

# PROYECTO FINAL

*Programación II.*



Camila Cortés Pérez // Natalia Contreras Aliaga.

Geoffrey Hecht  
Ivonne Flores

## ÍNDICE

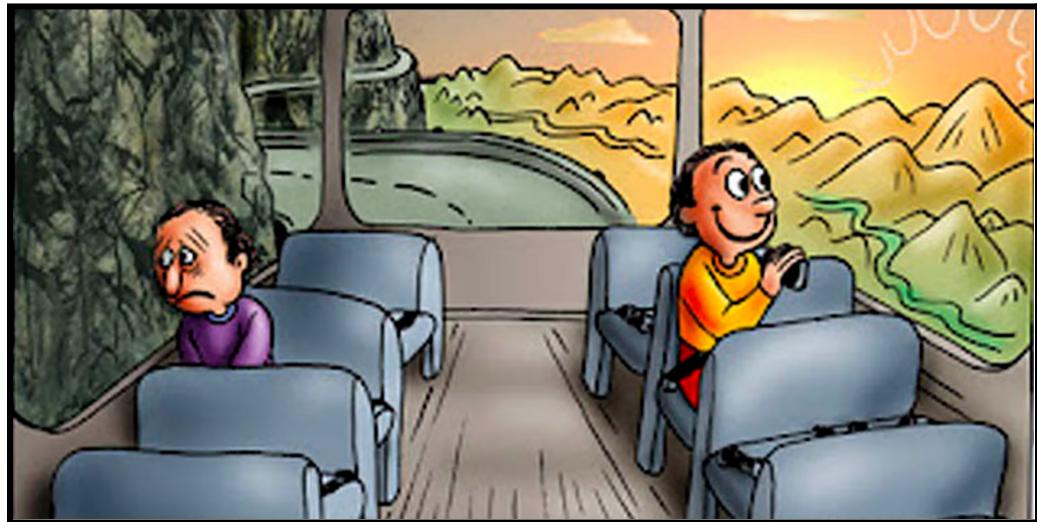
Introducción.....	2.
Enunciado General (Sistema de Reserva de Asientos).....	3.
Diagramas.....	4.
Patrones Usados.....	5.
Interfaz.....	6.
Decisiones Técnicas.....	7.
Problemas Encontrados.....	8.
Conclusión.....	9.

## INTRODUCCIÓN

La programación orientada a objetos es un paradigma de programación que parte del concepto de "objetos" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos.

Esta metodología fue estudiada a fondo durante el curso de Programación II, que ahora es implementada para nuestro proyecto a presentar "sistema de reserva de asientos de autobús".

A través de lo aprendido, implementando los conocimientos de las tareas anteriormente hechas, se crea un código que soluciona lo pedido. Además se presenta el proceso, las decisiones técnicas y las problemáticas.



## SISTEMA DE RESERVA DE ASIENTOS DE AUTOBÚS

La descripción del trabajo consiste en:

“El sistema de reserva de asientos de autobús permite al personal de una empresa de autobús elegir y reservar asientos de forma conveniente para su cliente. Los usuarios pueden visualizar una representación gráfica de los asientos disponibles en el autobús y seleccionar los que deseen ocupar. El sistema muestra información detallada sobre cada asiento, como su ubicación, número y categoría (por ejemplo, semi cama, Salón Cama).

Una vez que los usuarios seleccionan los asientos deseados, el sistema verifica la disponibilidad y permite confirmar la reserva mostrando el precio a pagar. En caso de que algún asiento ya esté reservado por otro pasajero, se informa al usuario para que pueda elegir otro asiento disponible. El personal confirma el pago (no gestionado por el sistema) lo que reserva los asientos.

El sistema debe gestionar varios tipos de autobuses (por ejemplo, con diferente número de plazas, o de 1 o 2 pisos...) y debe mostrar un menú que permita seleccionar el autobús en función de su horario y recorrido (se supone que estos datos están disponibles con los autobuses vacíos cuando se lanza el software)."

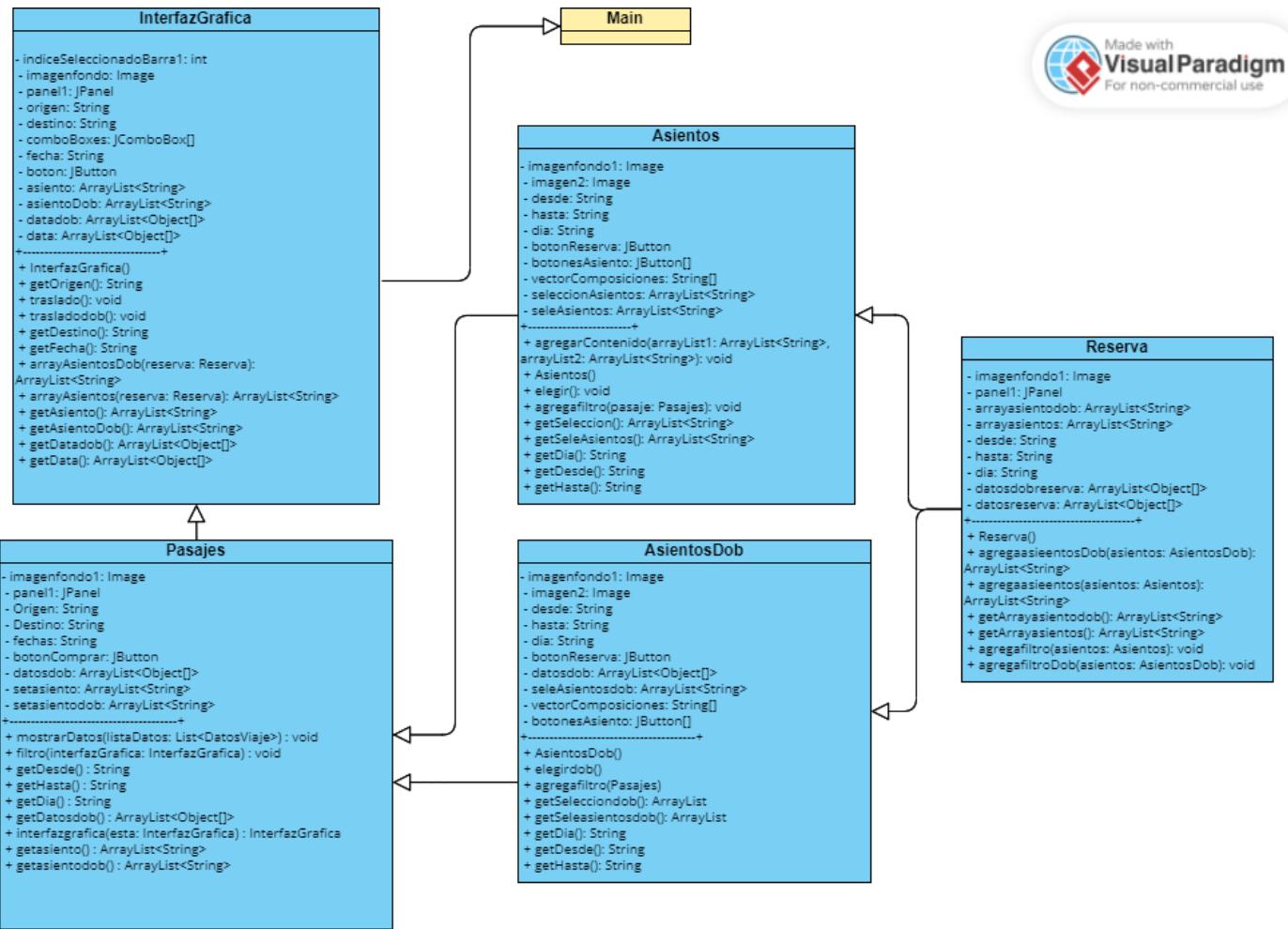
La primera versión del trabajo comentada por nosotras contemplaba comprar pasajes ida y vuelta, pero después de consultarla lo hemos reducido. También se había hablado de la primera interfaz contener los horarios, pero al final decidimos que era mejor que tuviese una propia.

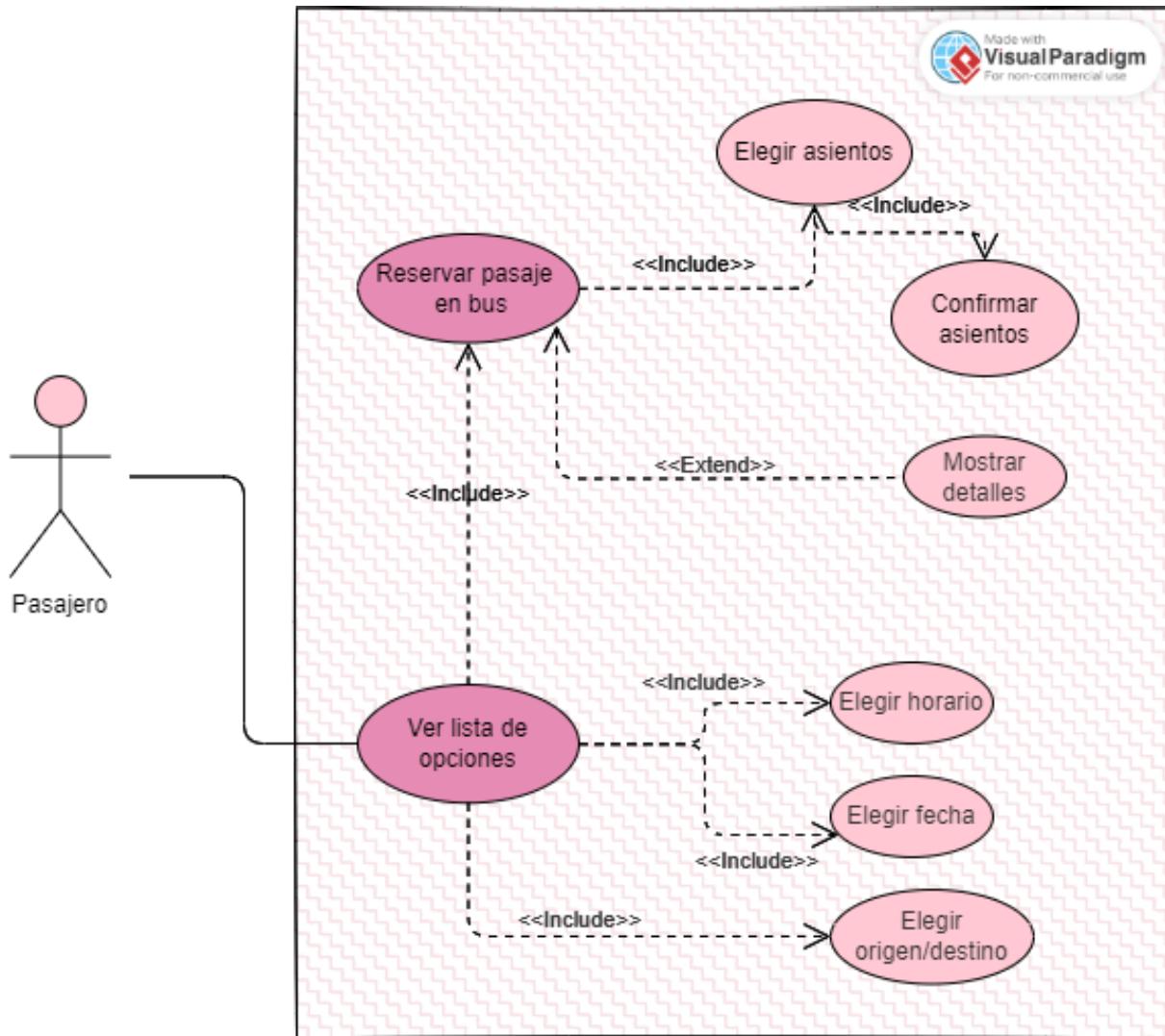
Esta primera entrega contenía inicialmente 3 ventanas (inicio con selección de origen, destino y día, selección de horarios y selección de asientos), además se había contemplado que los detalles de la compra se mostrasen en la interfaz de las selecciones de asientos. Después de empezar a avanzar, se decidió mostrarlo aparte ya que así se podría simular una ventana de confirmación de compra.

Finalmente, después de mucho avanzar, nos quedamos con 5 ventanas; inicio, selección de horario, selección de asientos (dos diferentes, para semi cama y salón cama) y finalmente la confirmación de la reserva.:

Lo más importante aquí es el uso de botones. Para hacer los asientos se decidió hacer todos los botones con un vector. Este crea una serie de botones de asiento en una interfaz gráfica. El bucle for se ejecuta dos veces: una vez para los índices del 0 al 19 y otra vez para los índices del 20 al 39. Cada iteración del bucle crea un nuevo botón de asiento, establece su posición en la ventana y le asigna un ícono.

## DIAGRAMAS





## PATRONES USADOS

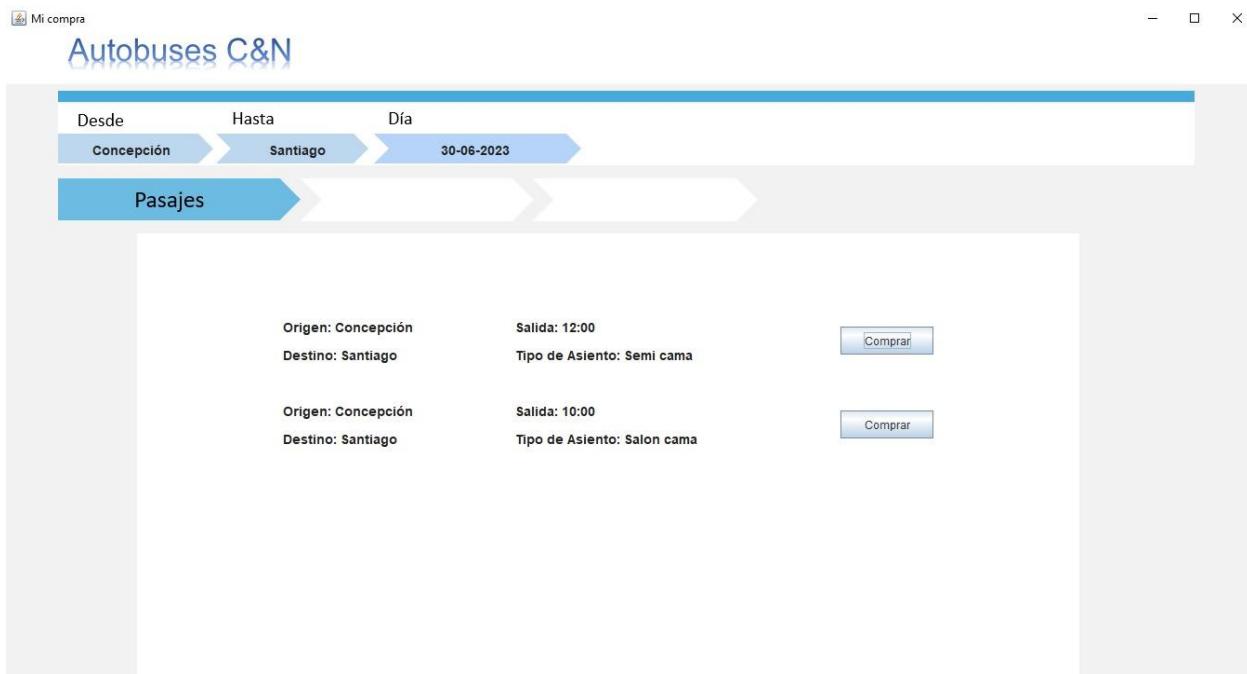
En la creación del código, se utilizó el patrón Modelo-vista-controlador (MVC). Este es un patrón de arquitectura de software que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

El patrón MVC separa la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador. Veamos cómo se aplican estos componentes en el código:

- **Modelo:** En este código, existe una representación de los datos y la lógica subyacente en las clases Ciudades, Pasajes, Asientos, AsientosDob y Reserva. Estas clases se utilizan para manejar la información relacionada con las ciudades, los pasajes de autobús y su reserva. El modelo se encarga de la lógica de negocio y el almacenamiento de datos.
- **Vista:** La clase InterfazGrafica extiende JFrame y representa la vista en este patrón. Se encarga de la interfaz gráfica de usuario, incluyendo la disposición de los componentes visuales como botones, etiquetas y barras desplegables. También muestra la imagen de fondo y los elementos de la GUI al usuario.
- **Controlador:** El controlador está representado por los ActionListener que se agregan a los componentes de la GUI, como los botones y las barras desplegables. Los ActionListener capturan las interacciones del usuario y definen el comportamiento de la aplicación en respuesta a esas interacciones. En este caso, los ActionListener están encargados de manejar los eventos de selección de las barras desplegables y el botón de búsqueda.

En resumen, el código utiliza el patrón de diseño MVC al separar la lógica de negocio y los datos en el modelo, la interfaz gráfica de usuario en la vista y la interacción del usuario en el controlador. Esto ayuda a mantener una estructura clara y facilita la comprensión y mantenimiento del código.

## INTERFAZ



Asientos

# Autobuses C&N

Desde      Hasta      Día

Concepción      Santiago      30-06-2023

Asientos

Piso 2		Piso 1	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>



Reservar

# Autobuses C&N

Desde      Hasta      Día  
Concepción      Santiago      30-06-2023

Compra

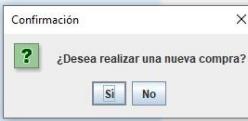
## DETALLES DE LA COMPRA

Tipo de Asiento: Semi Cama

Asientos : [C7]

Cantidad de asientos: 1 x 13000

Precio final: 13000



## DECISIONES TÉCNICAS

Al momento de crear el código, pasamos por diferentes procesos que nos llevaron a tomar diferentes decisiones:

1. VENTANAS: Al principio se había creado una sola ventana que se creaba desde Pasaje hasta Asiento, que representaba la selección del horario que dirigía a la selección de los asientos. Para poder distinguir entre Salón Cama y Semi Cama se decidió mejor separarlo en dos clases (Asientos y AsientosDob respectivamente), lo que permitía hacer dos redirecciones y además crear libremente los dos tipos de buses.
2. BOTONES: En un principio se había intentado hacer individualmente cada botón (que representan los asientos). Debido a complicaciones cuando se intentaba guardar las interfaces, además para optimizar y agilizar el proceso, se decidió crear un vector que creará todos los botones.
3. IMAGENES: Cuando se llegó al momento de darle un sentido más gráfico a la interfaz, en un momento se intentó crear colores y formas dentro del mismo código, pero fue mucho más fácil cuando se decidió que el fondo y los botones se crearan a partir de imágenes. Por ejemplo, cuando un asiento se reserva, lo que permite que cambie de color para mostrar que está reservado es simplemente el cambio del ícono del JButton.
4. GETTERS: Necesitábamos guardar la información de la primera interfaz y que llegase a la última, en Reserva. Para no tener que crear más punteros, se decidió que todos las clases tendrán getters, así, en la siguiente clase se puede utilizar y además pasar a la siguiente clase.

En general, se mantuvieron las primeras ideas compartidas en el grupo, ya que se discutían y después se ejecutaban.

## PROBLEMAS ENCONTRADOS

Durante el proceso de creación tuvimos que lidiar con diferentes problemas:

1. JLABELS Y ARRAYLIST: Cuando se intentó crear el JLabel para mostrar los detalles de la compra en Reserva, hubieron diferentes problemas. Primero, el ArrayList que contiene el número de asiento seleccionado aunque estaba siendo recibido correctamente desde las clases AsientosDob y Asientos, no estaba siendo impreso correctamente en la ventana. Para esto primero verificamos que el ArrayList estuviese funcionando de forma correcta; luego, vimos que como habían dos clases, había que en verdad recibir dos ArrayList (que, aunque suene obvio, en el momento no nos habíamos dado cuenta que no se había hecho). Luego de corregir ese problema, se pudo empezar a imprimir correctamente los ArrayLists.
2. ASIENTOS: Para crear y reservar los asientos, obviamente nuestra primera pregunta fue cómo los podemos seleccionar. Aunque ya se había decidido hacerlo con botones, la pregunta era cómo cumplir con lo demás. El primer paso fue crearlos, pero necesitábamos poder tener un botón que al seleccionarlo pudiese reservar y si se selecciona de nuevo lo quite. Así nos dimos cuenta que nuestra respuesta era más fácil de lo que pensábamos, era un switch. A partir de esto, las ideas empezaron a surgir y surgir, hasta incluso poder darle un identificativo a cada asiento.

Aunque surgieron más problemas dentro del trabajo, eran solucionados con sólo realizar una consulta. Las dos mencionadas fueron aquellas que más impactaron en la creación del código, pero al mismo tiempo fueron base en el avance y permitieron que surgieran más soluciones a incógnitas que aún no nacían.

## CONCLUSIÓN

Como conclusión, la creación de este proyecto de “sistema de reserva de asientos de autobús” permitió al grupo poner a prueba sus conocimientos aprendidos durante el curso de Programación II. Además, se tomó como vital el trabajo en equipo, que será vital en el futuro.

También, el buen uso de github, la buena comunicación y el uso de recursos permitió que no surgiesen mayores problemas al momento de crear el código. Se tomó en conocimientos los patrones de diseños, además de aprender a fondo el uso de interfaces gráficas, los UML y el uso de github y intellij.

