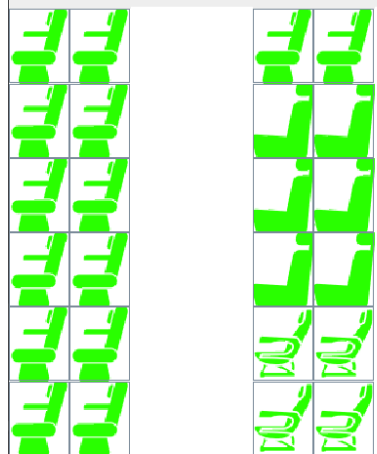


Proyecto final

Simulador de buses

Origen: SANTIAGO Destino: CONCEPCION Horario: Salida: 13:00 - Llegada: 15:00

Bus 1



Integrantes:	Camilo Tapia Matias Cruce (Grupo 6)
Fecha:	28/06/23
Tarea:	Proyecto final
Ramo:	Programación II
Profesor:	Geoffrey Hecht

Introducción

El sistema de reserva de asientos de autobús es una solución eficiente que permite a los clientes de una empresa de transporte seleccionar y reservar asientos de manera conveniente. Este sistema, referente a la propuesta de Ivonne Flores, ofrece una representación gráfica de los asientos disponibles en el autobús, lo que permite a los usuarios tomar decisiones informadas sobre su elección de asientos.

Cada asiento en el sistema está asociado con información detallada, como su estatus (disponible - ocupado), número y categoría, como "Premium" o "VIP". Los usuarios pueden navegar a través de la disposición de los asientos y seleccionar aquellos que deseen ocupar. El sistema verifica la disponibilidad en tiempo real y muestra el precio correspondiente a la reserva de los asientos seleccionados.

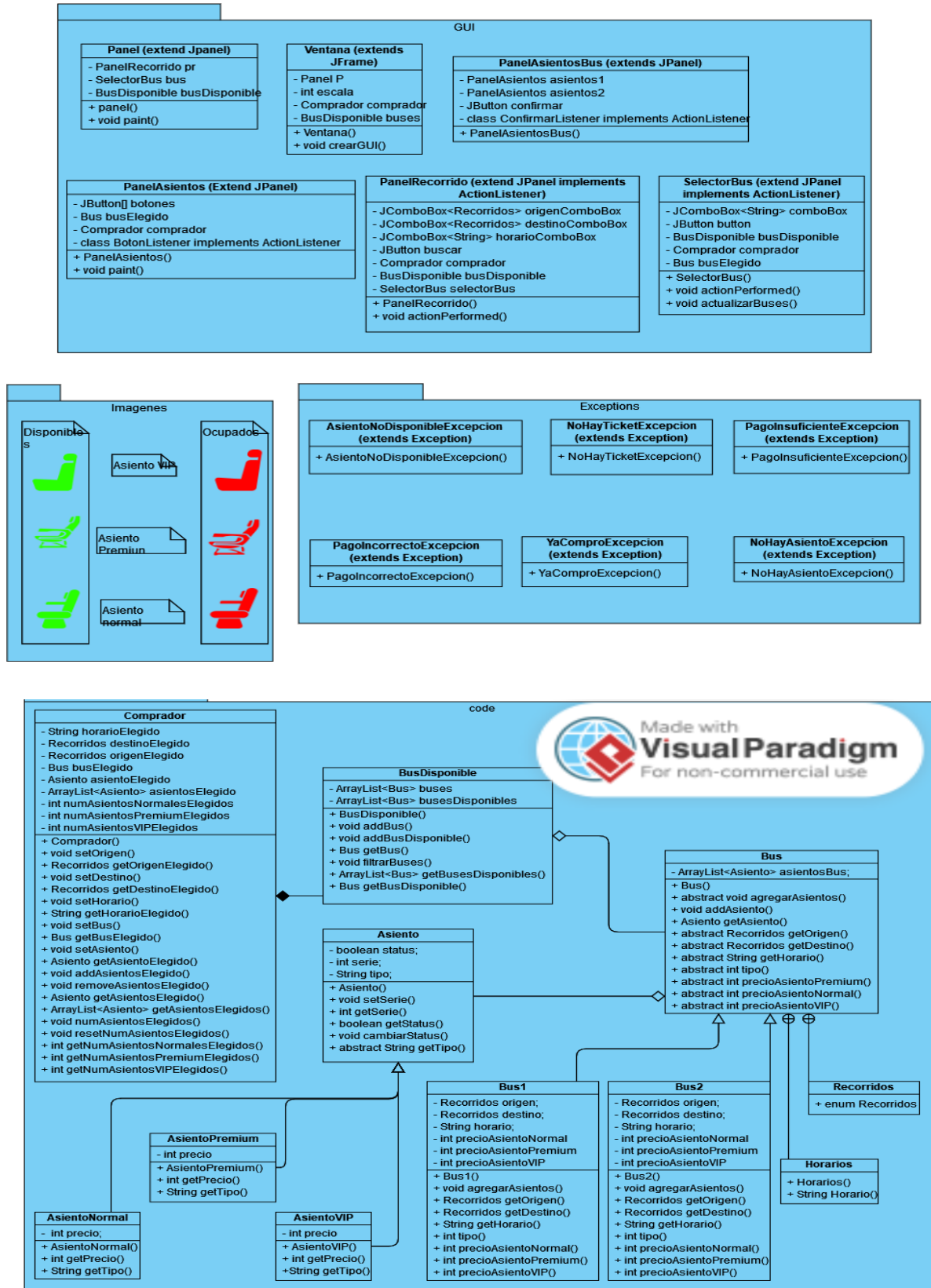
En caso de que algún asiento ya esté reservado por otro pasajero, el sistema notifica al usuario y le brinda la oportunidad de seleccionar otro asiento disponible. Esto garantiza que los clientes obtengan los asientos deseados y evita conflictos de reserva.

Desarrollo

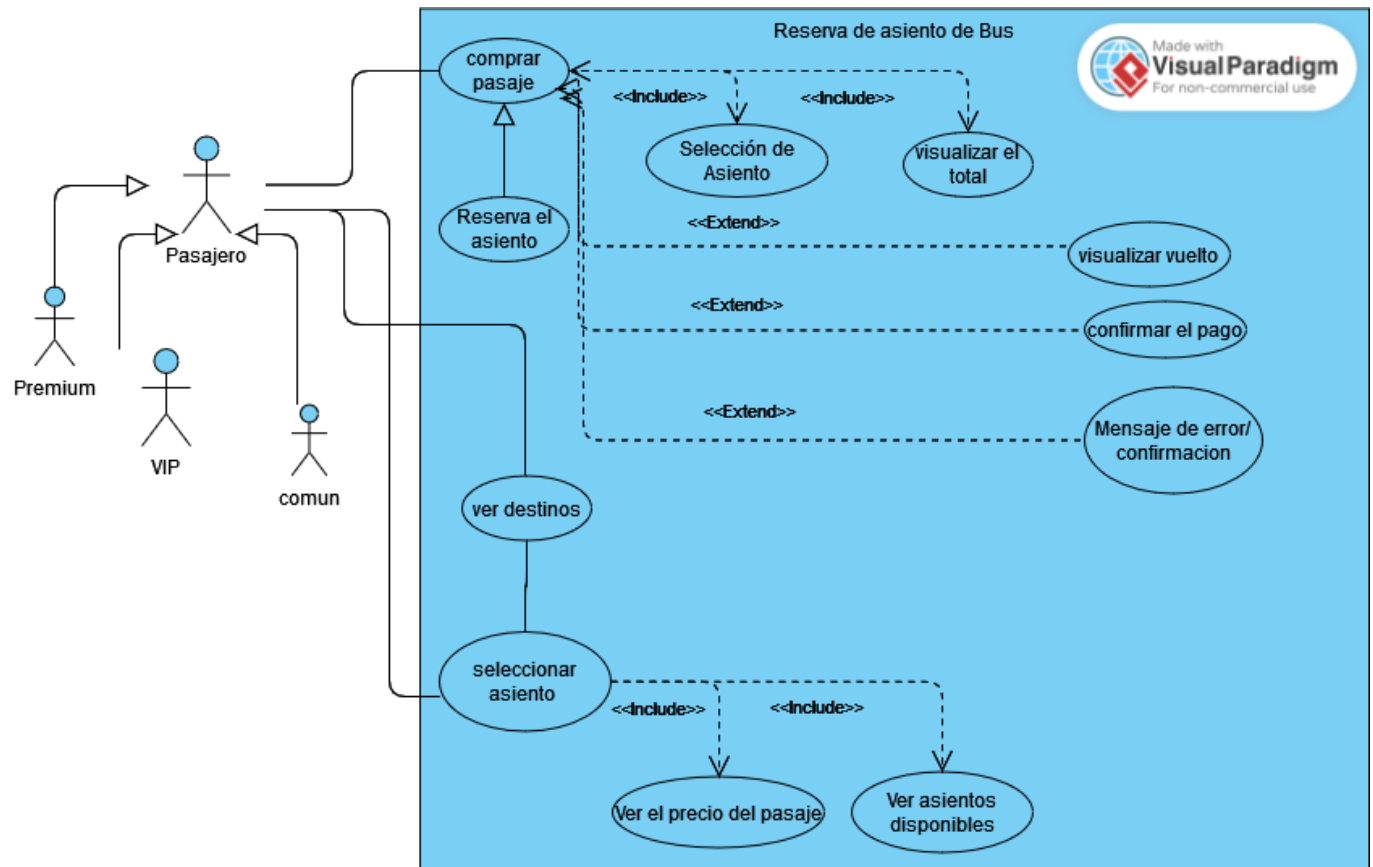
Imágenes utilizadas:

Disponibles	Nombres	Seleccionados	Ocupados
	Asiento Normal		
	Asiento VIP		
	Asiento Premium		

UML:



UseCase:



Interfaz programa:



Dentro del proyecto se intentó implementar varios patrones de diseño, unos con mayor éxito que otros. Entre estos encontramos:

- Patrón Observer: Este patrón fue utilizado porque teníamos la necesidad de crear una dependencia entre la clase “SelectorBus”, la cual actúa como el observador para recibir actualizaciones del objeto “BusDisponible”.
- Patrón Singleton: Se utilizó para que hubiera una única de las clases “Comprador” y “BusDisponible”.
- Patrón Composite: Empleado con la finalidad de tratar un grupo de objetos como si fueran un solo objeto, a la vez que se establece una jerarquía entre ellos. En nuestro caso en la clase “Ventana”, se agregan varios componentes (“PanelRecorrido”, “PanelAsientosBus”, “SelectorBus”) al panel principal “Panel”.
- Patrón Abstract Factory: Utilizado para crear familias de objetos, en nuestro caso las clases Asiento y Bus;
- Patrón Iterator: Al momento de recorrer los Arraylist para facilitar el almacenamiento de los objetos “Asientos”, que simulaban los asientos de un Bus.

Nuestro planteamiento para el desarrollo del programa consiste en que el usuario pudiera elegir a través de la interfaz gráfica un bus y asientos que ya estuvieran creados, por lo cual decidimos crear las instancia a estos objetos al inicio del programa.

Luego en la interfaz gráfica el usuario puede filtrar por origen, destino, horario y tipo de bus, para seleccionar el asiento que más le convenga.

Al momento de ya tener hecha su selección hay un botón que desplegará un aviso en la pantalla para confirmar los datos del asiento y del bus seleccionados, a su vez informa de los valores implicados y el total a pagar. Es aquí donde el usuario puede pagar o si bien lo desea, cancelar la operación, pudiendo modificar los parámetros que considere pertinentes.

En un comienzo íbamos a reutilizar el código de la tarea 2, adaptándolo a los requerimientos del proyecto, pensamos que esto nos iba a facilitar el trabajo. No obstante obtuvimos el efecto contrario, por lo que decidimos crear un código más limpio desde cero.

Durante el desarrollo del proyecto tuvimos una serie de problemas a la hora de implementar el código, en especial en la parte gráfica, en donde hasta la redacción del presente informe hay problemas que no pudimos resolver, como por ejemplo, no logramos que la interfaz se recargara correctamente, lo cual impide ver los cambios al momento de reservar un asiento. De igual manera ocurre al momento de querer realizar una segunda reserva, ya que no pudimos hacer que las variables implicadas se reiniciaran para realizar múltiples reservas.

Otro inconveniente que tuvimos a último momento fue el error al momento de actualizar el código, el cual gracias a github pudimos revertir, evitando perder todo el trabajo realizado.

Además de esto, como grupo atravesamos un gran número de inconvenientes, como por ejemplo la carga de trabajo de otros ramos, la gestión del tiempo y, como es común, la comunicación en el equipo de trabajo.

A pesar de lo anterior, intentamos llevar a cabo lo solicitado de la mejor manera posible, dando lugar a nuestro programa y el presente informe.

Conclusión

En conclusión, logramos poner en práctica los contenidos aprendidos durante el transcurso del semestre en la elaboración de este programa, siendo uno de estos el uso de github el cual resultó de gran utilidad al momento de controlar las versiones del programa.

A raíz de que hubieron problemas que no logramos solucionar, no hemos quedado del todo conforme con el resultado obtenido, ya que teníamos varias ideas que por diversas razones no pudimos implementar.

Como autocrítica destacamos el hecho de que tenemos que mejorar la comunicación durante el trabajo en equipo, a la vez de planificar mejor los tiempos y distribuir de forma más equitativa el trabajo.

Además de que por mala organización de algunos ramos no hemos podido estar presente la mayoría de los casos y nuestros cambios a pesar de estar de acuerdo eran muy vagos debido a que a veces corregimos aquello que ponemos y a veces funciones no servían.