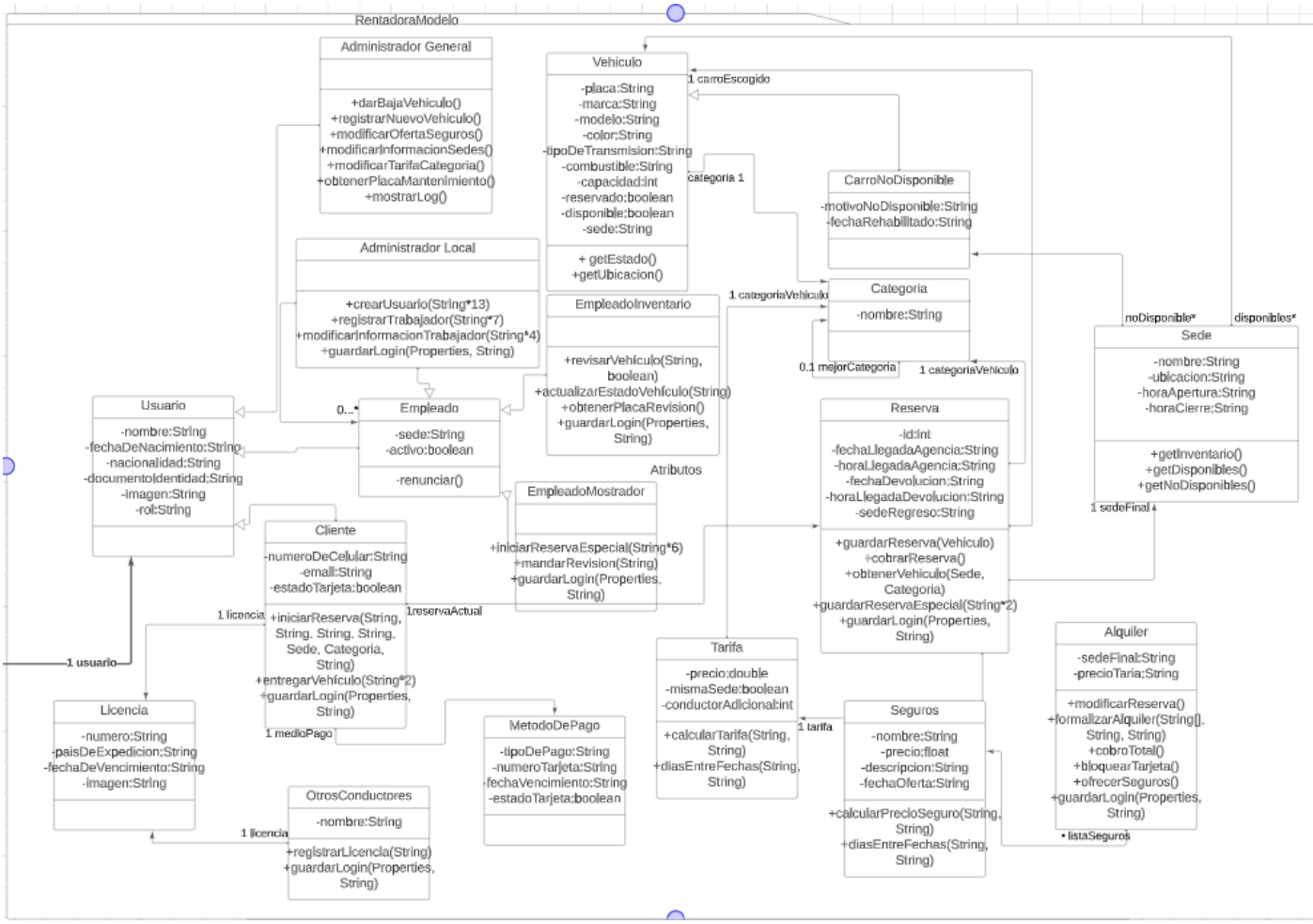

Imagen 1.0

manejaremos la reserva y el alquiler para que la lógica de una clase no hiciera conflicto con la otra. Por ejemplo, que si se hacían varias reservas desde un mismo usuario y luego se formaliza un alquiler, se borraran las otras reservas.

Aprendizajes:

En este proyecto se logró aprender a manejar archivos .txt, escribirlos con una estructura específica, leerlos, guardar su información y escribir nuevos datos. Adicionalmente, nos familiarizamos más con los diagramas de clases y el proceso previo a escribir código, esta segunda parte nos ayudó con el segundo proyecto ya que teníamos claros los pasos a seguir antes de escribir la interfaz.

Debilidades:

Al revisar nuestro resultado final, nos dimos cuenta que la consola quedó bastante larga lo cual hizo dificil releer el código para entender cómo adaptarlo a la nueva interfaz del proyecto dos. También, por temas de tiempo, no logramos implementar completamente el código por ejemplo para añadir el precio de los seguros, ya que la clase existe y los seguros también pero tienen precio adicional 0 entonces no afecta al final.

Fortalezas:

Creemos que para nuestro primer proyecto logramos dividir las responsabilidades correctamente al igual que la persistencia. Así, no hubo confusiones cuando manejamos los datos ya puestos o al añadir datos nuevos. De la misma forma cuando la aplicación se apaga, los datos quedan guardados correctamente.

Consejos:

Los siguientes consejos abarcan partes de las fortalezas, debilidades, aprendizajes y dificultades.

- Subrayar el documento con diferentes aspectos que sirvan para el diagrama
- NO usar el primer intento de diagrama para empezar a programar
- Dividir la consola en diferentes clases para cada tipo de usuario.
- Repartir la persistencia de manera tal que al cerrar la aplicación no sea necesario cargar nuevos datos, imagen de nuestra persistencia (Imagen 1.1)

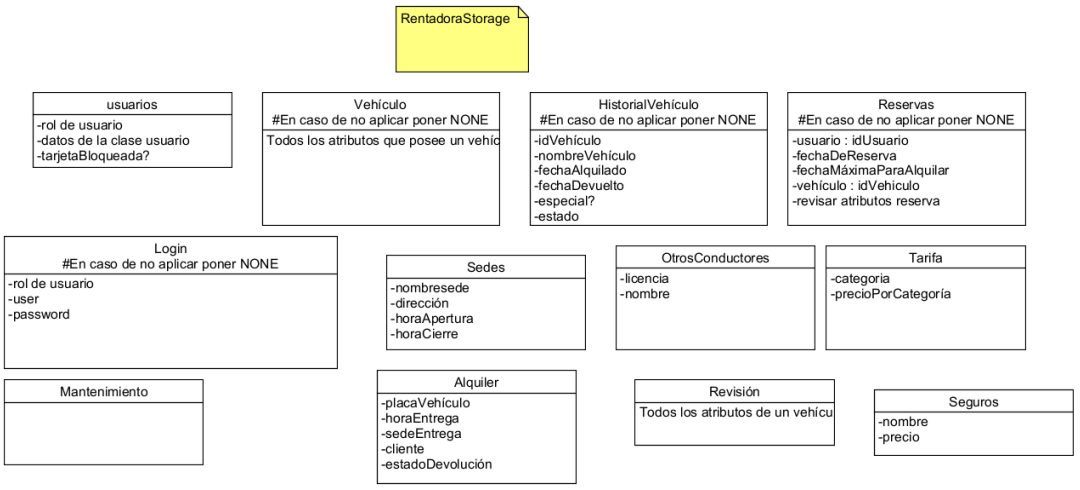

Imagen 1.1

Proyecto 2:

El proyecto dos y todo lo que conlleva fue en su mayoría creación e implementación de interfaces, solo que ya no basadas en consola, sino en el framework SWING (algo que nunca habíamos hecho) por lo que naturalmente surgieron problemas en su implementación. Igualmente, para poder crear la interfaz de manera organizada, decidimos usar la misma estrategia del proyecto 1, donde después de una lectura individual, hicimos una lectura grupal donde llegamos a diferentes conclusiones de cómo abordar este nuevo reto.

Dificultades:

Una problemática notable fue la concentración de grandes bloques de código en una sola clase, lo que generó desorden al migrar todo a la interfaz de usuario (UX). La persistencia de datos y su integración con la interfaz gráfica siempre representaron un desafío lógico que ralentizó el progreso y exigió la búsqueda de nuevas soluciones. Adicionalmente, al no tener experiencia previa con el framework nos vimos obligados a buscar la documentación en internet para solucionar las dudas que tuviéramos. Añadiendo a lo anterior, fue difícil conectar los paneles para que se comunicaran, es decir, que estando en un panel no se tiene información de lo que se encontraba en un panel anterior ya que no hay por parámetros entre paneles. Un ejemplo de lo anterior: estando en el panel de reserva con un cliente y cambias de panel, la información del panel de reserva no se va a encontrar directamente en el nuevo panel.

Dada la naturaleza laboriosa de este trabajo y los numerosos requisitos del proyecto que demandaban múltiples interfaces, la gestión de la información y la comunicación entre tantas interfaces dificultó la escritura del código.
Para abordar este problema, implementamos el patrón fachado. Asignamos el cardlayout al JPanel y a todas las demás clases, de modo que, para cambiar la interfaz, debían llamar a la clase principal, simplificando así esta acción. Además, para facilitar la comunicación entre paneles sobre la información ya clickeada al cambiar de panel, se estableció un mecanismo que evitó la necesidad de utilizar numerosas variables públicas. Esto mejoró la estructura del código y contribuyó a una transición más fluida entre las diferentes secciones de la interfaz.

Aprendizajes:

En este proyecto conseguimos un nivel básico en el manejo de interfaces gráficas basadas en SWING también a cómo unir código base de la aplicación con nuestra UX. Fuimos conscientes que decisiones de ventanas muy estilizadas requieren mucho trabajo, por lo que quedarnos con los aspectos más simples del framework ahora que estamos empezando siempre es mejor idea. También, entendimos cómo manejar los action listeners y de los mayores aprendizajes fue el manejo de errores y cómo hacer para mostrarle la existencia de un problema al usuario. Finalmente, en términos grupales, fue complicado adaptarse a leer código ajeno y entender la lógica de otra persona pero al lograrlo es una herramienta muy valiosa para futuros trabajos donde eso ocurre muy seguido.

Debilidades:

Consideramos que una debilidad que tenemos en este proyecto es que la clase principal tiene muchos atributos públicos lo que puede poner en riesgo la seguridad del programa. Tenemos una clase de verificador que en este momento consideramos que debería ser abstracta, para que haya pequeñas clases de verificación para manejar diferentes tipos de excepciones.

Fortalezas:
Para ser la primera vez usando SWING tuvimos una buena experiencia, y aunque costó siento que el resultado del proyecto como grupo fue satisfactorio. También como en el proyecto 1, aunque haya sido complicado manejar la persistencia, lograrlo de manera tan óptima generó que la implementación funcione más acorde a lo deseado.
La aplicación, por la manera en que implementamos la parte gráfica del diseño, resultó ser bastante intuitiva. Además como tiene bastantes manejos de errores, es muy complicado que la aplicación se cierre por un problema del usuario.
En términos de organización, decidimos poner una nueva clase de cargar archivos, y la consola la separamos en varios paquetes. Esto hace que siempre haya persistencia, los datos se guardan después de cualquier acción, por lo que la aplicación puede cerrarse en cualquier punto y los datos no se pierden, no hace falta 'guardar antes de salir'.

Consejos:
- Cada ventana tiene que tener un botón de volver a la pestaña anterior
- Programar con el fin de hacer todo reutilizable para reciclar código: A diferencia de los mini proyectos que hacíamos antes esta es una aplicación muy robusta, y no podíamos usar el mismo enfoque. Nos sirvió de mucho estandarizar funciones y comportamientos para escribir menos código.
- Intentar tener reuniones en donde se tengan discusiones sobre el código hecho en conjunto

Proyecto 3:

El objetivo de este proyecto es poder aumentar el manejo de errores para la aplicación e implementar nuevas funcionalidades. Para este último ya nos sentíamos más cómodos para crear un plan de acción antes de poder hacer código. Por lo que el proceso de diseño fue mucho menor.

Dificultades:
Dentro de los nuevos aspectos que teníamos que entregar con ese proyecto, está la emisión con factura en PDF, para esto teníamos libertad de elegir la librería por nosotros mismos. Sin embargo, este proceso fue más extenso de lo que creímos ya que, a pesar de haber encontrado varias librerías que decían cumplir con la generación de PDF, la mayoría de estas realmente no lo hacían como a nosotros nos convenía, además que varias de estas no tenían documentación haciéndolo aún más difícil.
En el caso de las pruebas, esto resultó ser más complicado de lo que esperábamos en especial porque surgió un inconveniente inesperado. Al empezar a hacer pruebas en la clase reserva se empiezan a crear datos nuevos o cambiar datos antiguos, esto hace que al finalizar las pruebas toque revertir los cambios en los documentos y datos hechos por estas ya que pueden quedar datos como una reserva sin utilizar. Entonces para hacer eso, fue necesario crear nuevas funciones para revertir estas pruebas. Otro ejemplo de estos casos es que al hacer las pruebas

de crear reservas, llegó un punto en donde todos los vehículos estaban en estado no disponible.

Fortalezas:
En este proyecto nos dimos cuenta que desde el proyecto dos ya habíamos empezado a implementar manejo de errores, por lo que para este trabajo necesitamos menos tiempo. Por otro lado, al haber organizado las clases y paquetes tan individualmente, no fue necesario conectar el trabajo de las nuevas implementaciones sino que solo se añaden ahorrandonos así bastante tiempo.

Debilidades:
Dentro de la aplicación, en la parte del pago, si el cliente hace un primer pago y la aplicación lo acepta pero por error el cliente hace un segundo pago y no es aceptado, se anula todo el proceso. Con esto en mente, el cliente puede pagar más de una vez una misma reserva.
Consejos:

- Aprender sobre instanciación dinámica desde el inicio del curso para poder usarla en el login ya que sirve para dar permisos más fácilmente.

Reflexión final:

A pesar de las dificultades que tuvimos de manera individual y grupal, logramos terminar los tres proyectos de manera satisfactoria. En términos grupales, sentimos que uno de los mayores problemas fue el juntar el código hecho por otro integrante con el propio, sin embargo, tener constantes reuniones de trabajo para manejar eso fue clave. Ahora, sabemos que para futuros proyectos debemos hacer un diseño previo antes de ponernos a hacer el código para evitar confusiones en revisiones futuras. Igualmente, aprendimos cómo editar archivos y manejar la persistencia adecuadamente para que no sea necesario guardar la información manualmente.