PROYECTO 3

DOCUMENTO DE DISEÑO

Camila Fajardo
Juan Francisco Rodríguez
Juan Diego Osorio
Laura Valentina Lara Díaz

2023

Responsabilidades

1.Cliente:

1. Reservar vehículo
2. Modificar reserva existente
3. Registrarse

Las responsabilidades este componente nos indican que es un Information Holder, ya que ahí se mantiene y se entrega información a varios componentes del sistema que permiten la ejecución de varios requerimientos funcionales. Además, que desde un principio debe ingresar sus datos, los cuales serán guardados dentro de sí mismo, para poder acceder a los demás componentes del sistema. A este componente se le asignarán todos los valores y la información correspondiente a un alquiler, de manera que el sistema tenga donde guardar todo lo que necesita para llevar a cabo la reserva de un vehículo.

2.Inventario:

1. Modificar inventario (añadir o eliminar vehículos)
2. Acceder a información de los vehículos
3. Modificar información sedes

Las responsabilidades de la modificación del inventario, en donde se pueden añadir o eliminar vehículos de una sede, son controllers ya que en estos es donde se puede tomar la decisión de si una reserva se puede o no llevar a cabo, al ser estos los que definen la disponibilidad que tiene el carro en su agenda y la disponibilidad de carros de la sede. Sin embargo, estos también son information holders ya que la funcionalidad de distintos componentes del sistema depende de esta información que los definen.

3.Reserva:

1. Verificar que el cliente no tenga otra reserva vigente al momento de crear otra
2. Generar una reserva para un cliente

La verificación de la existencia de otra reserva vigente es un information holder ya que es a partir de aquí que se reciben datos de otros componentes del sistema para comenzar el proceso de la creación de una reserva. Los datos contenidos en la reserva van a tener que ser validados junto con los del cliente y el del inventario de la sede para poder asignar un carro con disponibilidad a la reserva, además de que son estos los que modifican esta información.

Colaboraciones

Crear reseva:

- Le indica al componente *AgendaDisponibilidad* los datos calendario de los vehículos reservados.
- Le indica al componente *VehiculoRentalSystem* los datos de reserva.
- Modifica datos del *inventario*

Crear alquiler:

- Se modifica el componente *inventario* y entrega datos a *VehiculoRentalSystem*
- El componente *VehiculaRentalSystem* termina el proceso con los datos de *cliente*
- Actualiza historial de reserva

Crear usuario:

- Le indica al componente *VehiculoRentalSystem* los datos de login de los nuevos clientes
- Se modifica el componente *VehiculoRentalSystem*

Modificaciones de administrador general:

- Se modifica el componente *Vehículo*
- Se modifica el componente *Seguro*

Modificaciones administrador local:

- Se modifica el componente *Empleado*
- Se modifica el componente *Sede*

Modificaciones por empleado:

- Modifica el componente *reserva*
- Se actualiza el componente *inventario*

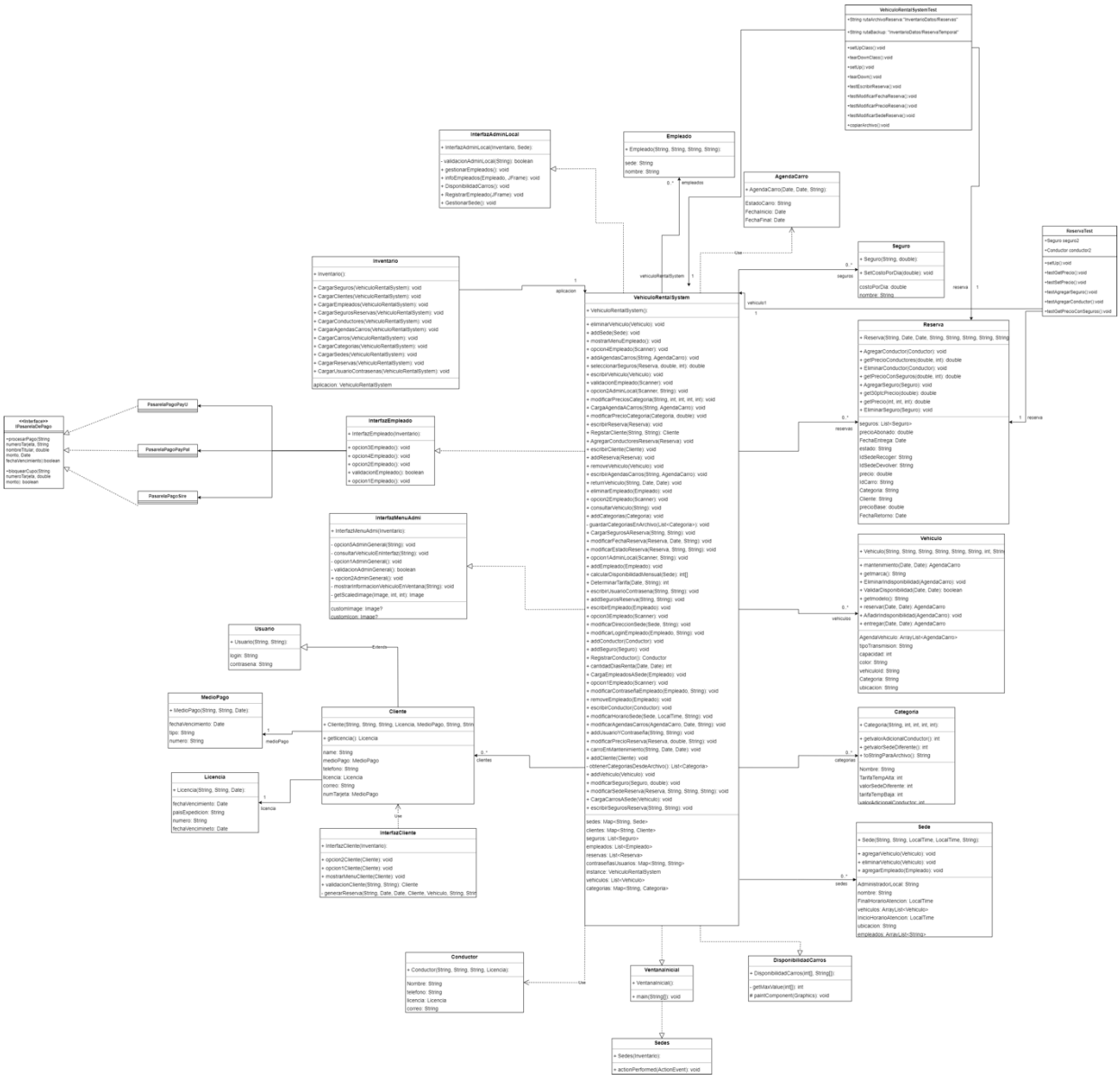
Restricciones del Proyecto

- Usuario: dependiendo del tipo de usuario (cliente, empleado, administrador general y administrador Local) las opciones de uso del sistema son diferentes.
- Zona Horaria: al momento de digitar las fechas, debido a que no todos los usuarios potenciales se encuentran en la misma zona horario, puede haber confusiones. Es por esto que se específica en qué formato se debe digitar.
- Alquiler: todos los procesos de alquiler de vehículos se guardan como reservas. Al realizar un alquiler se está realizando una reserva y se guarda así en los archivos. En caso de necesitar un alquiler instantáneo se llena el formulario para crear la reserva con la fecha actual y la fecha potencial de retorno.
- Política de cancelación: Definir las condiciones y tarifas asociadas a la cancelación de reservas.

- Multas y sanciones: Establecer políticas claras sobre multas y sanciones en caso de devoluciones tardías.

Diagrama de Clases de Alto Nivel

Este diagrama permite entender con mayor facilidad las relaciones entre clases y las implementaciones de las interfaces con el sistema. Como se puede observar las interfaces con el VehiculoRentalSystem se conectan con una relación de implementación. Adicionalmente, se encuentran las clases ReservaTest y VehiculoRentalSystemTest, las cuales pertenecen a un paquete aparte Tests, y utilizan el framework JUnit para implementar las pruebas.



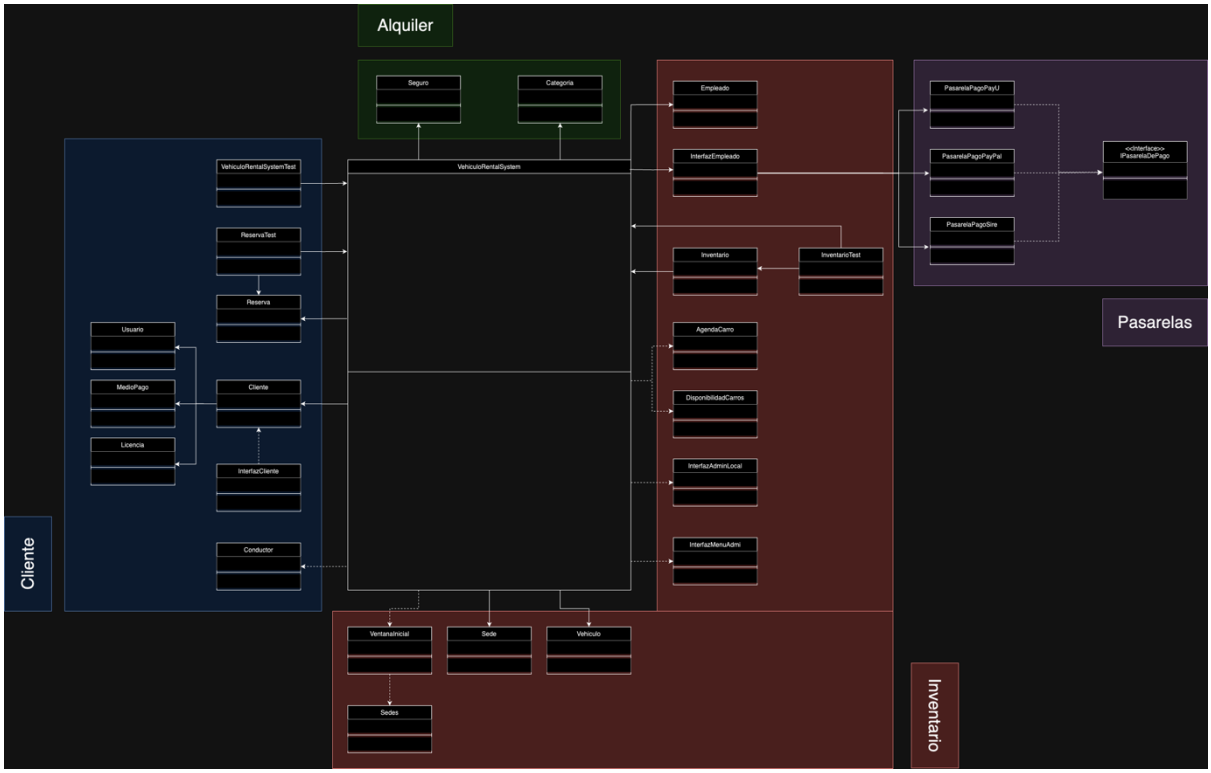


Diagrama de alto nivel de la interfaz:

Este diagrama permite entender qué elementos se incluyen para la creación de la vista del proyecto y cómo se relacionan con los elementos del dominio

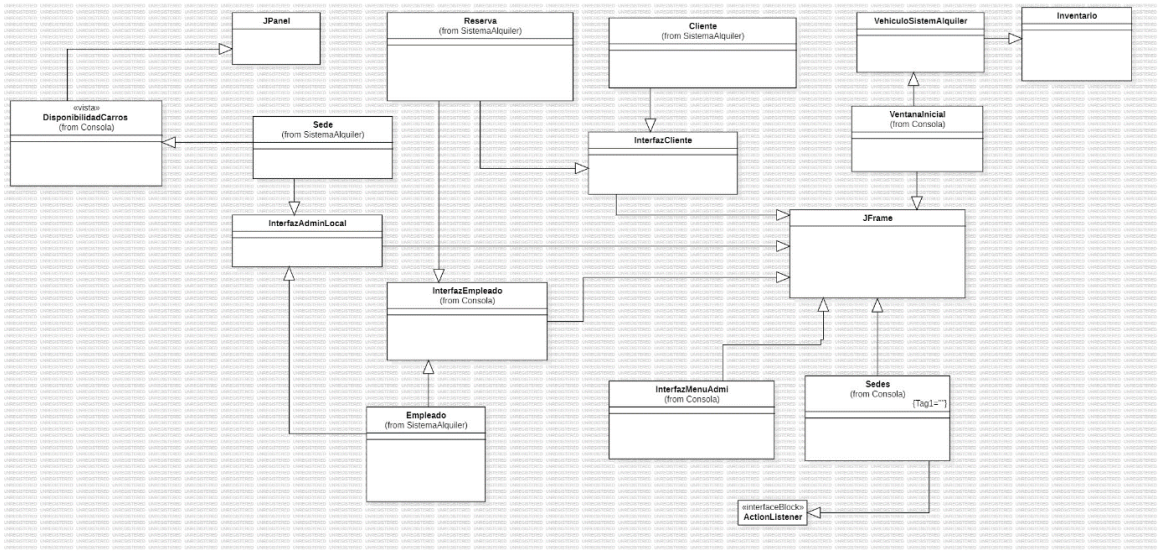


Diagrama de clases de la aplicación de clientes:

Este diagrama permite entender qué elementos se incluyen para la creación de aplicación para clientes que se encarga de generar reservas y mostrar la disponibilidad de carros para una sede durante un rango de fechas especifico.

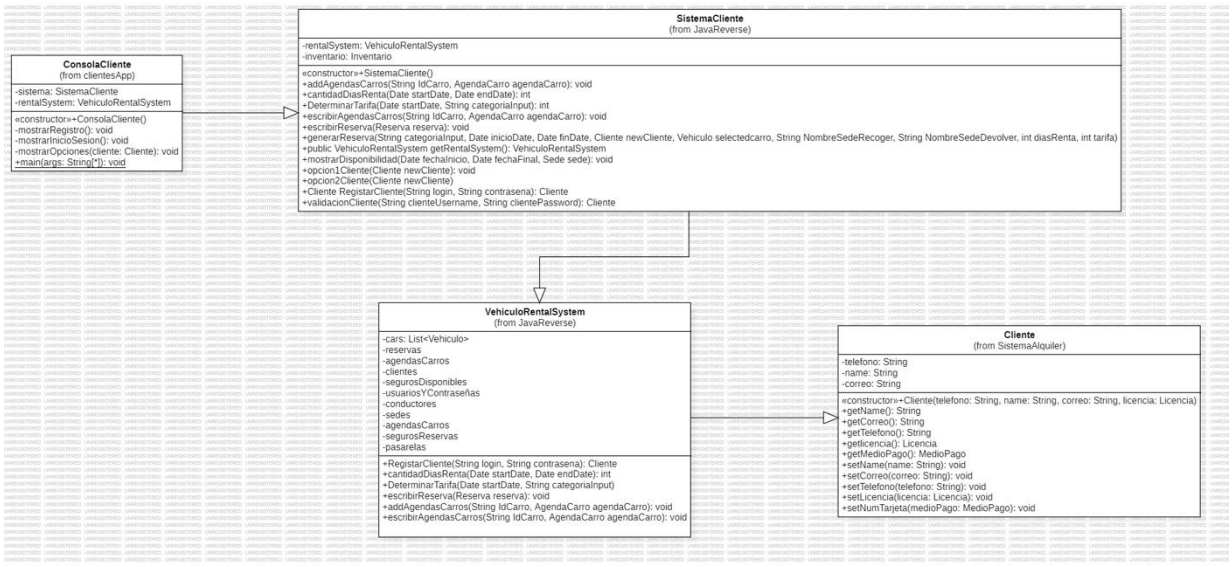
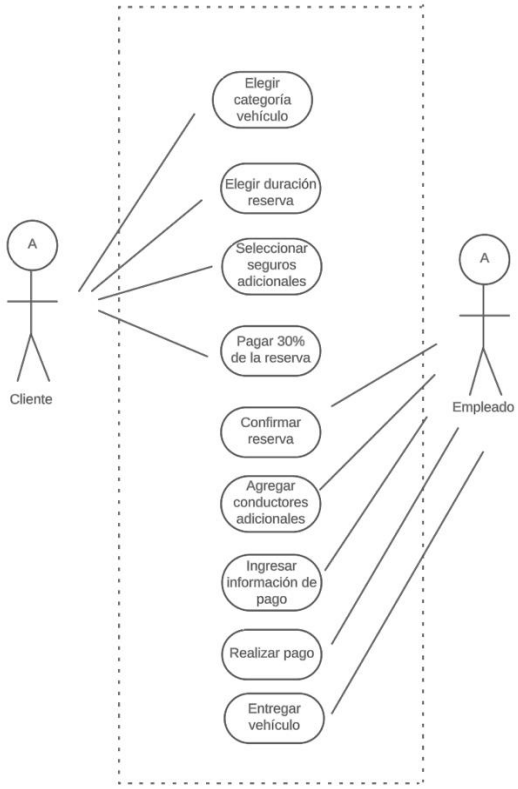


Diagrama de uso



λ+

TIPOS DE PROBLEMAS DURANTE EL DESARROLLO DEL PROYECTO

1.Problemas con las estimaciones de trabajo necesarias

La falta de consideración inicial en el plan respecto a aspectos específicos del comportamiento de la interfaz de usuario se convirtió en otro desafío sustancial durante el desarrollo. En particular, se observaron problemas relacionados con el manejo de ventanas, donde no se tenía en cuenta el cierre adecuado de las mismas al realizar ciertas acciones, como hacer clic en botones o en el icono de cierre (X) de la ventana. Estos inconvenientes no fueron previstos inicialmente en la estimación del trabajo, lo que llevó a que su resolución fuera más compleja de lo anticipado. La necesidad de abordar estos problemas no solo añadió una capa adicional de complejidad al proceso de desarrollo, sino que también subrayó la importancia de una planificación más detallada y exhaustiva que contemple posibles escenarios de interacción del usuario, para garantizar un funcionamiento suave y coherente de la aplicación. Este incidente resalta la importancia de la flexibilidad y adaptabilidad en el enfoque del desarrollo de software, especialmente cuando surgen desafíos no anticipados en el diseño inicial.

2.Desconocimiento de la tecnología

En el desarrollo del proyecto, nos enfrentamos a un desafío significativo relacionado con la generación de un PDF que contuviera la información de las reservas. Este problema surgió a raíz de la falta de familiaridad con la configuración de paths en Eclipse y la incertidumbre sobre qué versiones de librerías eran compatibles con nuestro entorno de desarrollo. La complejidad se vio agravada por la presencia de un error particular que resultó difícil de solucionar: en un equipo, la aplicación funcionaba correctamente con una versión más antigua de la librería, mientras que, en otro, con la misma versión, se encontraban errores inexplicables. Esta disparidad entre computadoras añadió un componente adicional de desafío técnico, que finalmente requirió un proceso de investigación minucioso y colaboración entre miembros del equipo para identificar y resolver eficazmente el problema. Estas experiencias resaltan la importancia de la gestión efectiva del desconocimiento tecnológico y la resolución de problemas en entornos de desarrollo variables.

3.Problemas realizando análisis de dominio

Durante el transcurso del semestre, nos encontramos inmersos en la complejidad de la gestión de cambios en los requisitos de dominio. Cada nuevo proyecto traía consigo la necesidad de incorporar funcionalidades adicionales, generando un escenario dinámico y desafiante. Esta situación reveló una problemática central en el ámbito del análisis de dominio: la dificultad para anticipar y adaptar nuestro modelo inicial a las cambiantes demandas del usuario. A medida que los requisitos evolucionaban, se evidenciaba la necesidad de revisitar el proceso de análisis y diseño, lo que subraya la importancia de una metodología flexible y receptiva para abordar los desafíos en constante cambio del desarrollo de software. La capacidad de ajustarse y redefinir estrategias de análisis de dominio resultó ser fundamental para garantizar que nuestras implementaciones estuvieran alineadas con las expectativas cambiantes de los enunciados de los proyectos.

4.Dificultad de diseñar en un entorno incierto

El proceso de diseño en un entorno incierto presentó desafíos significativos durante el desarrollo del proyecto. Inicialmente, se planteó un diseño basado en UML que, si bien parecía sólido en la teoría, se encontró con obstáculos sustanciales durante su implementación. La complejidad y las relaciones de ciertos componentes resultaron ser más intrincadas de lo anticipado, lo que llevó a la necesidad de revisar y ajustar parte del diseño original para mejorar la eficiencia y la coherencia. En particular, la relación entre ciertos módulos o la estructura de datos subyacente demostró ser más compleja de lo previsto. Esto llevó a replantear y reorganizar ciertas secciones del diseño para garantizar un rendimiento óptimo. La adaptabilidad se volvió esencial, ya que la comprensión completa del entorno y de los desafíos específicos solo se desarrolló a medida que avanzaba el proceso de implementación. Además, en el ámbito de diseño de interfaces, la transición desde un diseño visualmente atractivo en herramientas de diseño gráfico hacia su implementación real en el código presentó dificultades adicionales. Aunque se buscaba una interfaz estéticamente agradable, la uniformidad y consistencia en la apariencia de todas las ventanas y elementos visuales resultaron ser una tarea más compleja. Estos desafíos resaltan la importancia de la flexibilidad y la capacidad de adaptación en el proceso de diseño, especialmente cuando se opera en un entorno incierto, donde la comprensión total de los requisitos y desafíos solo se desarrolla con el tiempo y la experiencia.

COSAS QUE SALIERON BIEN

Al considerar qué cosas salieron bien y qué cosas salieron mal, así como analizar las decisiones acertadas y problemáticas, se obtiene una visión integral que contribuye a un aprendizaje valioso y a la toma de decisiones más informada. Este tipo de evaluación permite identificar las fortalezas a aprovechar y las áreas de mejora para futuras acciones. En este contexto, exploraremos los éxitos y desafíos, así como las decisiones acertadas y problemáticas que han surgido en el proceso o proyecto en cuestión.

- Logramos realizar con satisfacción que el sistema se integrara perfectamente con sinergia entre las clases, de este modo, logramos por medio de la persistencia de los datos que el sistema se preservara y fuera sostenible. Esta persistencia se logra desde la clase VehiculoRentalSystem, el cual una vez se crea la instancia desde el método main, esta carga los datos leídos desde los archivos txt y los guarda en estructuras de datos para su fácil acceso cuando sean requeridos.
- Logramos que el proceso de reserva sea exitoso y este se vea reflejado en el estado de disponibilidad de los vehículos del inventario correspondiente a la sede en la cual se realizó la reserva, esto por medio de la sinergia entre la clase VehiculoRentalSystem con los objetos Vehículo y Reserva. Este proceso de logra consolidar desde la clase InterfazCliente.
- Logramos trabajar utilizando Swing para crear las interfaces; creamos las clases InterfazAdminLocal, InterfazCliente, InterfazEmpleado, InterfazMenuAdmi y VentanaInicial, todas estas cuentan con una fácil interacción, intuitiva y simple pensadas para que los usuarios puedan hacer uso de ellas sin dificultad, además, todas ellas están conectadas entre sí con transiciones correctas, no obstante, se pueden mejorar diferentes aspectos.
- Logramos diseñar e implementar satisfactoriamente un sistema de cobro de acuerdo a las especificaciones del servicio de renta dadas por el usuario. Es decir, el cobro al usuario se realiza teniendo en cuenta la categoría del vehículo, la cantidad de días que lo rentará y si hay temporada alta o no, adicionalmente, si tiene conductores extra registrados y/o si adquirió algún seguro. Esto se logró mediante las relaciones de delegación e implementación que existen entre la clase VehiculoRentalSystem con los objetos Reserva, Vehiculo, Seguro, Categoria. Dichas relaciones es la que nos permite calcular el precio a cobrar al usuario de acuerdo a sus indicaciones.

- Logramos crear tres pasarelas de pago PasarelaPagoPayU, PasarelaPagoPayPal y PasarelaPagoSire, esto lo logramos por medio de una interfaz IPasarelaDePago, y estas clases implementan los métodos de procesarPago y bloquearCupo definidos por esta. Adicionalmente, hicimos un archivo con la configuración de las pasarelas, de tal forma que en futuras ocasiones se logre añadir nuevas pasarelas sin mayor problema. Por último, estas pasarelas al realizar la acción del método procesarPago, escriben en un archivo txt correspondiente a la pasarela la información de la transacción y su estado, de ser esta aceptada.

COSAS QUE SALIERON BIEN

Es imperativo examinar detenidamente las áreas que presentaron dificultades o no cumplieron con las expectativas. Identificar qué salió mal proporciona una valiosa perspectiva para comprender las causas de los desafíos y establecer estrategias de mejora. En esta reflexión, exploraremos las circunstancias y decisiones que contribuyeron a resultados no deseados, permitiendo así aprender de las experiencias y orientar futuras acciones hacia el éxito.

- Podemos mejorar en las interfaces y en la experiencia de uso de las mismas. Puesto que debido a que se trata de un proyecto educativo y de aprendizaje y los tiempos son justos, optamos por soluciones sencillas para lograr estas interfaces, y debido a que estamos aprendiendo, existen ciertos bugs que pueden surgir al realizar una acción fuera de lo común de lo que es el uso esperado de la aplicación. Como, por ejemplo, cuando se empieza el proceso de reserva, pero el usuario quiere cancelarlo, la aplicación se cierra del todo y toca empezar el proceso desde 0.
- Podemos mejorar en el diseño de las pruebas utilizando JUnit. Al realizar las pruebas nos encontramos con diversas dificultades, puesto que debido a la forma en que implementamos nuestro código, para realizar las pruebas nos encontramos con problemas como por ejemplo para probar la persistencia de los datos. Por ejemplo, queríamos probar que se preservara la persistencia de los datos probando el método modificarEstadoReserva(), pero para lograr esto queríamos escribir en un nuevo archivo que fuera exclusivamente para la prueba, esto no lo logramos debido a como está implementado el método que no recibe una ruta por parámetro, y hacer este cambio podría implicar cambios grandes en la aplicación.
- Se nos dificultó hacer una visualización de alto nivel que muestre gráficamente, para una sede, la cantidad de automóviles disponibles a lo largo de un año. Esto debido a que se nos dificultó entender cómo lograr hacer gráficamente utilizando Swing una solución para esto.

Decisiones Acertadas

Integración y Persistencia de Datos:

- Se logro una integración exitosa entre las clases, permitiendo que el sistema funcione con sinergia.
- La persistencia de datos desde la clase VehiculoRentalSystem garantiza que la información se conserve y sea accesible de manera eficiente.

Proceso de Reserva:

- La reserva de vehículos se refleja correctamente en el estado de disponibilidad del inventario correspondiente a la sede, gracias a la coordinación entre las clases VehiculoRentalSystem, Vehículo y Reserva.

Interfaces Gráficas (Swing):

- El uso de Swing para crear interfaces gráficas proporciona una interacción intuitiva y simple.
- Las interfaces están conectadas de manera coherente, permitiendo una navegación fluida entre ellas.

Sistema de Cobro:

- Se diseño y se implementó con éxito un sistema de cobro que tiene en cuenta varios factores, como la categoría del vehículo, la duración del alquiler, la temporada, conductores extra y seguros.
- Las relaciones de delegación e implementación entre las clases facilitan el cálculo preciso del precio a cobrar.

Pasarelas de Pago:

- La creación de tres pasarelas de pago (PasarelaPagoPayU, PasarelaPagoPayPal y PasarelaPagoSire) mediante la interfaz IPasarelaDePago es un enfoque modular y extensible.
- La configuración de las pasarelas en un archivo facilita la adición de nuevas pasarelas en el futuro.

Decisiones Problemáticas

Mejora en Interfaces Gráficas:

- Existen posibles mejoras en la interfaz gráfica para garantizar una experiencia de usuario más fluida.

Seguridad:

- Se podrían implementar medidas de seguridad adicionales, especialmente en operaciones sensibles como el procesamiento de pagos y el manejo de información personal.