Universidad de Concepcion

Facultad de Ingenieria

Desarrollo Orientado a Objeto

Informe Proyecto 21: Aplicación de Reserva de Clases Particulares

Sebastian Nova Sanchez

2025402688

Introduccion:

Este informe, posee las siguientes secciones:

- Un breve resumen sobre el proyecto en sí, de que se trata, por que lo elegí, entre otros.

- El diagrama de casos de uso del proyecto.

- Una captura de pantalla de la aplicación mientras esta iniciada.

- El diagrama UML del proyecto.

- Una breve explicación de los patrones de diseño elegidos
- Un de las decisiones en mi opinión más importantes que tome al hacer este proyecto.

- Una review de los problemas que me encontré mientras realizaba el trabajo más una autocrítica a mi gestión de este.

El proyecto que decidí hacer, es una aplicación que permita reservar clases particulares para prepararse para rendir las pruebas de entrada a la universidad (Como la prueba Matematica M1, la prueba de Ciencias Mencion Fisica, entre otras), siendo posible interactuar con ella de 2 maneras:

- El usuario como cliente (alguien interesado en clases particulares)
- El usuario como profesor (alguien que añade clases disponibles para que los estudiantes puedan reservar)

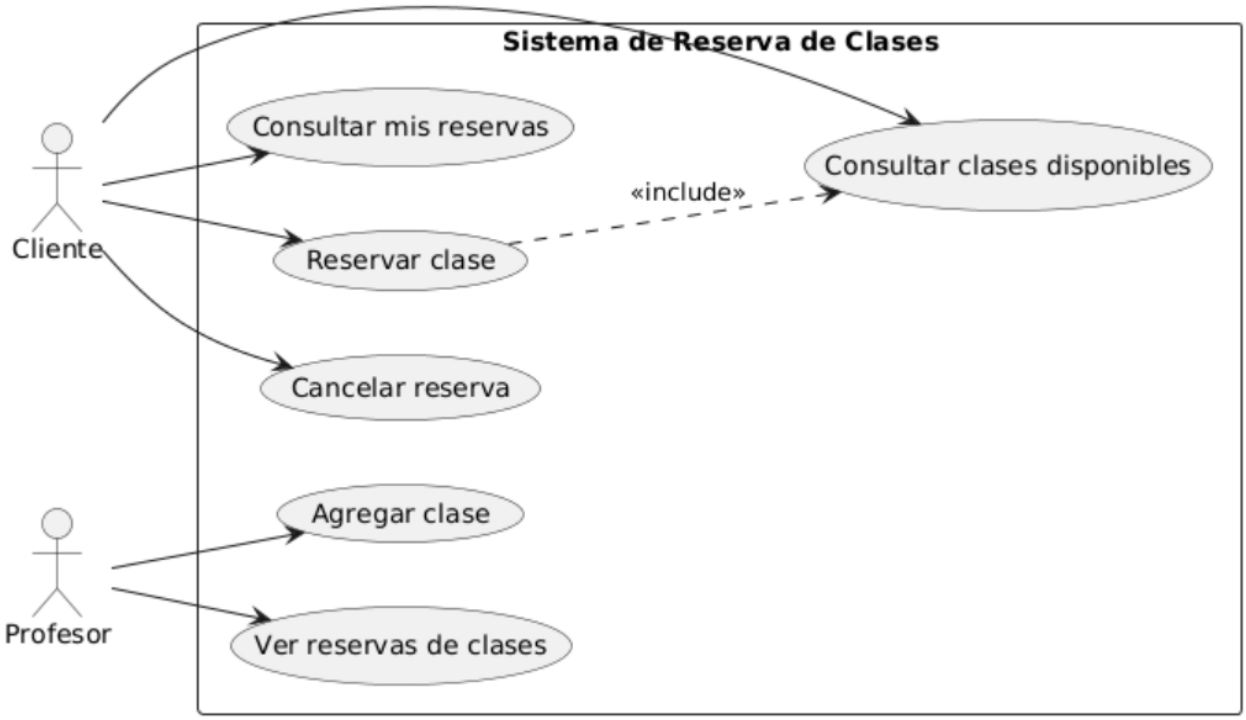
Elegí este proyecto entre las ideas disponibles por que sentí que me podía ayudar a pensar como desarrollador en el sentido de hacer cuadrar lo que ocurre detrás, el software que no se ve, el *"back end"* con lo que vera el usuario, el *"front end"*, las interfaces, las casillas de selección o los textos a ingresar, entre otras cosas. Siento que junto a las 3 tareas que hice este semestre es una muy buena muestra de lo que es programar pensando en la industria y utilizando herramientas más pesadas como swing. (Ya saliendo de la consola del IDE y entrando a código que abre aplicaciones)

Además de que encontré divertido (aunque a veces un poco abrumador) la cantidad de opciones y libertades que te da swing para realizar tus ideas (a veces, JAVA entrega demasiadas opciones, por lo que siento que haciendo este trabajo aprendí a discernir que métodos, librerías o herramientas usar entre las tantas disponibles para lograr un objetivo. El proyecto permite entrar como profesor, registrar nombre y prueba de especialidad, añadir clases personalizadas con "x" cantidad de estudiantes en "y" día a la hora "z" a un listado donde estudiantes registrados puede reservar una clase y elegir método de pago.

La interfaz es interactiva y funcional, permite pasar de profesor para añadir clases, a pasar a alumno para reservarlas, y se puede repetir el proceso sin tener que cerrar

la aplicación. Los estudiantes también después de registrarse y dar sus datos de contacto pueden cancelar las reservas hechas.

DIAGRAMA DE CASOS DE USO



CAPTURA DE PANTALLA DE LA INTERFAZ

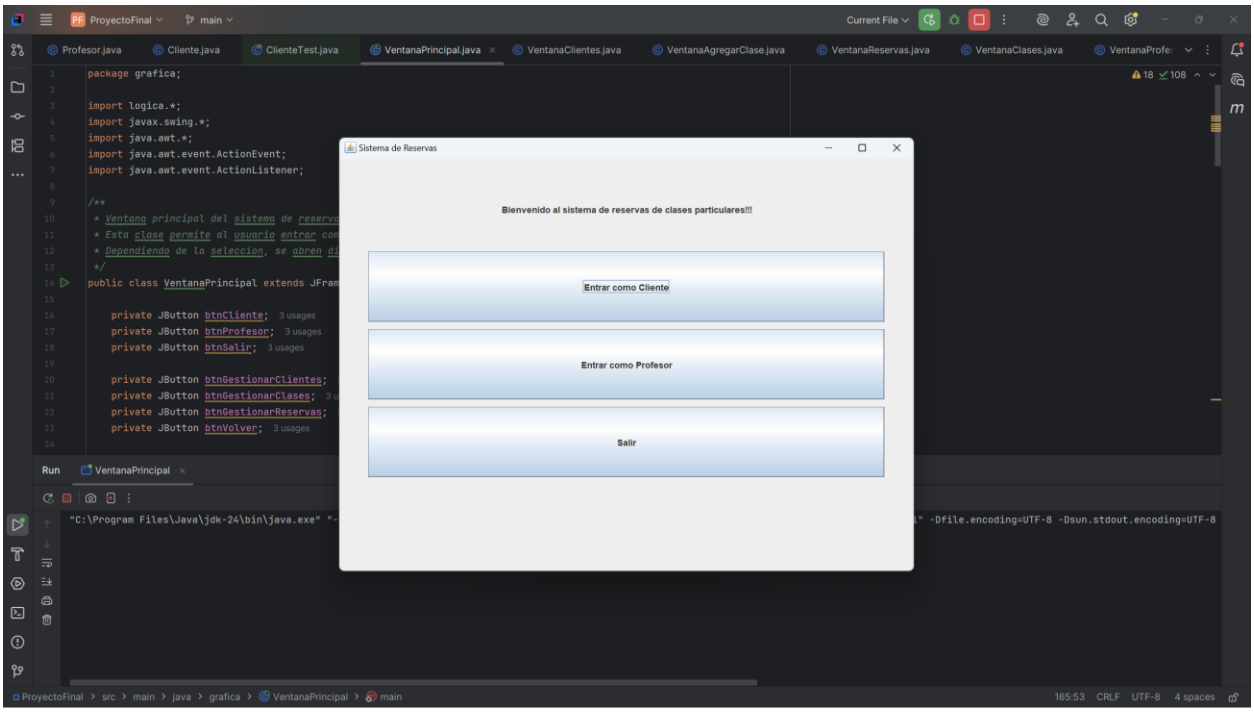
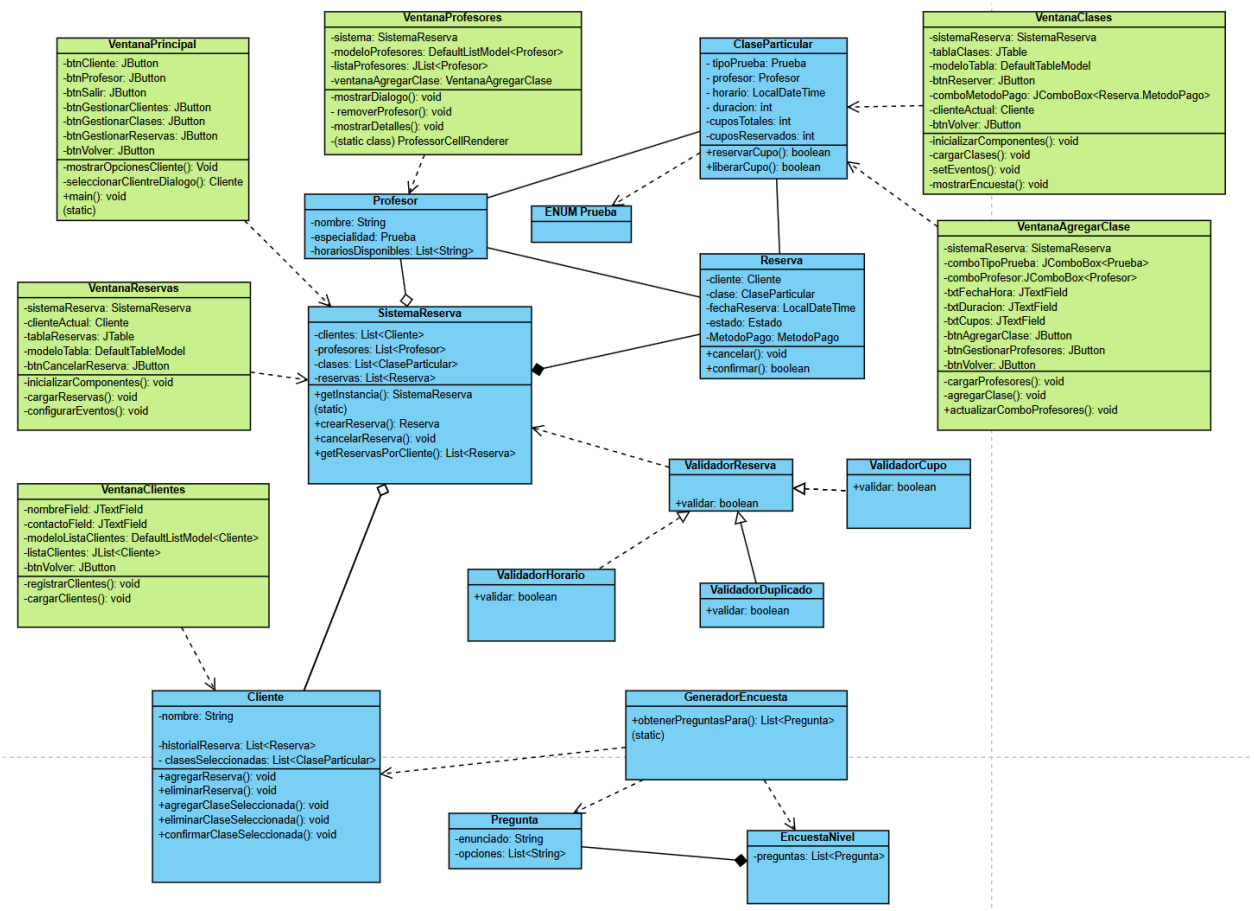


DIAGRAMA UML DEL PROYECTO (Con package logica y grafica)



USO DE PATRONES EN EL PROYECTO

En el desarrollo del proyecto, incorpore dos patrones de diseño:

Strategy y **Singleton**.

El patrón **Strategy** se aplica en las validaciones de reservas, donde se usa para verificar condiciones específicas antes de confirmar una reserva. Las clases que implementan este patrón son ValidadorCupo, ValidadorDuplicado, ValidadorHorario y la interfaz ValidadorReserva, cada una encargada de validar aspectos distintos, como la disponibilidad de cupos, la existencia de reservas duplicadas y la disponibilidad de horarios. Este enfoque permite añadir o modificar las validaciones de forma flexible y "desacoplada" de la lógica principal, ayudando con avances limpios en el sistema.

Por otro lado, el patrón **Singleton** se utiliza en la clase SistemaReserva, donde se asegura que solo existira una única instancia del sistema durante la ejecución del programa. Esto se logra mediante el metodo estatico getInstancia() y un constructor privado, lo que impide la creación de instancias adicionales fuera de la clase. El patrón Singleton garantiza que todas las operaciones de gestión de reservas, clases y clientes se realicen sobre una única instancia, lo cual es esencial para el funcionamiento del sistema. Ambos patrones contribuyen a un diseño del package Logica más robusto, flexible y fácil de mantener.

DECISIONES IMPORTANTES, PROBLEMAS IDENTIFICADOS Y AUTOCRITICA

Algunas decisiones importantes tomadas durante el proyecto fue como organizarlo con el tiempo que tenía disponible, muchas veces el momento donde tenía disponibilidad total para avanzar en el proyecto era en la madrugada, así que a esa hora realizaba avances.

Otra decisión importante tomada en este ámbito fue hacer todo el package lógica antes del package gráfica, para evitar tener problemas en el código que involucraran ambas partes del proyecto.

En general los errores en la parte grafica suelen dar más problemas que los errores en la parte lógica (al menos en mi experiencia), y cuando se mezclan es aún peor, así, desarrollando la parte lógica primero, se logra una base sólida para empezar a trabajar en la interfaz, de este modo, si más adelante se necesita un cambio en la parte lógica, este suele ser menor.

Otra decisión fue los patrones de diseño utilizados, para evitar poner patrones porque si, se tomó la decisión de usar solamente 2, pero que cumplen cosas importantes en el package lógica.

Algunos problemas que tuve fue con la interfaz gráfica del proyecto, en particular mantener el flujo del trabajo en orden, en un momento dado se quiso añadir que, al abrir la aplicación, preguntara si el usuario es cliente o profesor, lo cual terminé

implementando exitosamente pero no sin antes tener el problema de que al elegir que eras cliente, la aplicación bloqueaba el acceso al panel de cliente por que el cliente no estaba registrado registrado, pero para registrarse había que hacerlo desde el panel de cliente, entonces se armó una "paradoja" que se tuvo que arreglar.

También fue difícil implementar correctamente la encuesta en la aplicación después de reservar una clase, este proyecto me ayudo a entender las diferencias entre tener dificultad creando algo, y tener dificultad implementándolo correctamente.

Otra dificultad más fue el simplificar la parte grafica del proyecto para tener una versión completamente funcional y "testeable", una autocrítica que me hago es que no distribuí los tiempos de la mejor manera, estando demasiado tiempo pensando en el diseño general del proyecto y en el package lógica, en el cual estuve más tiempo del debido, lo que quizá me quito tiempo para el package grafica.

Este curso me enseño, que es mejor un poco de trabajo todos los días que mucho trabajo en poco tiempo, he logrado hacer estas tareas y proyectos en 3 o 4 días pero de forma muy exhaustiva, no es sostenible hacer trabajos en tan poco tiempo cuando requieren de 20 a 30 horas de trabajo efectivo (si no es que mucho más), incluyendo planteamiento del diseño general, 2 packages, pseudocodigo para entender la idea general y recién ahí empezar a ver que herramientas utilizar, es bastante exigente pero también me confirma que me gusta mucho la programación y los desafíos que entrega. Esta carrera es increíble y espero seguir mejorando y aprendiendo.