**Cambios adicionales en el aplicativo – Entrega 3**

**Integrantes:**

Maicol Yojan Antonio Rincón - 202027329

Juan Pablo Peña Jaime – 202212756

**Contexto del problema:**

Se desean realizar nuevas funcionalidades en el hotel. De esta forma, para dar inicio, con lo referente a la implementación de la información del hotel y los nuevos elementos de cada habitación, se extiende la funcionalidad original de la creación de una habitación. Así pues, será necesario agregar los componentes adicionales a la interfaz y, en términos de lógica se amplian los parámetros en el constructor del objeto de tipo Habitación.

Luego, para la implementación de las gráficas informativas para la administración, se realizan por medio de la librería JFreeChart, pues se desea que el administrador tenga la capacidad de obtener esta información por medio de gráficos legibles.

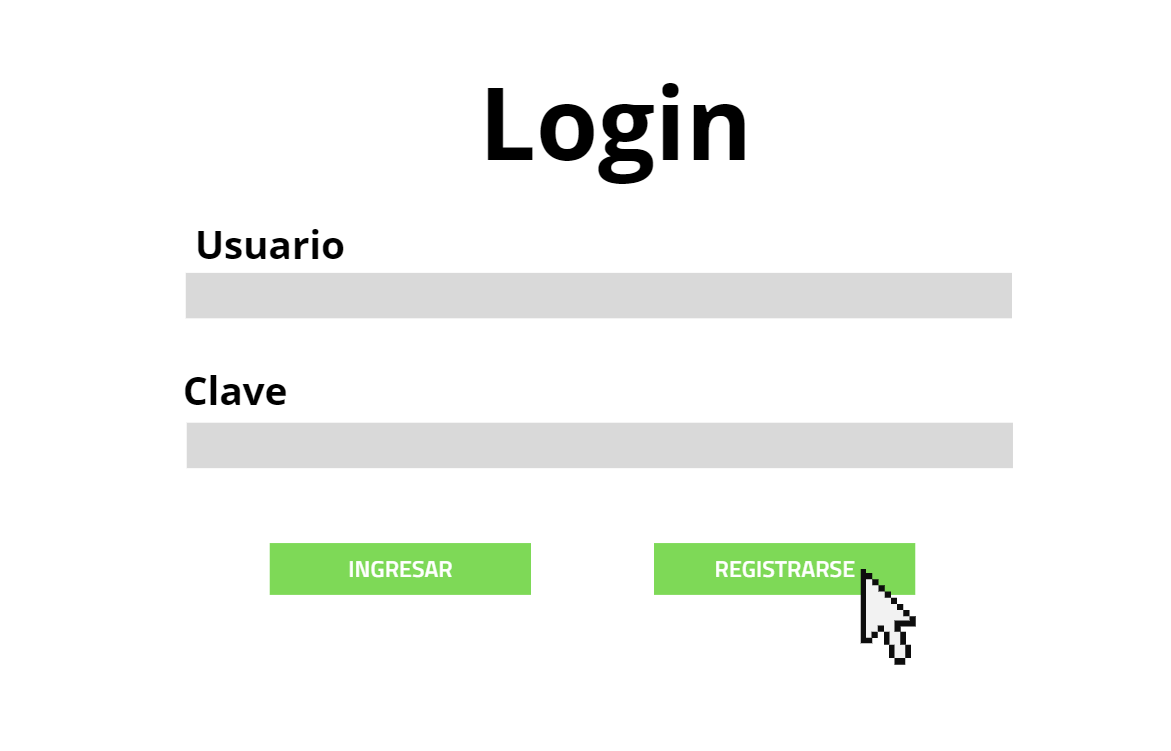
Por otra parte, se debe implemmentar una aplicación para huéspedes, en la que, con base en los requerimientos de reserva de habitaciones y obtener la disponibilidad de habitaciones, se debe desarrollar una interfaz completa, acoplada a la implementación de la nueva lógica capaz de soportar los requerimientos deseados.

Finalmente, deben realizarse pruebas unitarias y de integración apoyados en JUnit, para comprobar la realización de una reserva y la carga de datos referentes a la inforamción pertienente para el funcionamiento del hotel.

Así pues, una vez aclarados los retos de las nuevas implementaciones, se dispone el siguiente diseño para la interfaz en temas de la aplciación de huéspedes y la adición de los nuevos botones para los menúes de adminitrador y recepcionista:

**Diseño de la interfaz de la aplicación de huéspedes:**

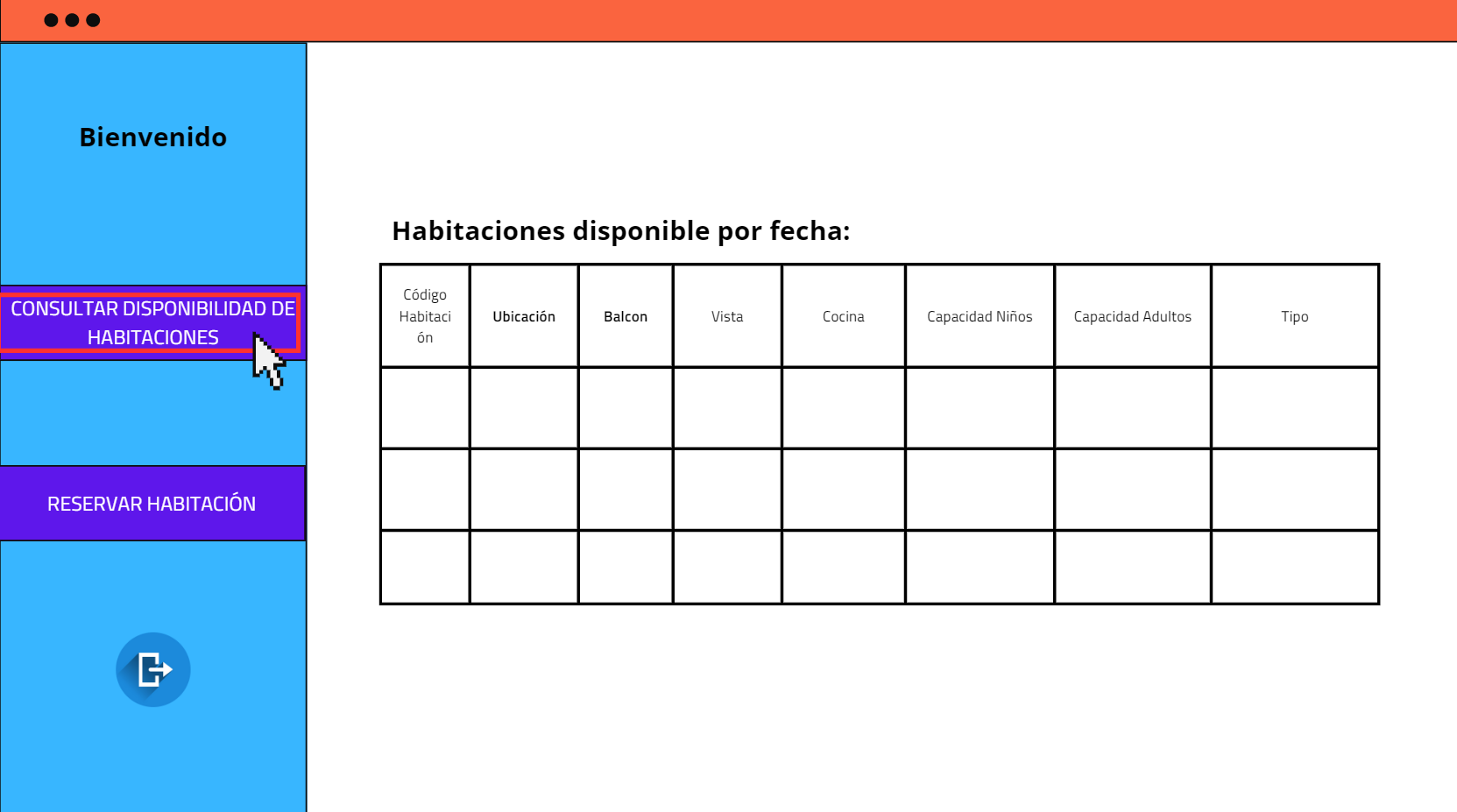
**Login de un usuario de un huésped:**



**Registro de un usuario de un huésped:**



**Primera opción, consultar disponibilidad de habitaciones:**

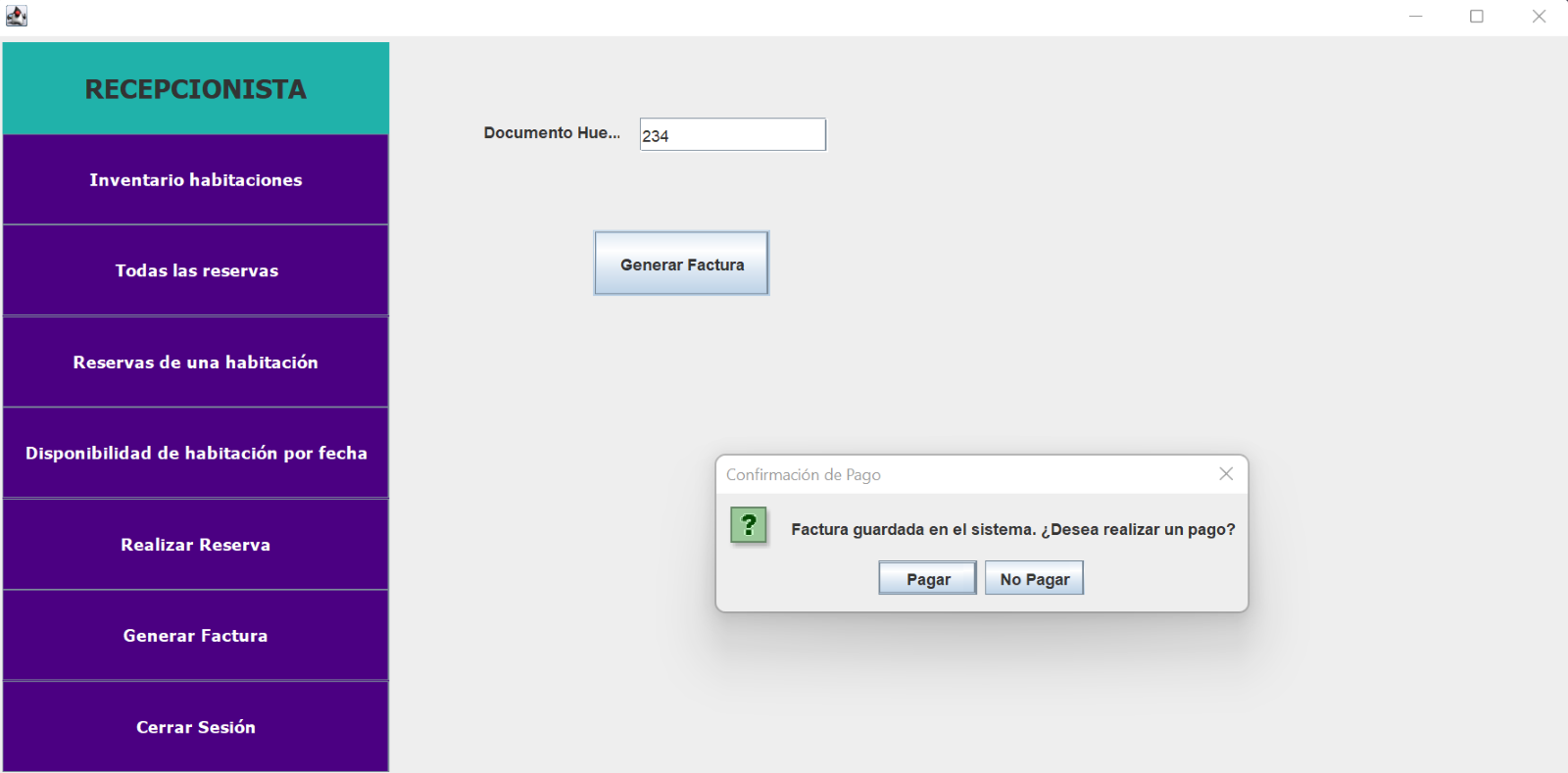


**Segunda opción, reserva de una habitación:**



**Diseño de la interfaz de la funcionalidad de pagos:**

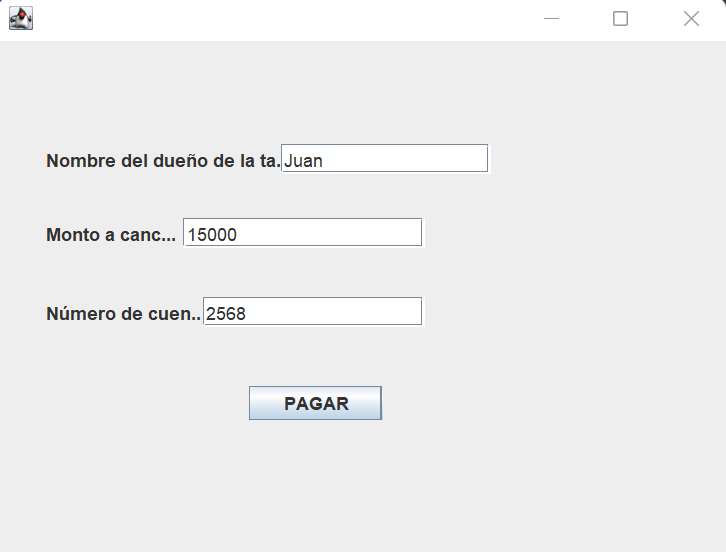
**Selección generar factura:**



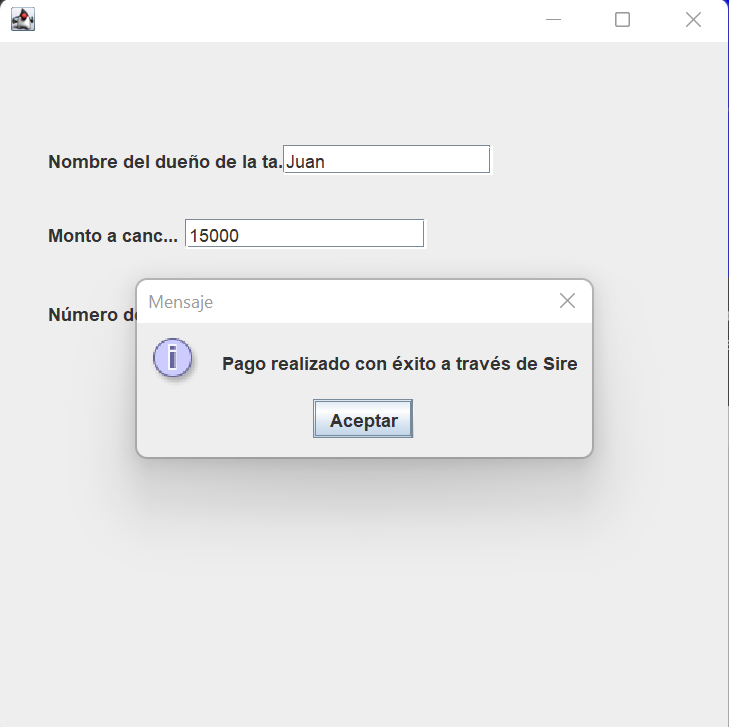
**Selección del método de pago:**



**Página para completar datos:**



**Diálogo con la confirmación del pago:**



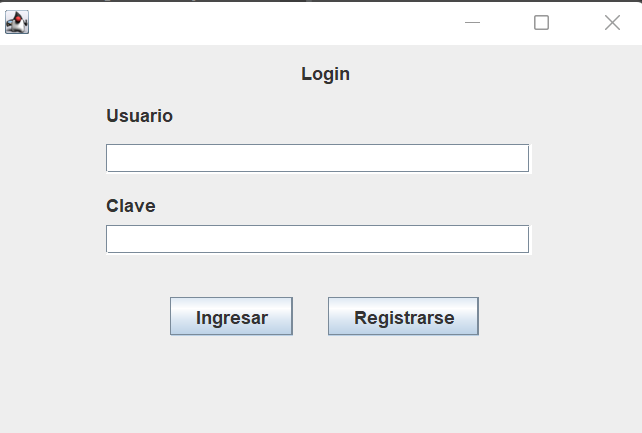
**Aplicación para empleados:**

**Interfaz gráfica**

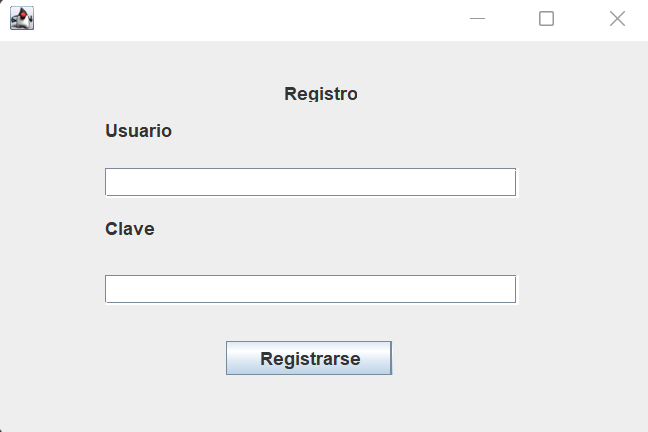
**Login:** En esta parte inicial, se da la opción de ingresar en caso de contar con un usuario ya registrado o, en caso contrario, se registrará el usuario.

**Información de interés:** Usuario: maur , Contraseña: 123. Este es un usuario para la

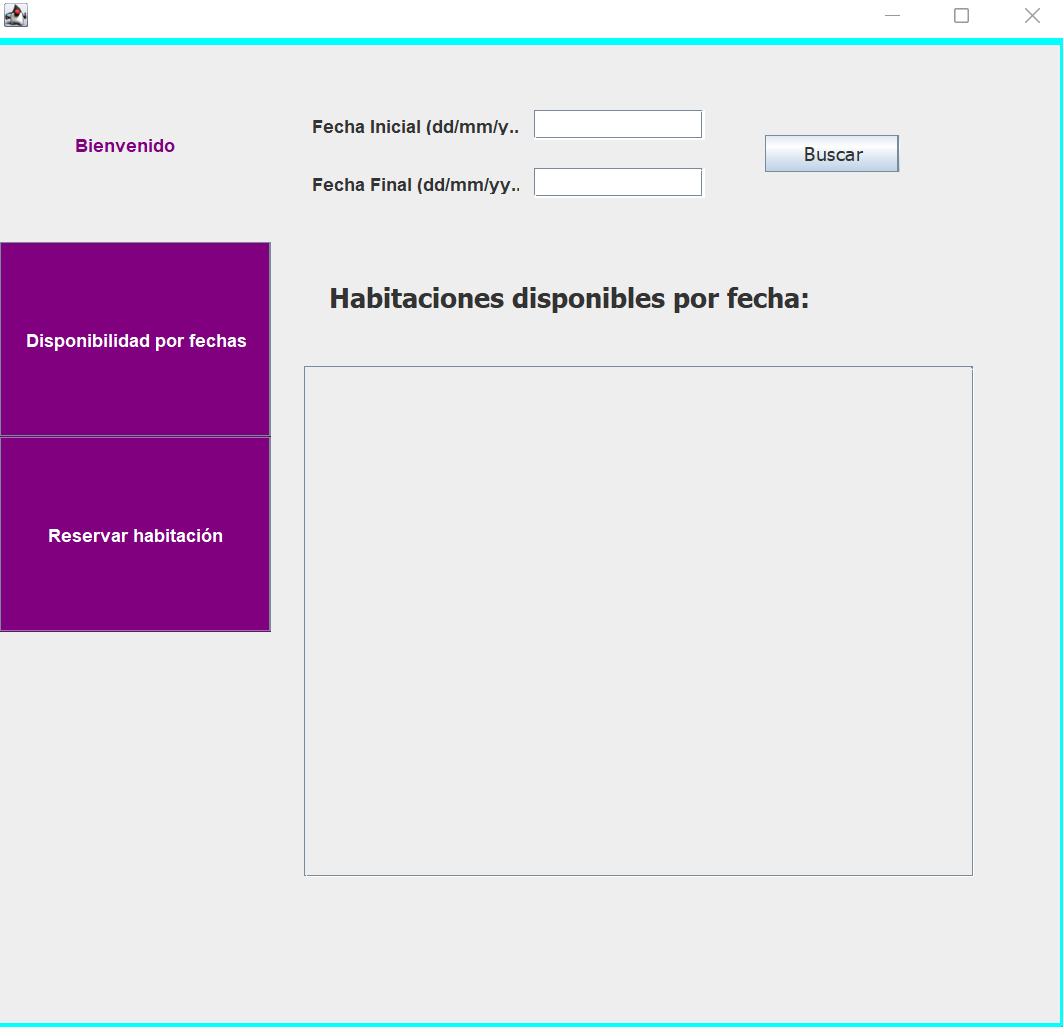
aplicaicón de huéspedes.



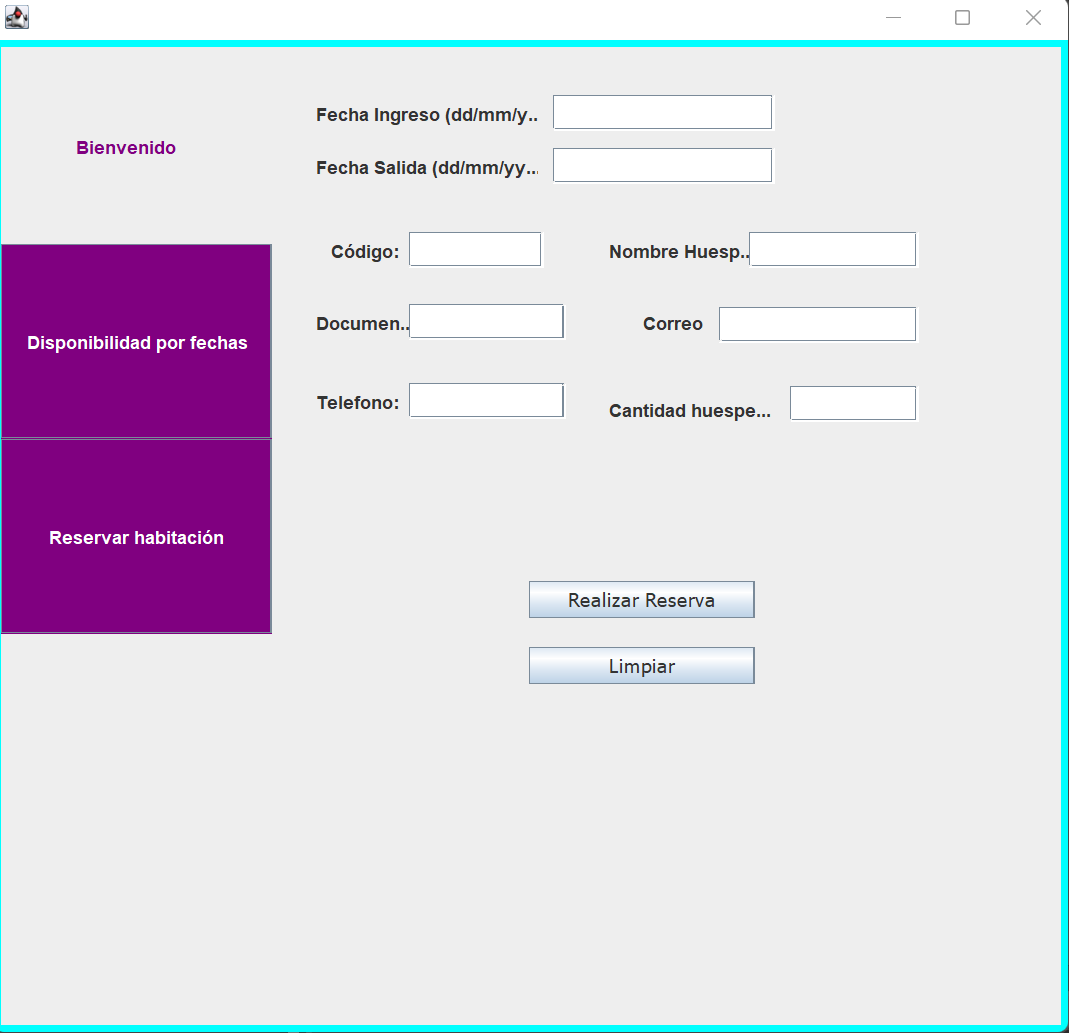
Registro: En esta opción, se podrá ingresar un usuario y contraseña para poder hacer uso de la aplicación de huéspedes.



**Igreso en la aplicación opción “Disponibilidad de por fechas”:** Con este botón, se podrá revisar la disponibilidad de habitaciones en el hotel indicando en los textFields la fecha de ingreso y fecha de salidas del hotel.



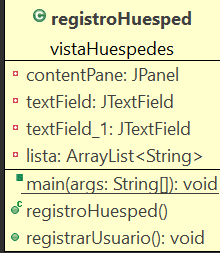
**Ingreso en la opción “Reservar habitación”:** Ahora, luego de revisar la disponibilidad de habitaciones con la anterior opción, el usuario deberá ingresar datos personales, fechas de entrada y salida, la cantidad de huéspedes y el código de la habitación que desea registrar para que se guarde en el sistema. Además, si la habitación no se encontraba entre las disponibles, el sistema arrojara una alerta de que la habitación no está disponible.



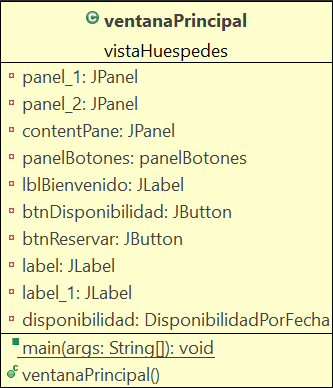
**Diagramas de clase con los cambios implementados:**

**Clases de la aplicación de huéspedes:**

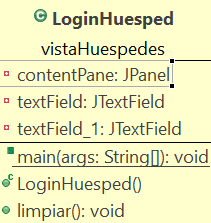
Encargada de registrar el registro de huéspedes, verificando creando el usuario en la plataforma.



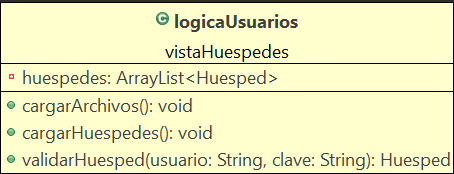
Ventana encargada de soportar los paneles que irán alternando con la selección de los botones. Además, incluye los botones con las funcionalidades disponibles para los huéspedes.



El login se encarga de verificar si el usuario ya existe en la plataforma, en caso contrariolanza alerta para advertir la no existencia del usuario.



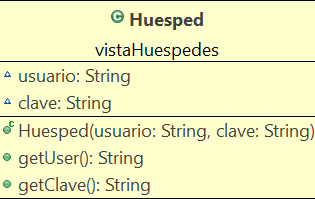
La lógica, es la clase que conecta la carga de Archivos y la carga de huéspedes con la lógica de la clase de los usuarios relacionados con el funcionamiento del hotel. De esta forma, se asegura que los datos estén en tiempo real de la misma plataforma.



Panel que presenta la distribución de los botones que se presentan en el panel de selección para los huéspedes.



Clase que se encarga de la creación de usuarios, con sus respectivos getters para clave y usuario.



**Diagrama de clases de la aplicación de huéspedes:**

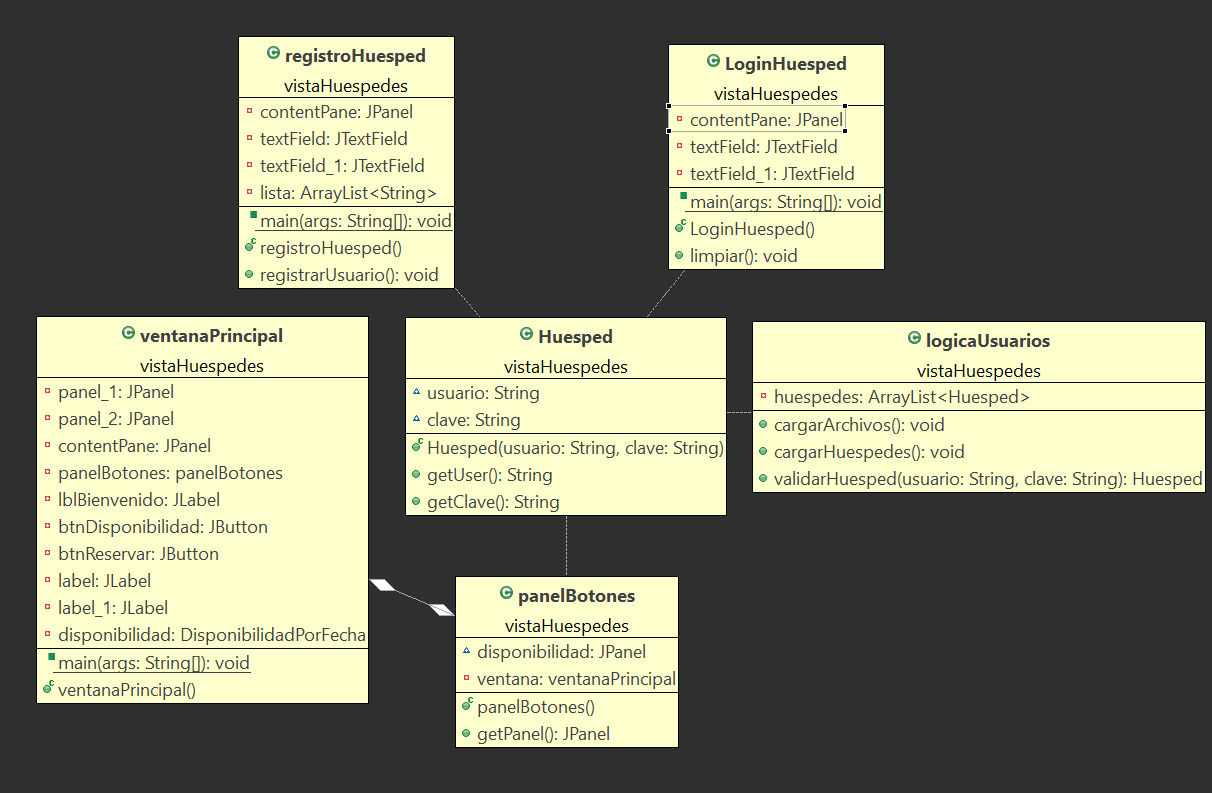
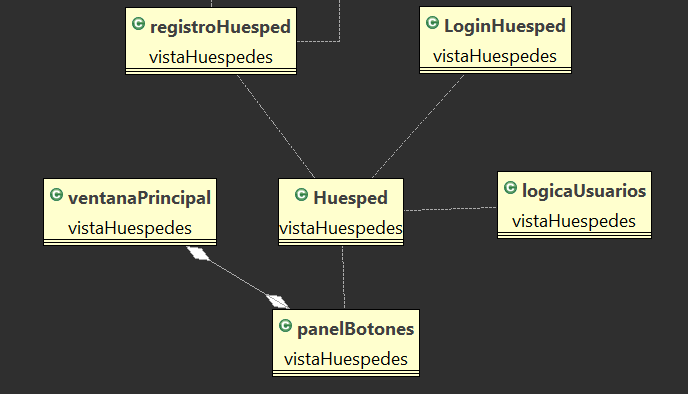


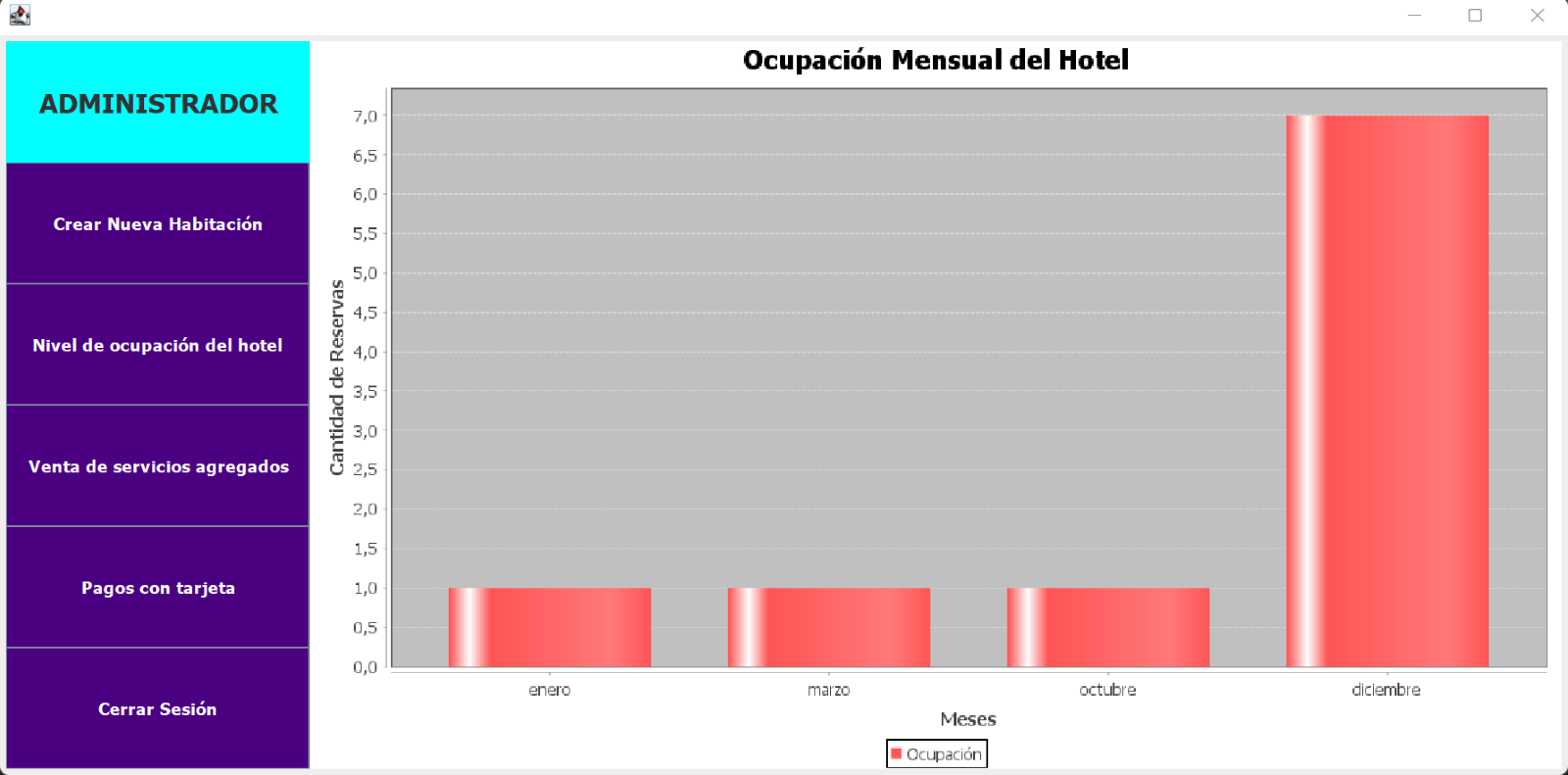
Diagrama de clases de alto nivel para la aplicación de huéspedes:



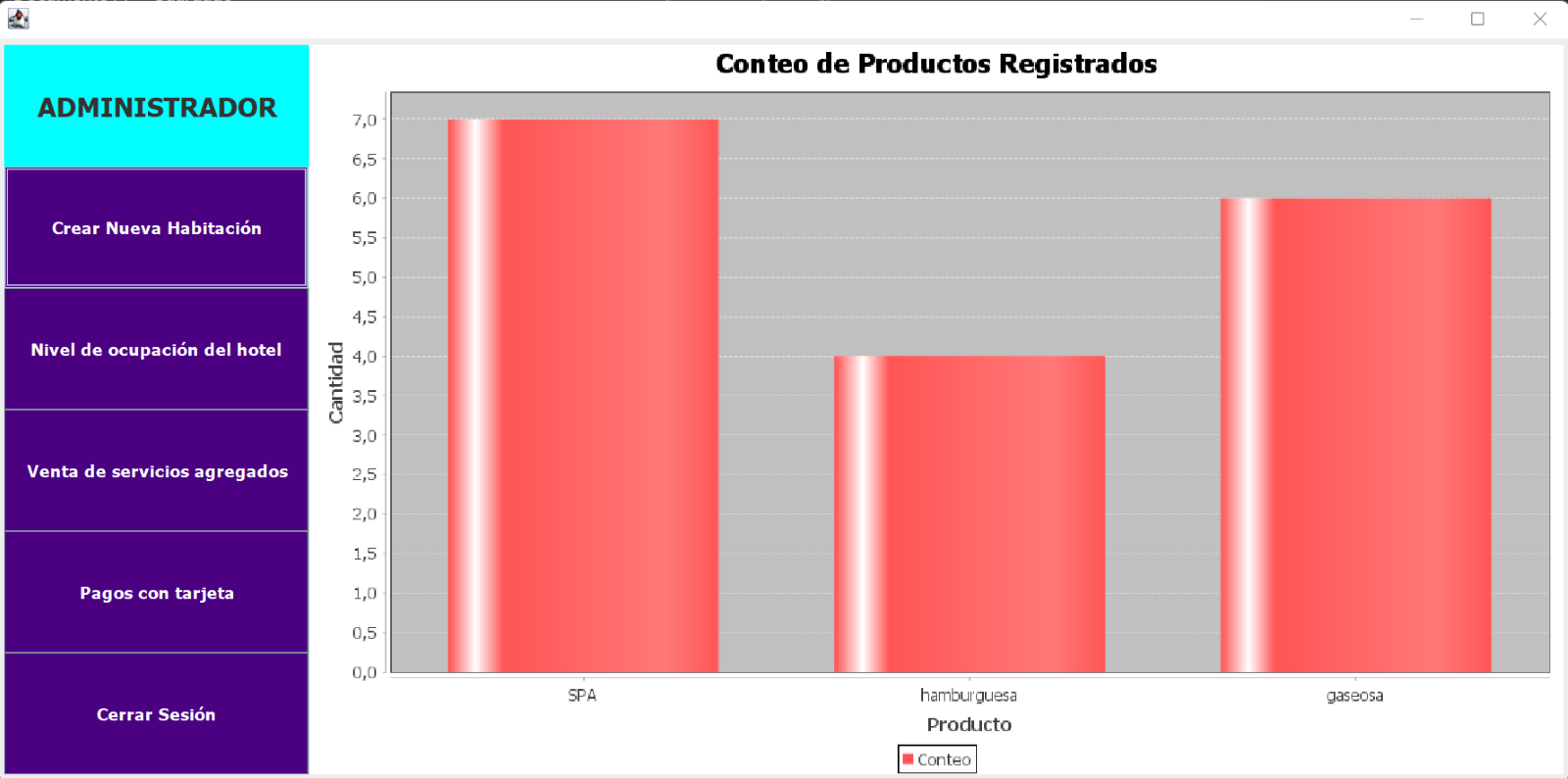
**Adición gráficos informativos menú administrador:**

1. **Gráfica con el nivel de ocupación del hotel para los diferentes meses:**

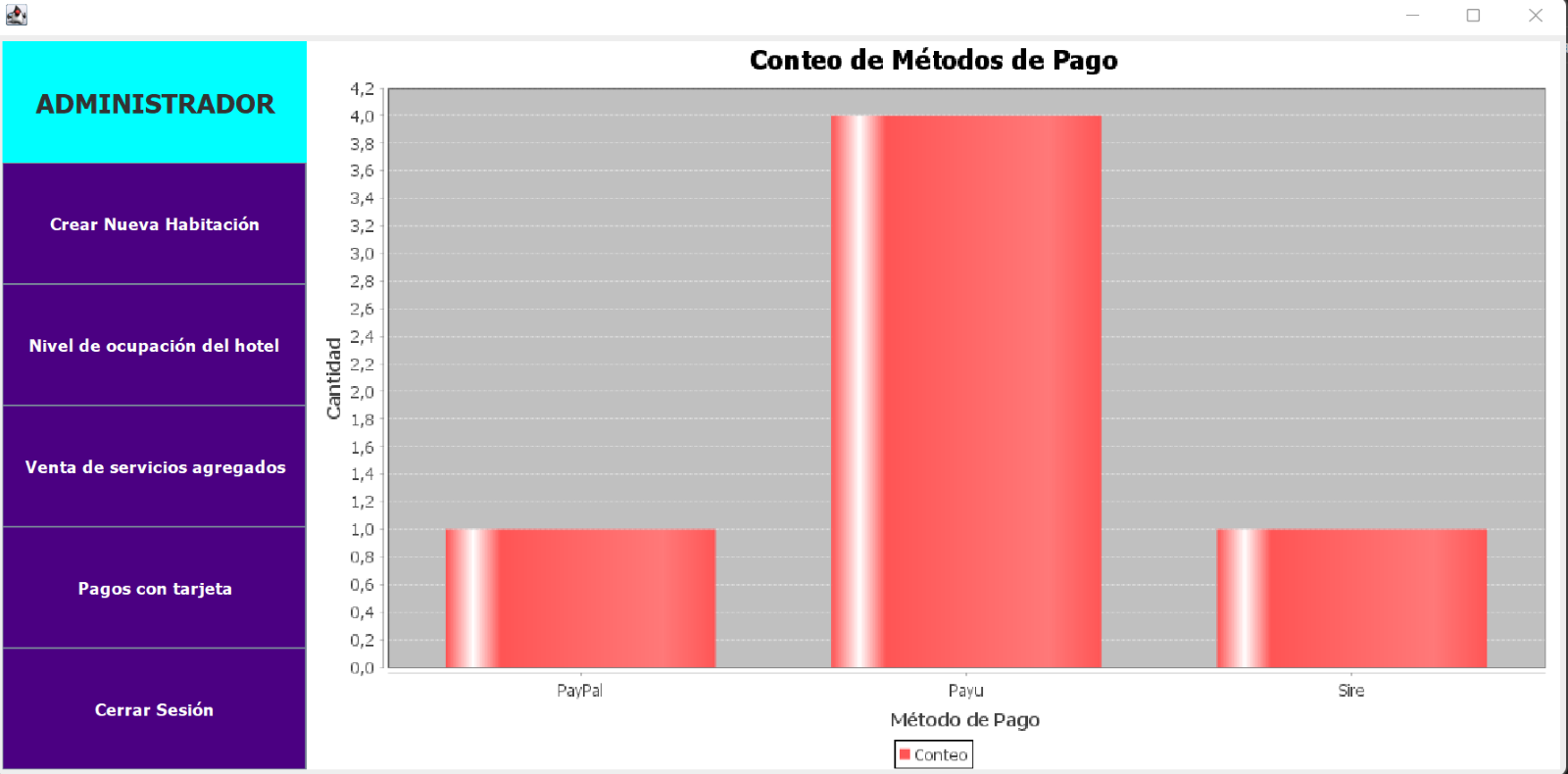
**Información de interés: Usuario: jose1, Contraseña:** 123



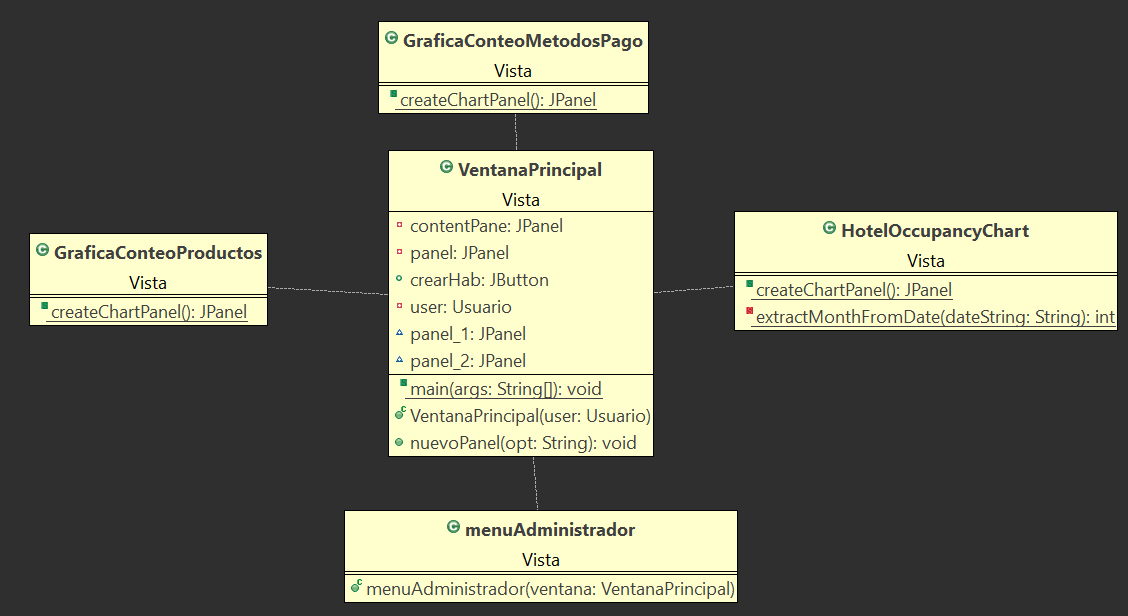
1. **Gráfica venta de servicios en el hotel:**



1. **Gráfica conteo de pagos con tarjeta:**



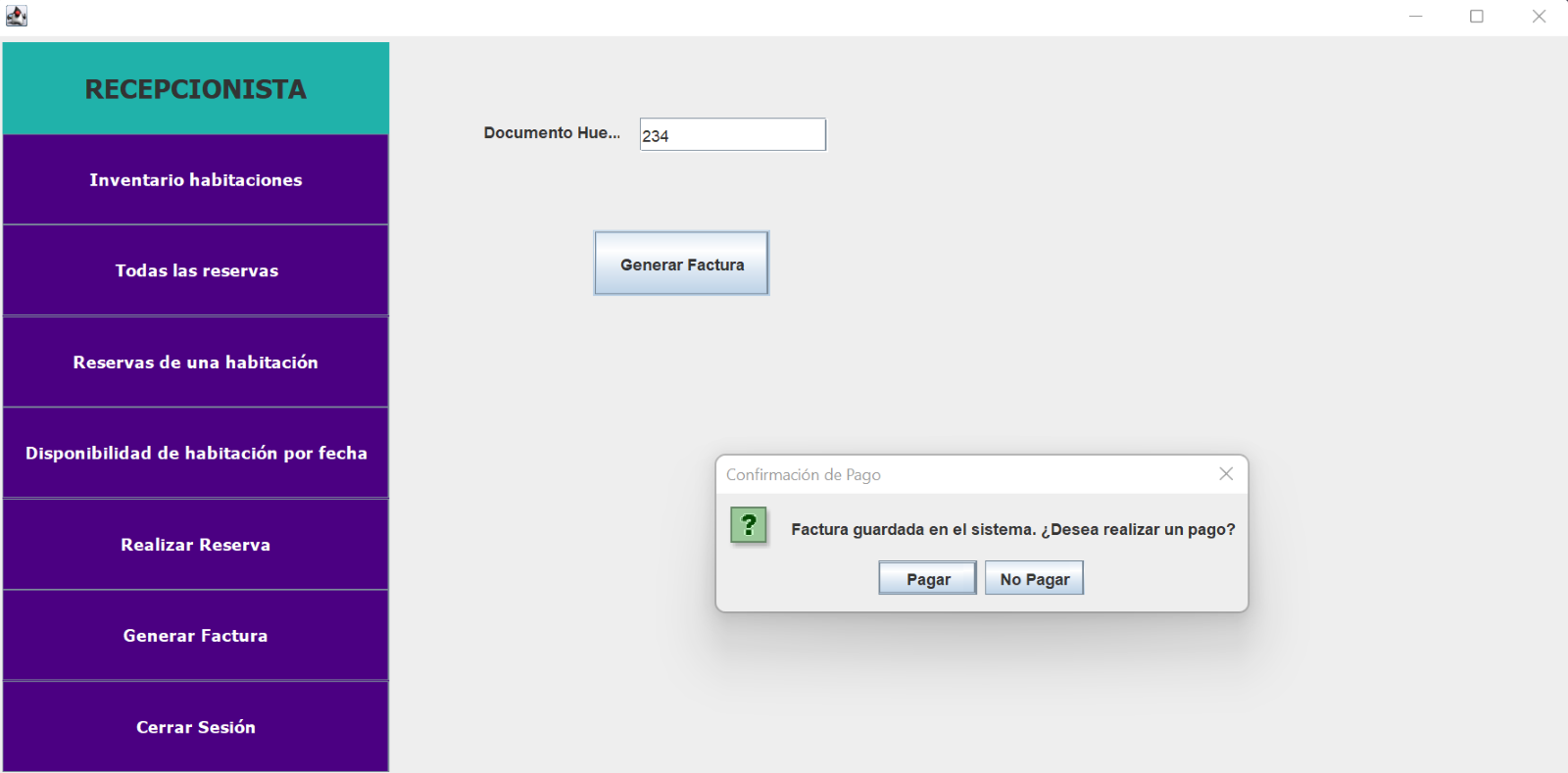
**Diagrama UML relación gráficos con el menú administrador:**



**Adición métodos de pago del hotel (menú recepcionista):**

**Información de interés: Usuario: maur, Contraseña: 123**

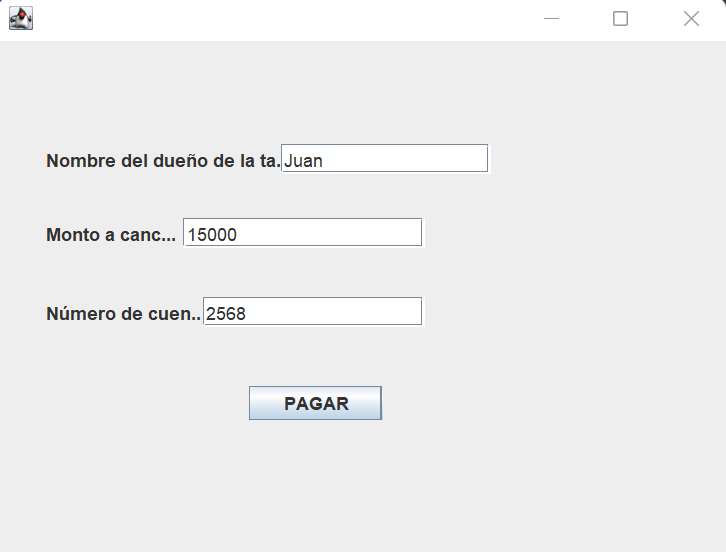
En primer lugar, se genera la factura, en este caso se usa como ejemplo el documento 234 asociado a uno de los huéspedes del hotel.



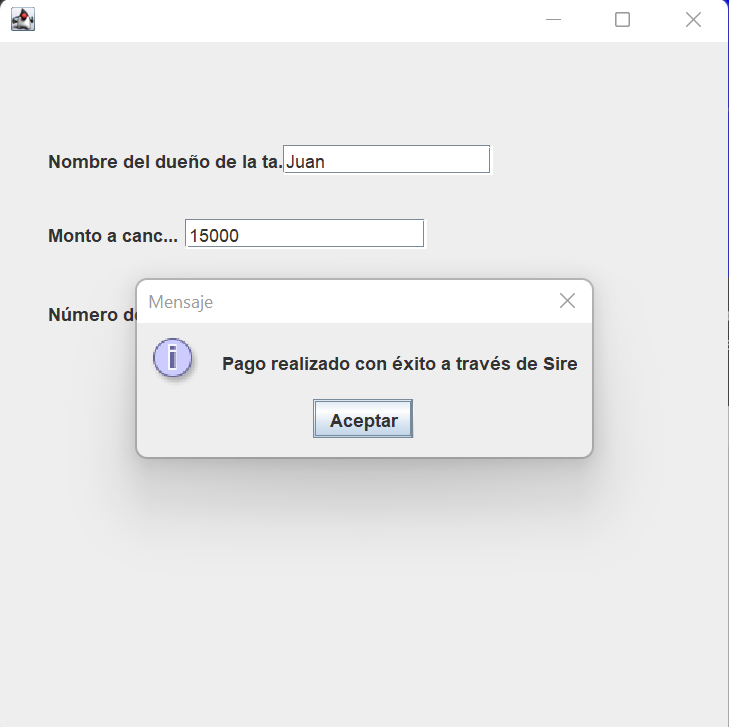
Luego, se selecciona la opción de pagar para que se despliegue el menú de los métodos de pago disponibles:



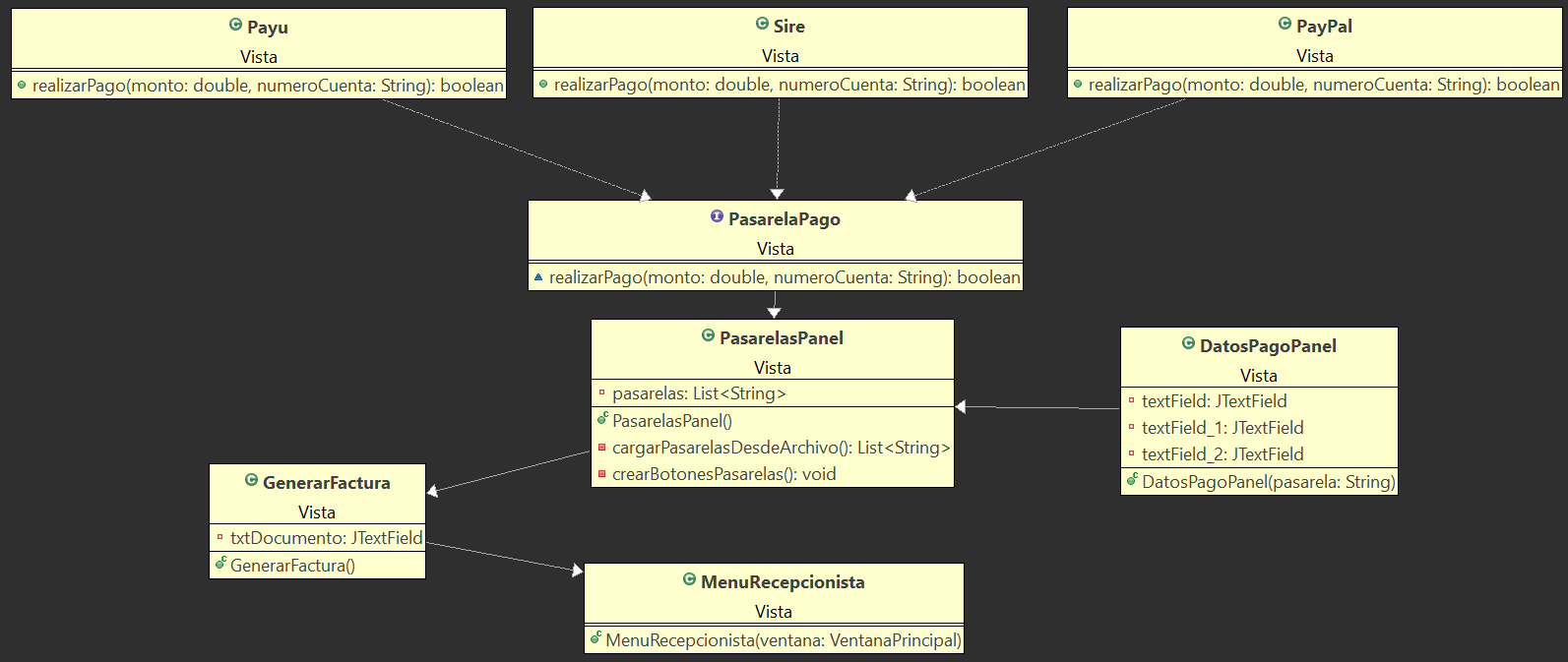
Después, al seleccionar el método de pago, en este caso Sire, se abre una página para completar los datos correspondientes al nombre del titular de la cuenta, la cantidad a cancelar y el número de la tarjeta con la que se realiza el pago. De esta forma, se guardará el pago en el sistema.



Finalmente, al seleccionar pagar, se informará si el pago fue realizado con éxito.



**Diagrama UML relación nueva implementación de métodos de pago:**



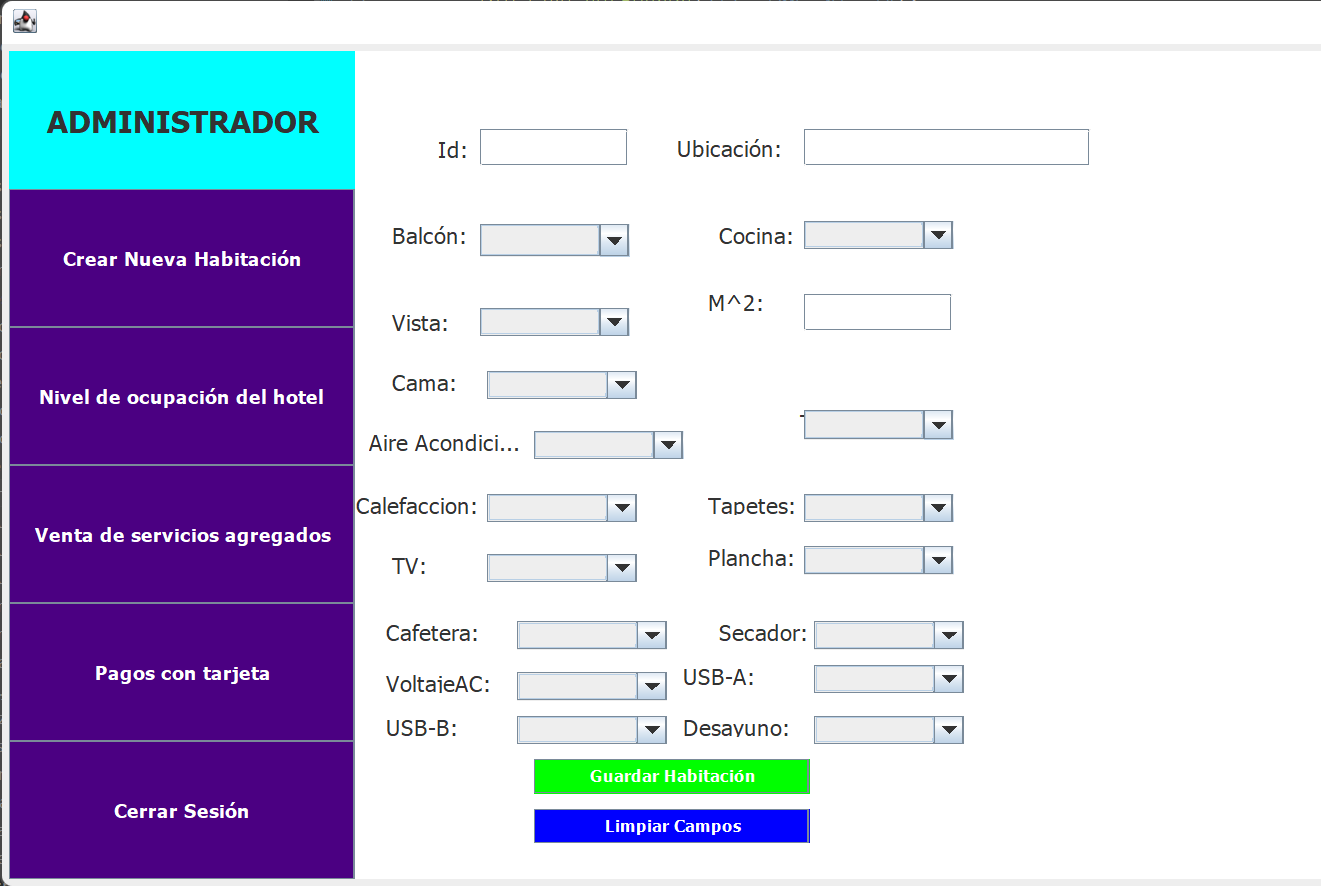
**Colaboración:** Los métodos de pago implementan su propio método de la interfaz PasarelaPago. Así pues, en primer lugar, el menú recepcionista usa el método de GenerarFactura. Luego, PasarelasPanel, que se basa en la selección de alguna de las PasarelasPago que permite la selección de cualquiera de los métodos presentes en el sistema. Así pues, PasarelasPanel, llama a DatosPagoPanel para completar la información relevante para realizar el pago que será el encargado de registrarlo en el sistema.

**Características del hotel en el menú recepcionista:**

Al revisar el inventario de las habitaciones, se podrán observar las nuevas características del hotel y se identifican los nuevos elementos que componen las habitaciones.



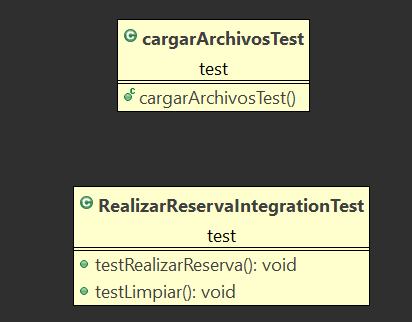
**Adición de elementos en la creación de una habitación desde el menú del administrador:**



**Pruebas unitarias:**

Con relación a las pruebas de integración y unitarias, CargarArchivosTest comprueba el idóneo funcionamiento de cada uno de los archivos que soportan el funcionamiento de la aplicación. De esta forma, en el ambiente se verifica lo que se considera pertinente. Adicionalmente, RealizarReservaIntegrationTest, verifica la integración de clases para realizar una reserva de forma correcta, así pues, en el ambiente se implementan los Asssert correspondientes para verifficar la creación y adicionamiento correspondientes en el proceso.

Las clases de las pruebas se ven de la siguiente manera, cada una incluye los métodos que desean probarse.



**Persistencia:** La persistencia en la aplicación se encarga de almacenar y recuperar la información relacionada con los pagos con tarjeta y los gráficos obtenidos. En el caso de los pagos con tarjeta, se utiliza un mecanismo de almacenamiento de datos que permita registrar los detalles de cada transacción, como el método de pago utilizado, el monto pagado y el cliente asociado. Esto garantiza que la información de los pagos esté disponible incluso después de reiniciar la aplicación.

Para lograr esto, se pueden utilizar tecnologías de persistencia de datos como una base de datos relacional o un sistema de archivos. La elección dependerá de los requisitos y preferencias del proyecto. Por ejemplo, se podría utilizar una base de datos SQL para almacenar los pagos con tarjeta, con tablas que representen las transacciones y sus atributos correspondientes. Cada vez que se realice un pago, se registraría en la base de datos para su posterior consulta y análisis.

En cuanto a los gráficos obtenidos, la persistencia puede implicar la capacidad de guardar y cargar los datos utilizados para generar los gráficos. Esto permite conservar los datos históricos y facilita la visualización y comparación de tendencias a lo largo del tiempo. Por ejemplo, se podría almacenar en un archivo o en una base de datos los valores utilizados para generar los gráficos, como las cantidades de pagos realizados con cada método de pago (PayPal, Sire, Payu). De esta manera, se pueden actualizar y recrear los gráficos en cualquier momento, incluso después de cerrar y reiniciar la aplicación.

Además, es importante diseñar la aplicación con la facilidad de extender nuevos elementos en la creación de una habitación. Esto implica utilizar una arquitectura flexible y modular que permita agregar fácilmente nuevos tipos de habitaciones con sus respectivas características y atributos. Esto puede lograrse mediante el uso de principios de programación orientada a objetos, como la herencia y la encapsulación, lo que facilita la adición de nuevas clases y la modificación de las existentes sin afectar el funcionamiento general de la aplicación.

En resumen, la persistencia en la aplicación garantiza el almacenamiento y recuperación de la información relacionada con los pagos con tarjeta y los gráficos obtenidos. Esto se logra mediante el uso de tecnologías de persistencia de datos adecuadas, como bases de datos o sistemas de archivos. Asimismo, es importante diseñar la aplicación con la capacidad de extender nuevos elementos en la creación de habitaciones, aprovechando los principios de programación orientada a objetos.

**Análisis de lo que salió bien y lo que salió mal:**

En el primer proyecto, nuestro diseño y lógica de la aplicación funcionaron de manera acorde a lo que teníamos planificado. Esto fue un aspecto positivo, ya que logramos comprender y satisfacer los requerimientos específicos de los diferentes roles dentro del hotel, como administrador, recepcionista y empleado. La razón detrás de este éxito radica en la buena planificación y análisis de los requisitos antes de comenzar la implementación.

Sin embargo, encontramos dificultades en la gestión de las reservas. Estos problemas podrían haber surgido debido a una falta de validación adecuada al momento de realizar una reserva, errores en la lógica de disponibilidad de habitaciones o posibles fallos en la interacción con la persistencia de datos. Reconocemos que estas dificultades podrían haberse evitado si hubiéramos realizado pruebas más exhaustivas y si hubiéramos tenido una comprensión más profunda de los requisitos relacionados con las reservas.

Pasando al segundo proyecto, inicialmente estábamos entusiasmados con el diseño propuesto, que implicaba sustituir JPanels en un mismo JFrame. Este enfoque nos permitió reducir la cantidad de ventanas emergentes y lograr una interfaz más cohesiva y fácil de entender. Durante la implementación, logramos cumplir con todos los requisitos establecidos y creamos una interfaz agradable y amigable para el usuario. Este éxito se debió a una planificación cuidadosa y un enfoque sólido en el diseño de la interfaz de usuario.

En cuanto al tercer proyecto, encontramos que los requerimientos adicionales fueron fáciles de comprender y no requirieron modificaciones importantes en el código existente. Esto fue positivo, ya que nuestra implementación anterior fue sólida y adaptable, lo que nos permitió extender fácilmente nuevas funcionalidades sobre lo que ya estaba hecho. Este aspecto demostró la modularidad y extensibilidad de nuestro código.

Sin embargo, enfrentamos dificultades durante la implementación de métodos de pago con carga dinámica y con la lógica de reserva. Estos desafíos surgieron principalmente debido a una comprensión insuficiente de los requisitos específicos y a una falta de planificación adecuada para abordar estos aspectos. Afortunadamente, pudimos resolver estos problemas después de un análisis detallado y una adaptación de nuestro enfoque. Estas dificultades nos enseñaron la importancia de comprender completamente los requisitos antes de comenzar la implementación y de dedicar suficiente tiempo a la planificación de cada aspecto del proyecto.

En resumen, nuestros proyectos tuvieron resultados positivos en términos de diseño, implementación y cumplimiento de requisitos específicos. Sin embargo, reconocemos que hubo desafíos relacionados con las reservas, los métodos de pago y la lógica de reserva. Estos problemas fueron resultado de una comprensión insuficiente de los requisitos, falta de pruebas exhaustivas y planificación insuficiente en áreas específicas. Aprendimos valiosas lecciones de estas dificultades y las utilizaremos como oportunidades de mejora para futuros proyectos.

**Análisis de las causas de los problemas:**

Nuestro equipo inicialmente constaba de tres miembros, pero antes de la entrega del proyecto 1, uno de los integrantes abandonó la materia. Esta situación generó desafíos adicionales, ya que tuvimos que redistribuir las responsabilidades y ajustar nuestro plan de trabajo para adaptarnos a la nueva situación. La falta de un miembro del equipo nos afectó en términos de carga de trabajo y colaboración, ya que cada uno de nosotros tuvo que asumir más responsabilidades individuales.

Uno de los problemas que enfrentamos fue trabajar con librerías sobre las que no teníamos un dominio completo. En particular, la librería JFreeChart nos llevó tiempo de análisis y comprensión, lo que retrasó nuestro progreso. Esta situación nos hizo conscientes de la importancia de investigar y familiarizarnos con las tecnologías y librerías antes de utilizarlas en nuestros proyectos, para evitar retrasos innecesarios.

Además, las diferencias de horario entre los miembros del equipo a veces nos obligaban a trabajar de forma individual. Esto dificultaba la comunicación y la colaboración en tiempo real, ya que cada uno tenía sus propias clases y responsabilidades. Aprendimos que la coordinación y la comunicación efectiva son fundamentales para mantener un flujo de trabajo eficiente, especialmente cuando los horarios no son sincronizados.

En algunas ocasiones, nos encontramos con la falta de especificaciones claras en el PDF del trabajo. Esto nos llevó a tener que hacer suposiciones o buscar aclaraciones adicionales, lo que consumía tiempo y generaba cierta confusión. Para futuros proyectos, nos aseguraremos de tener una comunicación clara y fluida con los profesores o tutores para obtener una comprensión completa de los requisitos desde el principio.

También enfrentamos dificultades para implementar algunos diseños de interfaz más complejos en Eclipse. En ocasiones, las herramientas y características de Eclipse no eran tan intuitivas como esperábamos, lo que nos llevó más tiempo del planificado para implementar ciertas funcionalidades de la interfaz. Este desafío nos hizo reflexionar sobre la importancia de evaluar la viabilidad técnica de los diseños propuestos y de tener un conocimiento sólido de las herramientas y entornos de desarrollo que utilizamos.

En conclusión, hemos realizado un análisis exhaustivo de las causas de los problemas que enfrentamos en nuestros proyectos. Hemos identificado áreas clave como la falta de colaboración de un miembro del equipo, la necesidad de comprender y dominar las tecnologías utilizadas, las diferencias de horario, la claridad en las especificaciones y las dificultades en la implementación de diseños. Estos problemas nos brindan una valiosa lección sobre la importancia de la planificación, la comunicación efectiva y el conocimiento técnico para evitar situaciones similares en proyectos futuros. Utilizaremos esta experiencia como una oportunidad de aprendizaje para mejorar nuestras habilidades y evitar los mismos problemas en el futuro.