**Proyecto 3 Documento de Diseño - Entrega final**

**Integrantes:**

Maicol Yojan Antonio Rincón - 202027329

Juan Pablo Peña Jaime – 202212756

1. **Contexto del problema**

Dentro del proyecto del curso vamos a construir un sistema que los hoteles llaman “Property Managament System” o PMS, el cual se encargará de administrar muchos de los elementos más importantes del hotel. El PMS se encargará de tres grandes funcionalidades. En primer lugar, el sistema tendrá un inventario de las habitaciones del hotel. En segundo lugar, el sistema tendrá el catálogo de servicios ofrecidos en el hotel y un registro de consumo. En tercer lugar, el sistema debe ser capaz de manejar todos los aspectos relacionados con las reservas, los huéspedes y los pagos.

**Inventario**

El sistema debe ser capaz de manejar un inventario de habitaciones 1 para el hotel que serán utilizados para las reservas. Cada habitación tendrá un identificador (número o nombre), una ubicación e información sobre su capacidad. El hotel tiene habitaciones tipo estándar, suite, y suite doble. Independientemente del tipo, algunas habitaciones tienen balcón, algunas tienen vista y otras tienen cocina integrada. Para las camas que aparecen en la descripción de cada habitación es importante saber el tamaño y la cantidad de personas que caben en ellas. Por su tamaño, algunas camas sólo pueden ser utilizadas por niños. El sistema debe permitir que el administrador del sistema cree las habitaciones una por una, o cargue un archivo con la información de nuevas habitaciones (o información actualizada de habitaciones existentes). Los empleados de la recepción del hotel podrán consultar el inventario y las características de las habitaciones, para saber qué características tienen, quién las está ocupando en un determinado momento, y quién las tiene reservadas en una determinada fecha. El administrador del sistema también puede cargar las tarifas por tipo de cuarto, las cuales serán válidas en determinadas fechas. En el momento de configurarlas, el administrador especificará el tipo de cuarto (ej. suites), un rango de fechas (ej. entre 10 de enero y 18 de marzo) y unos días de la semana (ej. viernes, sábado y domingo) para los cuales aplicará una tarifa. Si el administrador se equivoca y aplica más de una tarifa para el mismo tipo de habitación para la misma fecha, el sistema debe tomar la tarifa más económica. El sistema debería avisarle al administrador si hay alguna fecha dentro de los próximos 365 días en la que no exista una tarifa asignada para un cierto tipo de habitación. El administrador también puede cambiar las tarifas de los servicios que ofrezca el hotel, cargar los menús del restaurante y configurar todo lo que sea necesario de cada plato.

**Servicio**

El sistema debe ser capaz de registrar los servicios que hayan sido consumidos por un huésped, registrar los pagos que se hayan hecho, y generar las facturas correspondientes. La mayoría de los consumos se pueden pagar inmediatamente o se pueden registrar a una habitación para que el huésped sólo tenga que pagar en el momento de hacer el registro de salida (check-out) 2 . Los consumos serán registrados en el sistema por un empleado del hotel (ej. mesero, encargado del spa). El principal servicio que presta el hotel es el de alojamiento: cualquier huésped del hotel debe estar alojado en una habitación, y cada habitación generará un cobro (se paga por habitación, no por persona). Los huéspedes del hotel pueden utilizar otros servicios que ofrezca el hotel, como por ejemplo el uso del spa (que se cobra por persona) o el servicio de guía turístico (que se cobra por grupo). Los servicios de restaurante que presta el hotel pueden ser tomados en el comedor o en la habitación, pero no todas las opciones del menú pueden llevarse a la habitación. El sistema debe tener información completa sobre el menú, incluyendo los nombres de los platos y bebidas, los precios, y los rangos de horas en los que están disponibles.

**Reservas, registro y facturación**

El sistema debe permitir que un empleado de la recepción del hotel realice una reserva a nombre de un huésped, y le informe la tarifa total para las noches seleccionadas. Las reservas también pueden cancelarse, pero no en las últimas 48 horas. Para hacer una reserva se necesita el nombre del huésped, documento, correo electrónico, número de celular y la cantidad total de personas que se esperan. Al hacer una reserva, las habitaciones necesarias quedarán bloqueadas en las fechas de la reserva. En el momento de hacer el registro, el empleado de la recepción le pedirá al huésped los datos básicos de él y de sus acompañantes para registrarlos en el sistema. Un huésped y sus acompañantes se consideran un grupo y pueden tener una o varias habitaciones asignadas. Un grupo siempre debe registrarse y hacer el registro de salida al tiempo, después de haber pagado por todos sus consumos y por la estadía. El límite de personas por habitación depende de la capacidad de las camas, pero los menores de dos años no necesariamente necesitan cama –es decisión de los padres- aunque siempre deben quedar registrados dentro de los huéspedes. La recepción podrá generar las facturas para todos los consumos realizados durante la estadía de un grupo en el hotel.

**Figura 1. Contexto del problema.**

**Nivel 1**

**1.1. Componentes candidatos y estereotipos**

1. La aplicación es utilizada por tres tipos de personas: Administrador, Empleados y Recepcionistas. Por ende, debe existir un componente para cada uno, ambos serán de tipo coordinador dado que serán los encargados de asignarles tareas a otros componentes. Además, independientemente del tipo, estos se agruparán en una clase **Usuario.**
2. Otro candidato puede ser **Tarifa** que va a ser la encargada de guardar la información del precio de una habitación, esta tarifa depende de la información proporcionada por el Administrador, por ende, debe ser un Information holder el cual se encargar de cargar y entregar información de las Tarifas dadas por el administrador.
3. En la aplicación debe existir un componente para **Hotel**, el cual es importante para mantener las relaciones con las demás clases, se le asignara el estereotipo de Structurer.
4. Un hotel debe tener habitaciones, están habitaciones se encargan de tomar decisiones importantes y controlar las acciones de otros componentes por lo tanto el estereotipo será de tipo Controlador.
5. En el contexto del problema podemos identificar 3 habitaciones: Suite, Suite doble y Estándar, también importantes dentro de la aplicación. Sin embargo, como ya tenemos un candidato que es **Habitación** que abarca los 3 tipos de habitaciones, entonces la información sobre cada uno será proporcionada por esta clase.
6. Dentro de los atributos de una habitación encontramos camas. Sin embargo, dado que cama debe tener un tamaño y una cantidad de personas que pueden dormir en ella decidimos tomarla como otro candidato y al igual que los tipos de habitación su función será la de proporcionarle información a la habitación por tanto también será un Information holder. De esta manera, la clase encargada será **Cama**.
7. Otro candidato importante es **Reserva** el cual debe proporcionar información acerca del estado de reserva de una habitación, y también debe ser capaz de realizar la reserva por lo tanto será de tipo Controlador.
8. Otro posible controlador puede ser **Servicio**, el cual es el encargado de generar la factura, registrar los servicios y registrar los pagos del huésped. De esta manera, se le asigna el tipo Information Holder.
9. Otro componente candidato será **Huesped,** su función es de Information Holder, puesto que, es el encargado de recopilar la información sobre el huésped y la cantidad de personas que vienen con este. De esta manera, se deriva la clase **Acompañante,** la cual registrará (en caso de que los haya), la información de los acompañantes.
10. Ahora, nos encontramos con la clase **Factura,** la cual recopilará el monto total de los servicios consumidos. Por tal motivo, será un Information Holder.

**2.2. Responsabilidades**

**Habitación**

**-** Conocer información sobre la habitación

**Administrador**

- Cargar tarifas

- Crear o cargar habitaciones

- Cargar menú del restaurante

**Empleado**

- Consultar el menú y lo demás servicios

- Registrar un servicio a nombre de un huésped

**Recepcionista**

**-** Consultar el inventario de las habitaciones

- Consultar las reservas

- Consultar el estado de reserva de una habitación específica

- Consultar la disponibilidad de las habitaciones dad una fecha

- Realizar una reserva

- Generar factura

**Servicio**

- Registrar pagos que se hayan hecho

**Tarifa**

- Generar factura habitación

**Reserva**

**-** Conocer estado de las habitaciones (Disponibilidad)

**-** Conocer tarifa de la reserva

- Cancelar una reserva

**Factura**

- Guardar la factura

- Mostrar la factura con su monto total

**Usuario**

- Verificar la existencia de un usuario en el sistema

**Cama**

**-** Brindar información sobre la cama

**Huesped**

- Brindar información del huésped

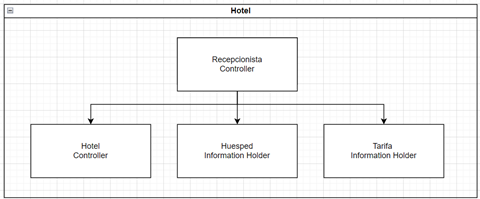
**Acompañante**

- Brindar información sobre los acompañantes (en caso de que los haya).

**2.3. Colaboraciones**

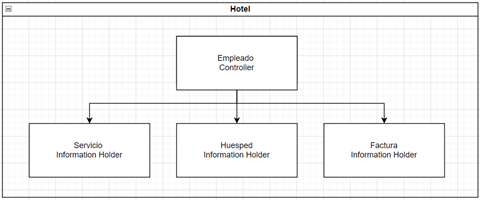
Hacer una reserva**:** Implica que el Recepcionista, al recibir una reserva:

1. Le indica a hotel que le muestre las habitaciones disponibles para la fecha de la reserva, con sus respectivas características.
2. Toma los datos referentes a la información sobre el huésped.
3. Le indica a tarifa que proporcione el precio por la habitación que se va a reservar.



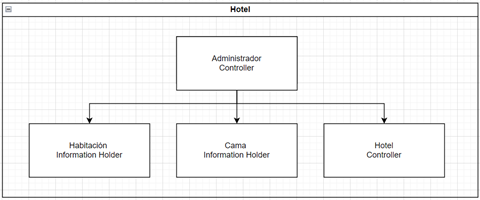
Registrar un servicio: Implica que el Empleado, al registrar un servicio:

1. Obtenga información del tipo de servicio y precio que ofrece el hotel.
2. Digite el documento del huésped al que se le va a registrar el servicio.
3. Añada el servicio a la respectiva factura del huésped.



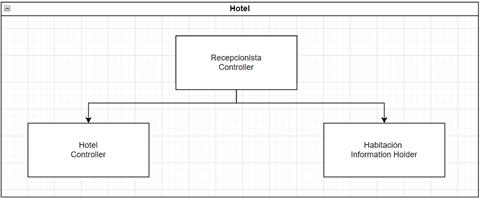
Crear o cargar una habitación al sistema implica:

1. Conocer la información de la habitación.
2. Conocer la información acerca de las camas que la componen.
3. Añadirla al inventario del hotel.



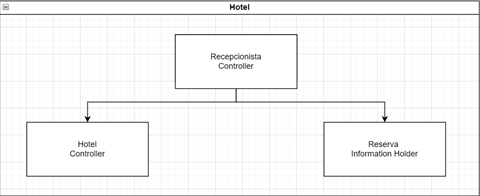
Conocer las características de una habitación: Implica que el Recepcionista al ejecutar esta opción:

1. Le informe a hotel el número de habitación en cuestión, para que sea retornada.
2. Habitación le informe las características generales de la habitación.



Conocer si una habitación está reservada:

1. Le informe a hotel el número de habitación en cuestión, para que sea retornada.
2. Con el número de habitación, Reserva puede verificar el estado de la habitación.



**Nivel 2**

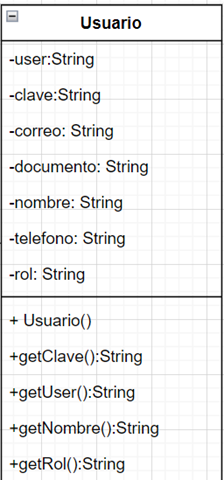
**3.1 Habitación**

**-** Conocer información sobre la habitación

Se le atribuye el grado de clase, debido a que es fundamental para recopilar información importante, como lo son las características de las habitaciones. Además, es poseedora de varios métodos fundamentales dentro de las responsabilidades de los usuarios.

**Usuario**

Clase encargada de validar la información bien sea de un Administrador, Recepcionista o Empleado.

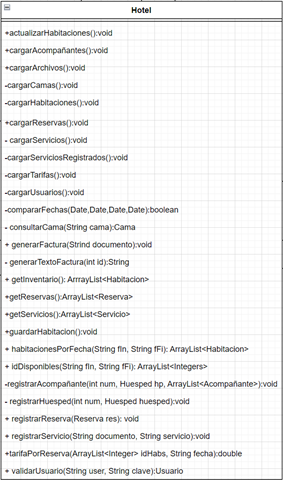


U,{9741eb02-7bea-4894-925f-08b4dca4b112}{210},3.125,3.125

Para cada uno de los anteriores se encuentran las siguientes funcionalidades, las cuales nos dirigen a la clase **Hotel.**

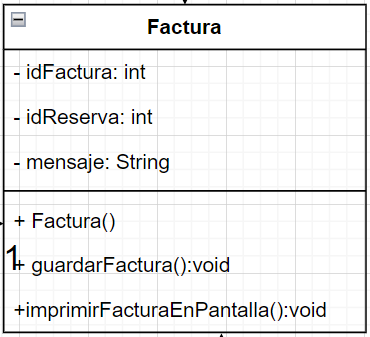
**Hotel**

Esta clase, podría considerarse en cierta manera como el “corazón del sistema”, pues de aquí nacen varias funciones claves para el correcto estructuramiento de las funciones del sistema. Así mismo, posee gran cantidad de funciones usadas por los tres tipos posibles de usuario. De esta forma, el Administrador invoca aquellas relacionadas con la carga o creación de datos. Por otra parte, el recepcionista usa aquellas referentes a las consultas de inventario, reservas y facturas. Finalmente, el Empleado invoca aquellas encargadas de gestionar el funcionamiento de los diversos servicios y su inclusión en la factura.



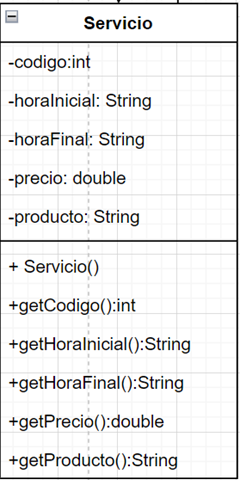
**Factura**

Su importancia se encuentra en el papel que cumple a lo largo de la ejecución del programa, pues en ella se adicionan los diversos servicios con sus respectivos montos. Significando, de esta forma que cumplirá un factor importante desde el principio hasta el final.



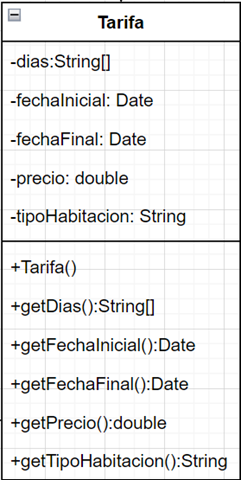
**Servicio**

Ligado, a la clase anterior, servicio identificará las funcionalidades que brinda el restaurante para las huéspedes, almacenando toda la información relativa a estos, siendo así, crucial para registrar las actividades que realice un huésped o grupo de huéspedes.



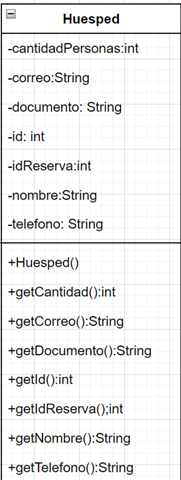
**Tarifa**

Se le establece como clase, por su capacidad de asignar un monto dado un tipo específico de habitación. Por tal motivo, colaborará en gran medida con muchas otras clases.



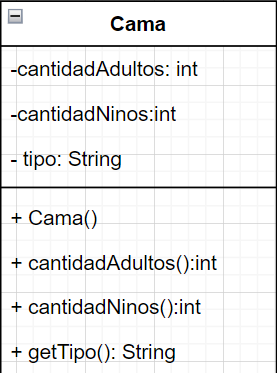
**Huesped**

Esta clase se encargará del manejo apropiado de la información de los huéspedes del hotel. De esta forma, estará en constante uso en medio de las demás funcionalidades del programa.



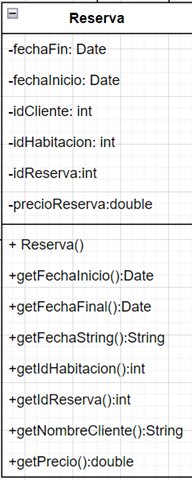
**Cama**

La clase cama, se encarga de complementar la asociación entre el cuarto y sus componentes, pues brinda información sobre este mueble específico.



**Reserva**

Esta clase tendrá por función registrar las diversas reservas que realicen los huéspedes, asimismo, obtendrá y gestionará toda la información pertinente.



**3.2 Responsabilidades**

La aplicación cuenta con tres grandes responsabilidades:

1. Gestionar un inventario de habitaciones (clase Habitacion) de hotel con información detallada sobre su capacidad, ubicación, y características.

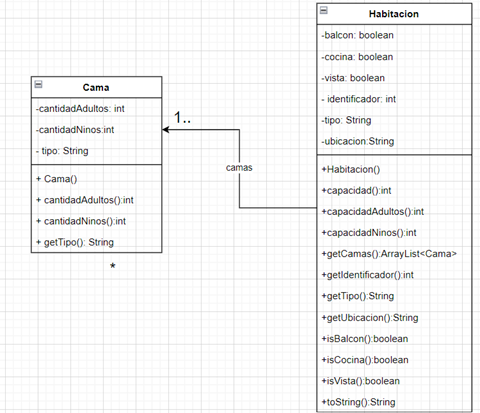
2. Llevar un registro de consumo de servicios (clase Servicio) ofrecidos en el hotel. Así mismo, debe mostrar las respectivas tarifas (clase Tarifa) y menús. Además, debe permitir el manejo de reservas, huéspedes y pagos. En cuanto al inventario de habitaciones, el sistema permite la creación y actualización de habitaciones, y el manejo de tarifas por tipo de cuarto y fechas. En cuanto al registro de consumo de servicios, el sistema puede registrar los servicios consumidos y generar facturas correspondientes.

3. El empleado se encarga de que se hagan reservas a nombre de huéspedes, se registren huéspedes y sus acompañantes, y se lleve a cabo el registro de salida (check-out).

**3.3 Colaboraciones**

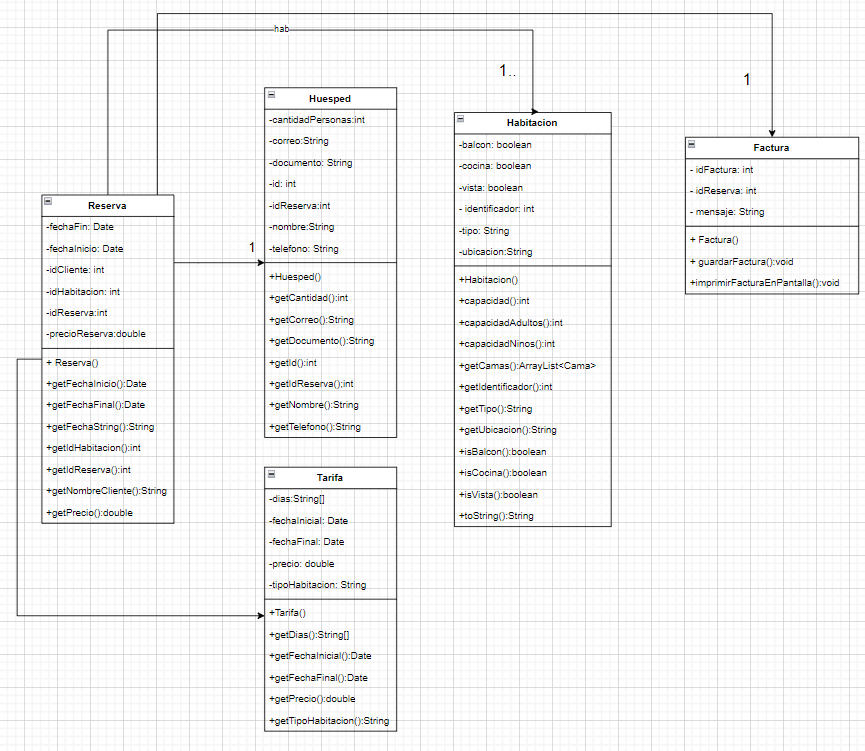
**Habitación:**

La creación de una habitación implica la existencia de un número de camas (una o más) que se encuentran dentro de la misma.



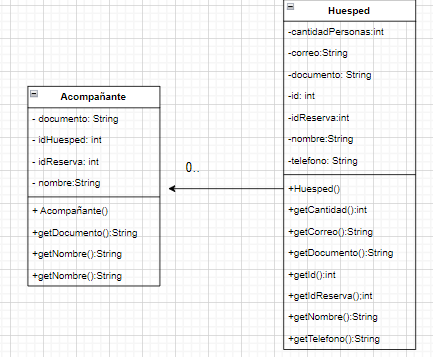
**Reserva:**

La creación de una reserva se compone de la inclusión de una habitación y un huesped asociados a la misma. De esta manera, también es posible establecer una conexión directa con “Tarifa” y “Factura”, pues esta informará el monto total de la reserva, para luego anexarla en la factura.



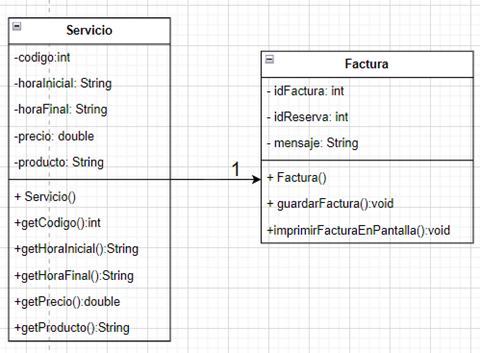
**Huesped:**

Un huesped puede estar compuesto de 0 o más acompañantes. Por tal motivo, en el momento en el que el huesped venga acompañado, se establecerá conexión con a clase acompañante, encargada de registrar la información de estos.



**Servicio:**

Un servicio implica su inclusión en una factura, pues de esta forma es posible llevar el registro detallado de los mismos.

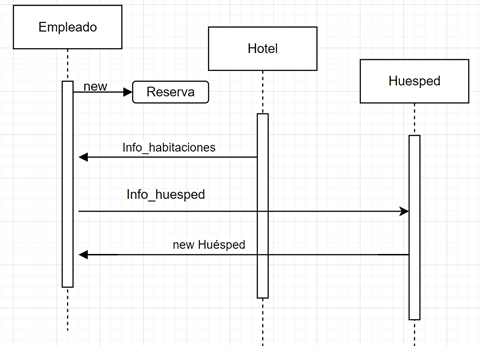


**Hotel:**

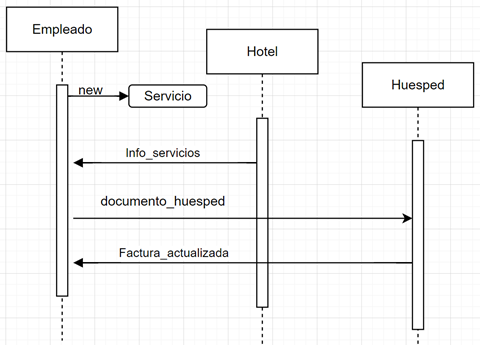
Como componente central del programa, hotel posee conexión directa con la mayoría de las clases, pues de esta manera podrá controlar el correcto funcionamiento de estas. Así pues, esta clase posee facultades sobre casi todas las clases, bien sea en función de obtener información o alterarla, según convenga. Adicionalmente, dentro de esta clase, es posible evidenciar el funcionamiento de un usuario de manera implícita, ya que, la mayoría de los métodos están pensados para ser ejecutados por un Empleado, un Administrador o un Recepcionista.

**Diagramas de secuencia**

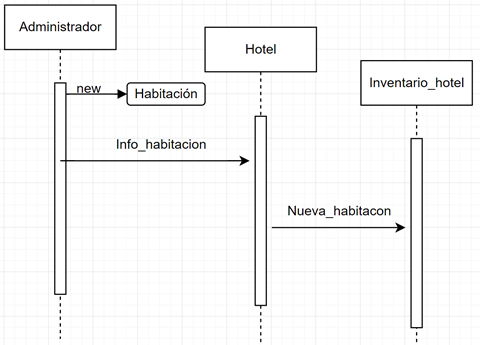
**Generar una nueva reserva.**



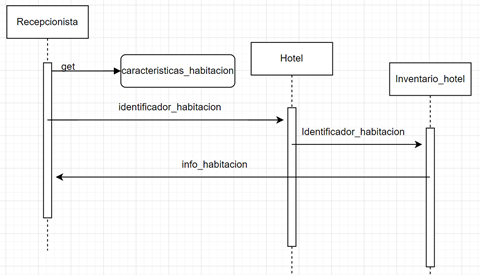
**Añadir un servicio a Factura.**



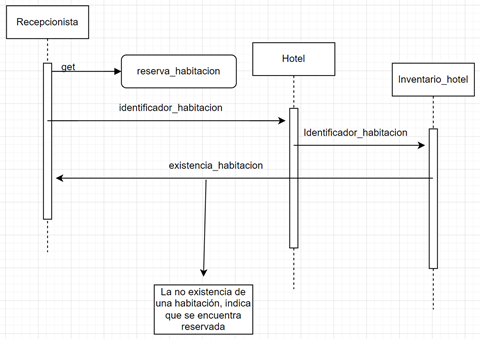
**Creación de una nueva habitación.**



**Obtener información sobre una habitación.**



**Obtener información acerca del estado de reserva de una habitación.**



**Persistencia a lo largo del tiempo:**

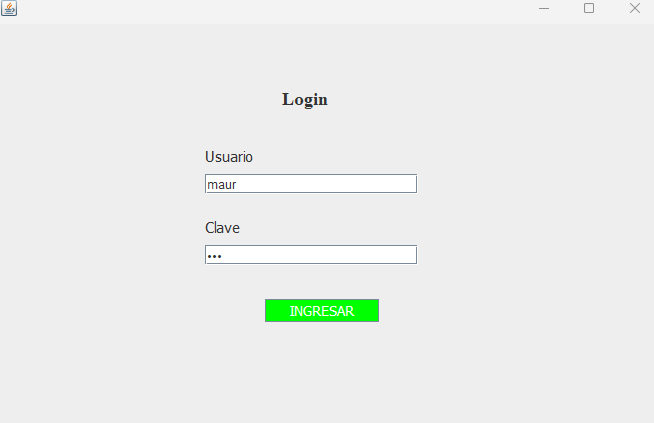
El programa será persistente en la medida en que, si bien pueden incluirse cambios en algunos funcionamientos claves del hotel, como podrían ser: alteraciones referentes a los servicios, reestructuraciones en los atributos de una habitación u otros. Para gran cantidad de casos que no exijan un cambio absoluto, podría acoplarse sin muchos inconvenientes. Lo anterior, se debe principalmente a su capacidad de “escalar”, pues admite nuevas adaptaciones, las cuales pueden ser aceptadas dentro de la naturaleza general planteada en el diseño de los objetos. Adicionalmente, al brindar la posibilidad de añadir o eliminar información dentro del sistema, asegurará que la memoria de este, siempre se encontrará en constante movimiento y soportará gran cantidad de operaciones dentro de las capacidades que el mismo hotel vaya a permitir como instancia física. Finalmente, su implementación se encuentra pensada para que, aunque algunas políticas del hotel cambien y, en algunos casos puedan llegar a exigir una edición del código del programa, no sea necesario llevar cambios a gran escala, pues cada clase posee cierta autonomía, logrando de esta forma que el sistema no dependa de un único centro. Así pues, los cambios en cuestión podrían llegar a ser sorprendentemente simples, pues en muchos casos podría bastar con cambiar una línea de código.

**Diagrama de alto nivel**

1. **Diseño final**

De esta forma, luego de haber definido las relaciones y haber identificado las colaboraciones pertinentes en el nivel 1 con su respectiva descomposición clase a clase en el nivel 2, se presenta el diseño final, previo a la implementación de la solución.

**Interfaz Gráfica**



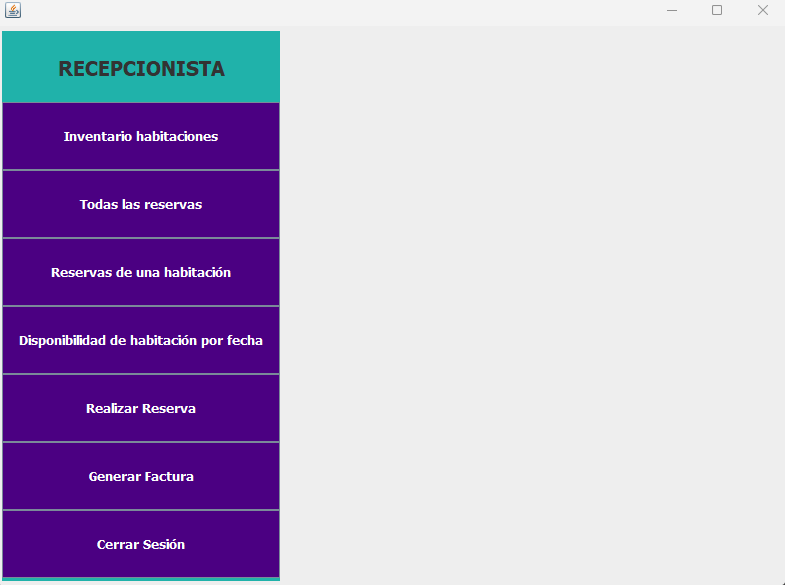
**Se debe ingresar siempre desde el login, esto para evitar que alguien ingrese desde la ventana principal saltándose el login**

El programa inicia con una ventana de Login, en la que el usuario debe colocar su usuario y contraseña, según el rol del usuario se desplegara otra ventana con diferentes opciones. Se tienen 3 Roles: Empleado, Recepcionista y Administrador. Adjunto el usuario y la clave de cada uno para que puedan ingresar y verificar las diferentes opciones de cada uno.

RECEPCIONISTA  
 Usuario: maur  
 Clave: 123

EMPLEADO  
 Usuario: luca  
 Clave: 123

ADMINISTRADOR  
 Usuario: jose1  
 Clave: 123



Este es la ventana que se le muestra al recepcionista, en la que puede escoger diferentes funciones haciendo click los botones del panel lateral.

Imagen que contiene Interfaz de usuario gráfica

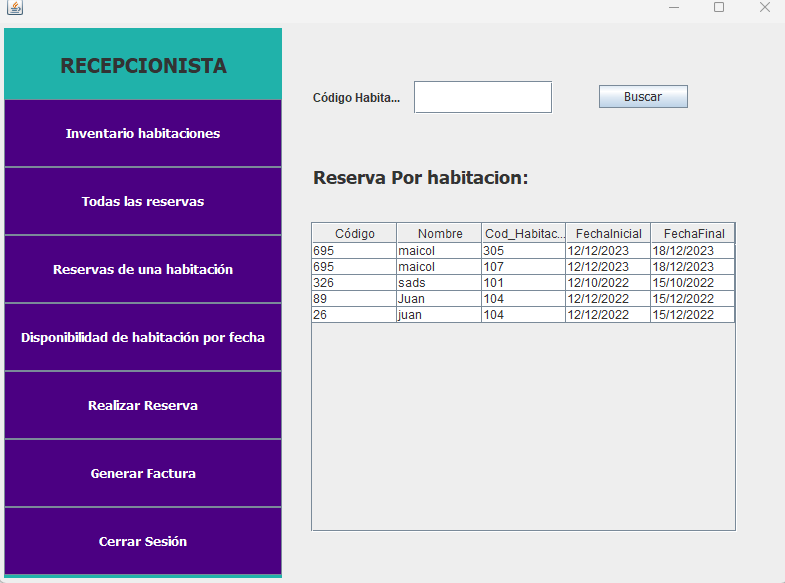
Descripción generada automáticamente

La primera Opción “Inventario Habitaciones” le muestra a detalle todas las habitaciones y sus datos en un JTable.

Tabla

Descripción generada automáticamente con confianza media

La siguiente opción “Todas las reservas” le muestra al recepcionista un detalle de todas las reservas realizadas en un JTable.

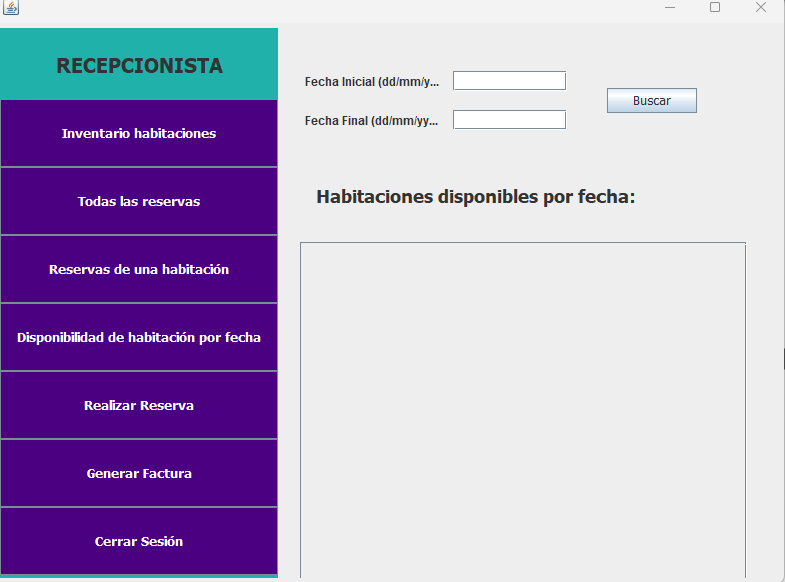


La siguiente opción “Reservas de una habitación” de principio muestra la misma información que el anterior JPanel, sin embargo, este cuenta con la opción de realizar la búsqueda para una habitación en específico, bastan con colocar el cogido de la habitación y darle clic en el botón buscar.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Así se ve como la aplicación después de realizar la búsqueda.



La siguiente opción “Disponibilidad de habitación por fecha” se muestra de esa forma, en la parte de arriba el recepcionista puede colocar la fecha inicial y final de la reserva y consultar las habitaciones que se encuentran disponibles.

Interfaz de usuario gráfica

Descripción generada automáticamente

Así vería el recepcionista las habitaciones disponibles en esa fecha.

**Interfaz de usuario gráfica

Descripción generada automáticamente**

La siguiente opción “Realizar Reserva” permite realizar una reserva, el recepcionista debe ingresar la información que pide el sistema y darle clic en el botón realizar reserva, adicionalmente agregamos un botón que permite limpiar todos los campos en caso de que el recepcionista requiera borrar los datos de manera rápida.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

La siguiente opción de “Generar Factura” permite generar la factura de una reserva, la cual queda almacenada dentro de una carpeta de la aplicación en “Facturas”.

Y un botón de cerrar sesión que cierra la aplicación.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

La anterior ventana es la que se muestra cuando el empleado ingresa al sistema.

Tabla

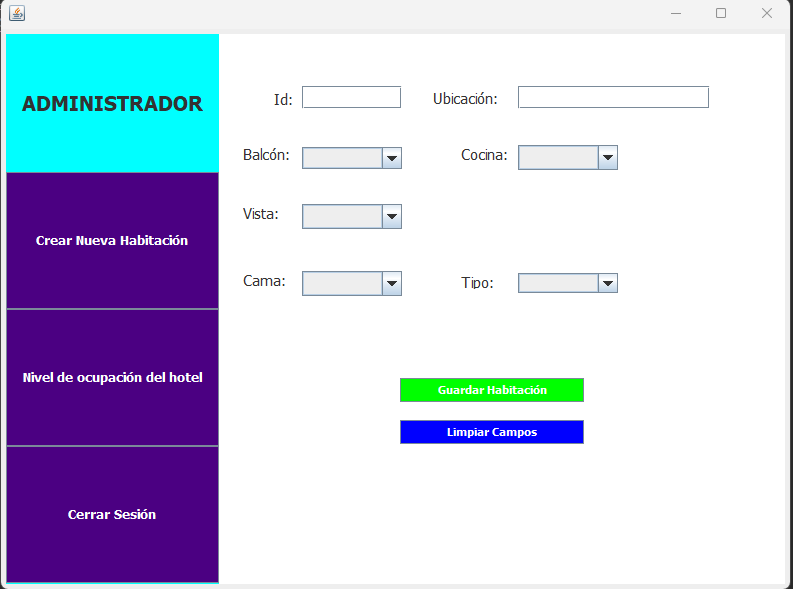
Descripción generada automáticamente

La opción “Consultar menú y servicios” le muestra al empleado todos los servicios y productos que ofrece el hotel.

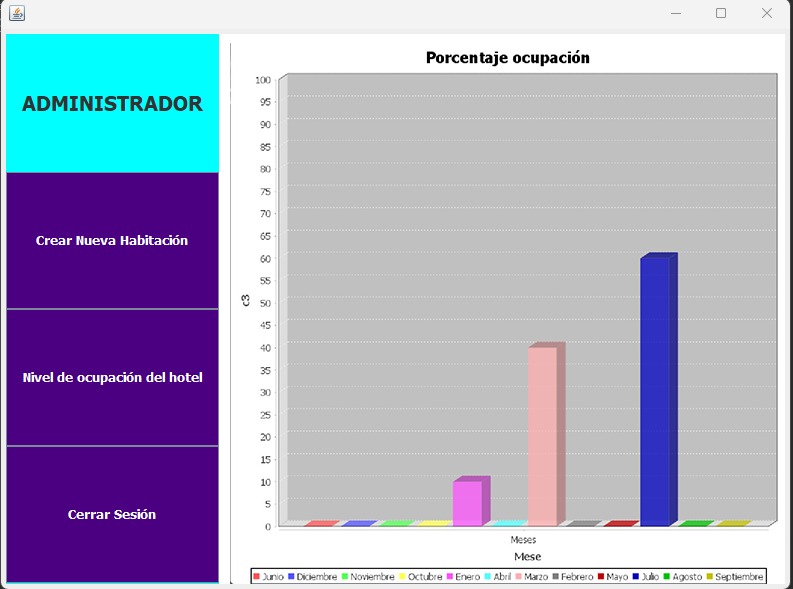
**Interfaz de usuario gráfica

Descripción generada automáticamente**

La opción de “Registrar Servicio” le permite registrar un servicio al empleado, este puede hacer clic en la tabla y el código se llenará automáticamente o si lo prefiere puede escribirlo de manera manual y posteriormente darle a registrar servicio.

****

Para el menú administrador, puede crear una nueva habitación y consultar el nivel de ocupación del hotel, al igual que cerrar Sesión como los otros usuarios.



**Diagramas de clases interfaz**

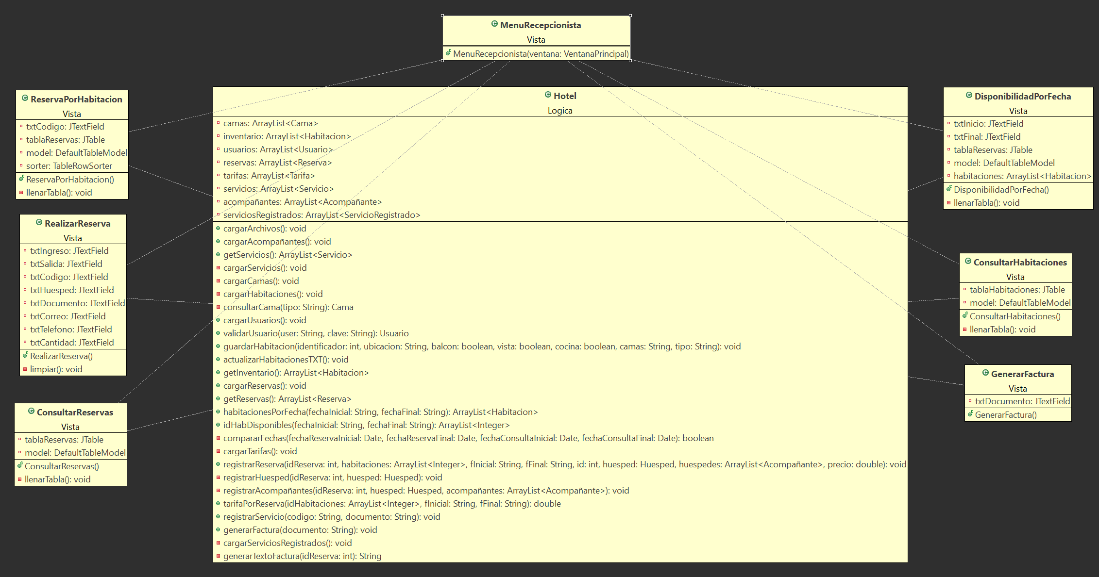
**Menú administrador**



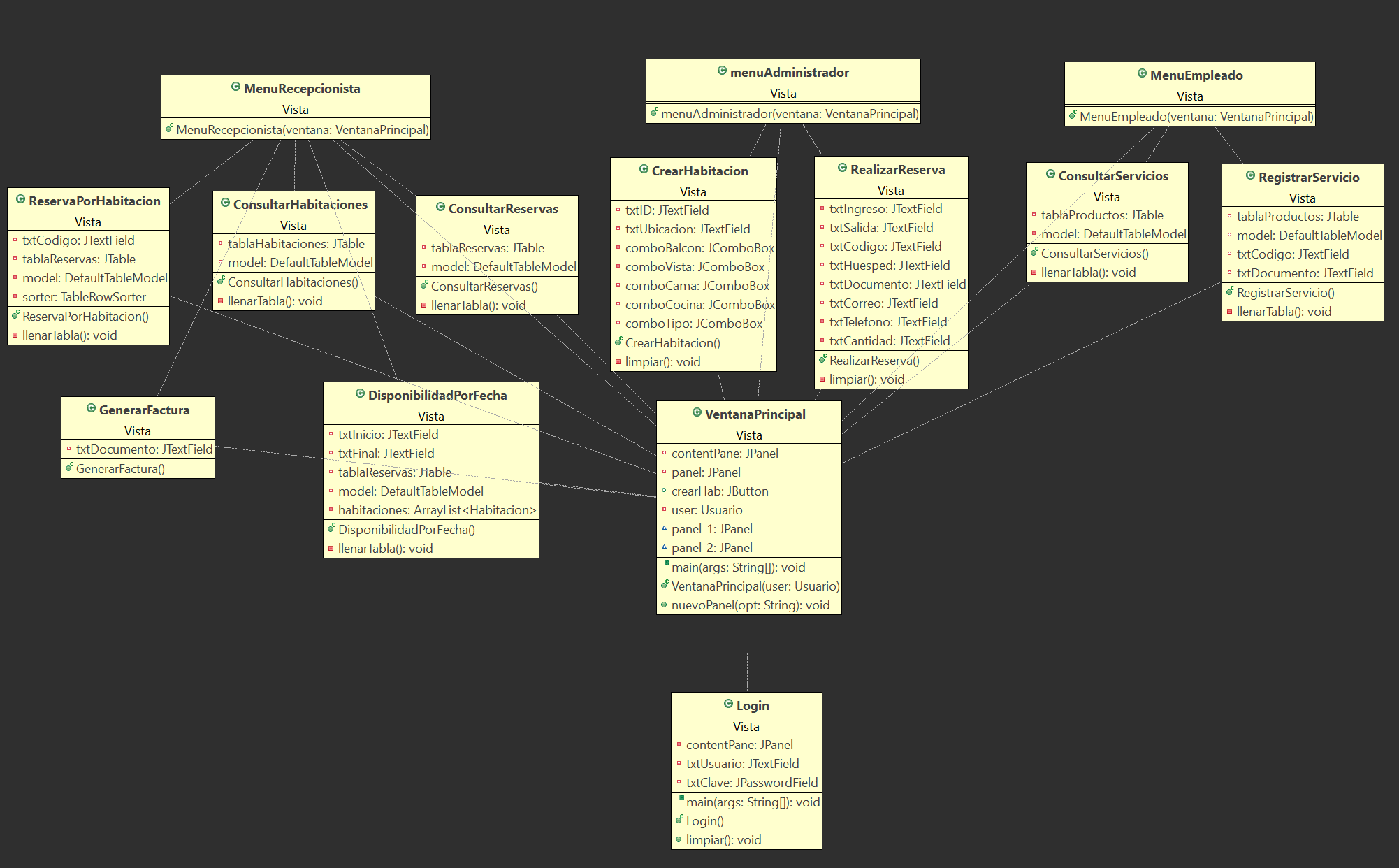
**Menú Empleado**



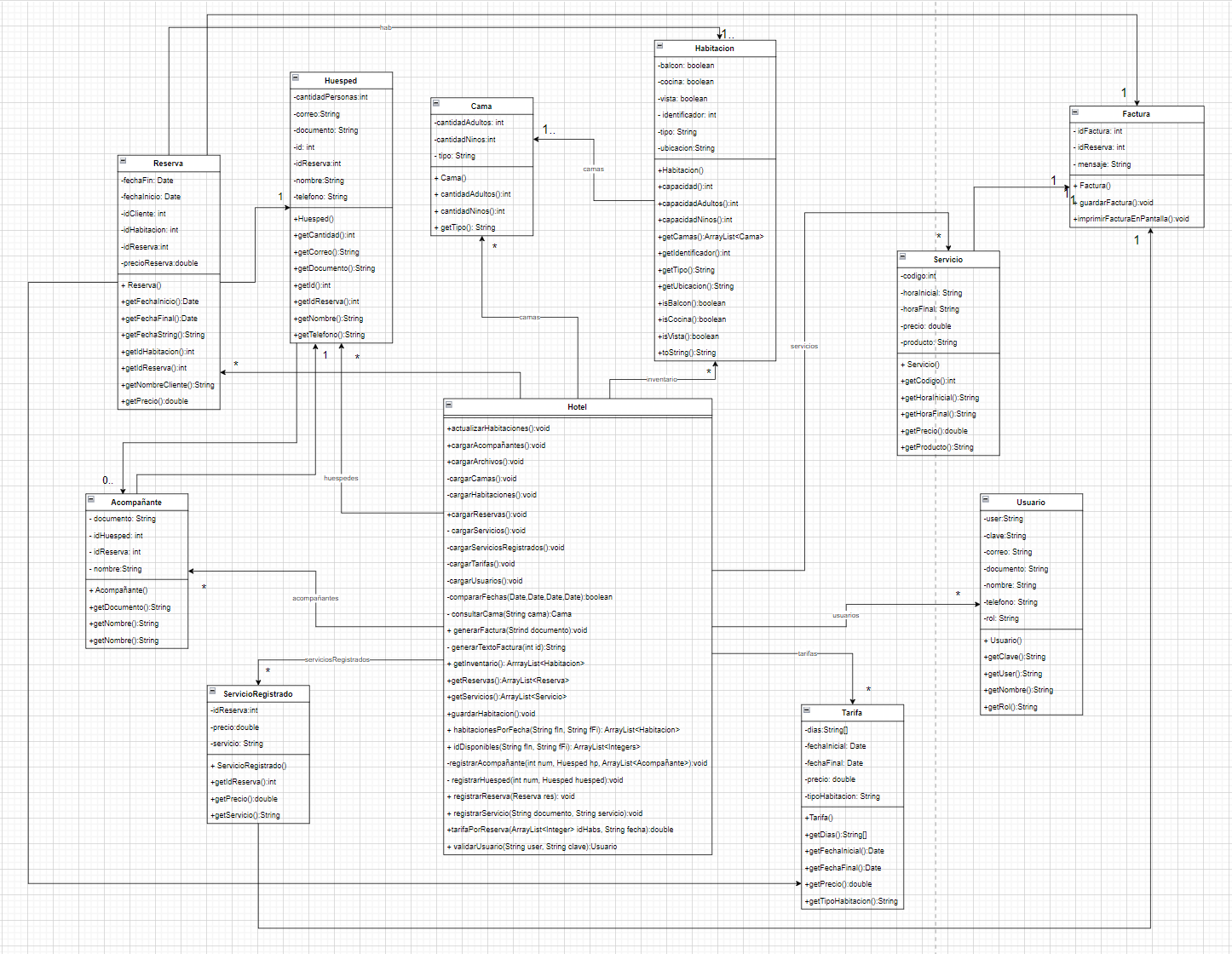
**Menú Recepcionista**



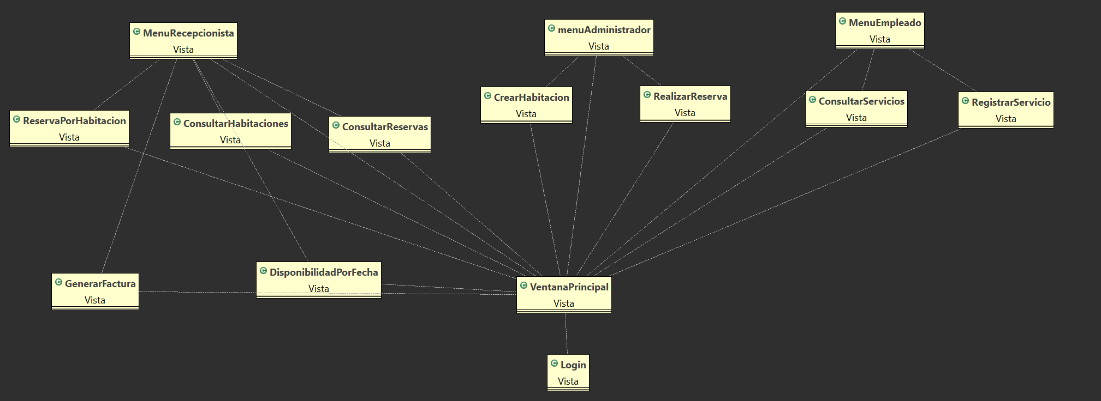
**Diagrama final interfaz**



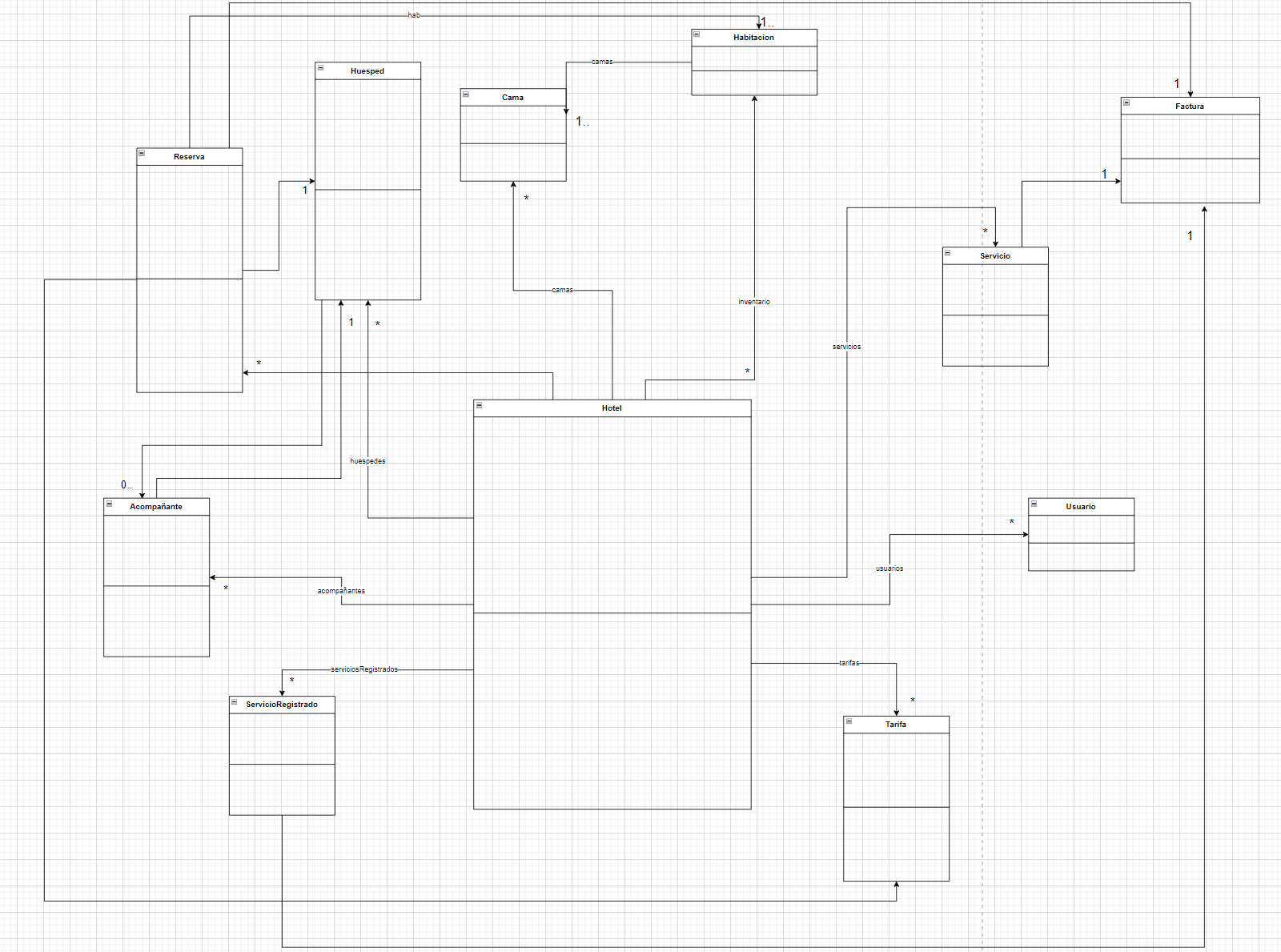
**Diagrama de la lógica**



**Diagrama de alto nivel interfaz**



**Diagrama de alto nivel lógica**



**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.

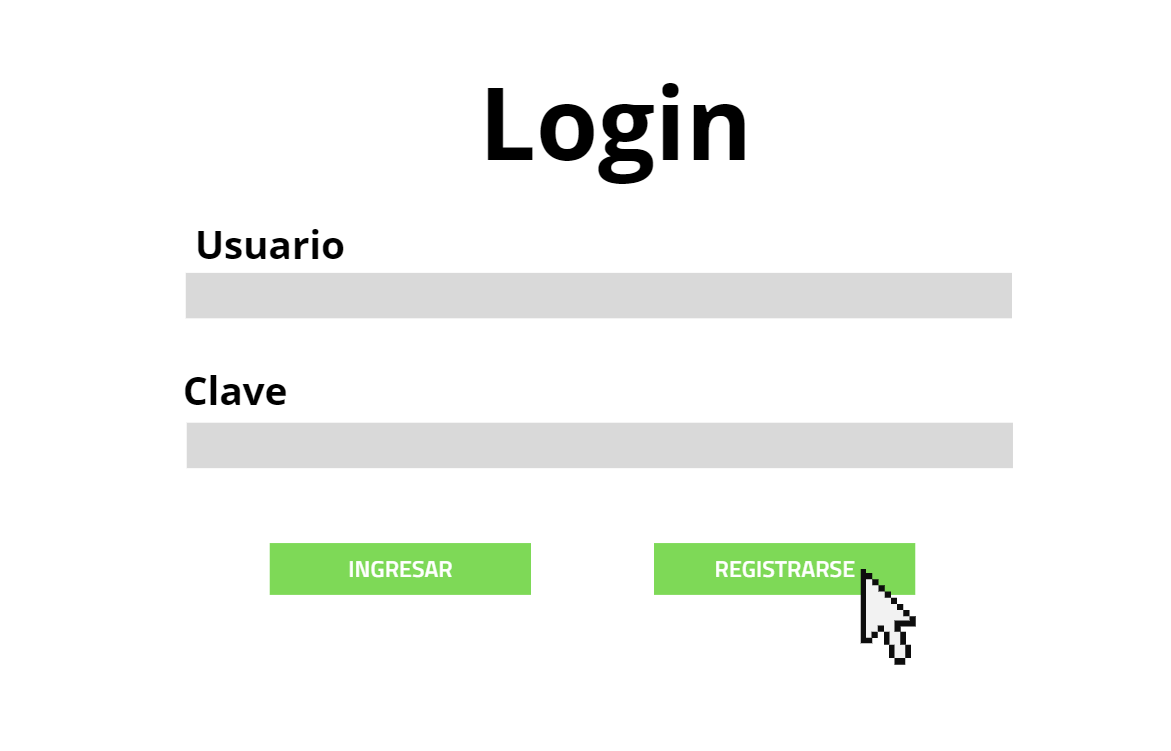
Por otra parte, se debe implemmentar una aplicación para huéspedes, en la que, con base en los requerimientos de reserva de habitaciones y obtener la disponibilidad de habitaciones, se debe desarrollar una interfaz completa, acoplada a la implementación de la nueva lógica capaz de soportar los requerimientos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

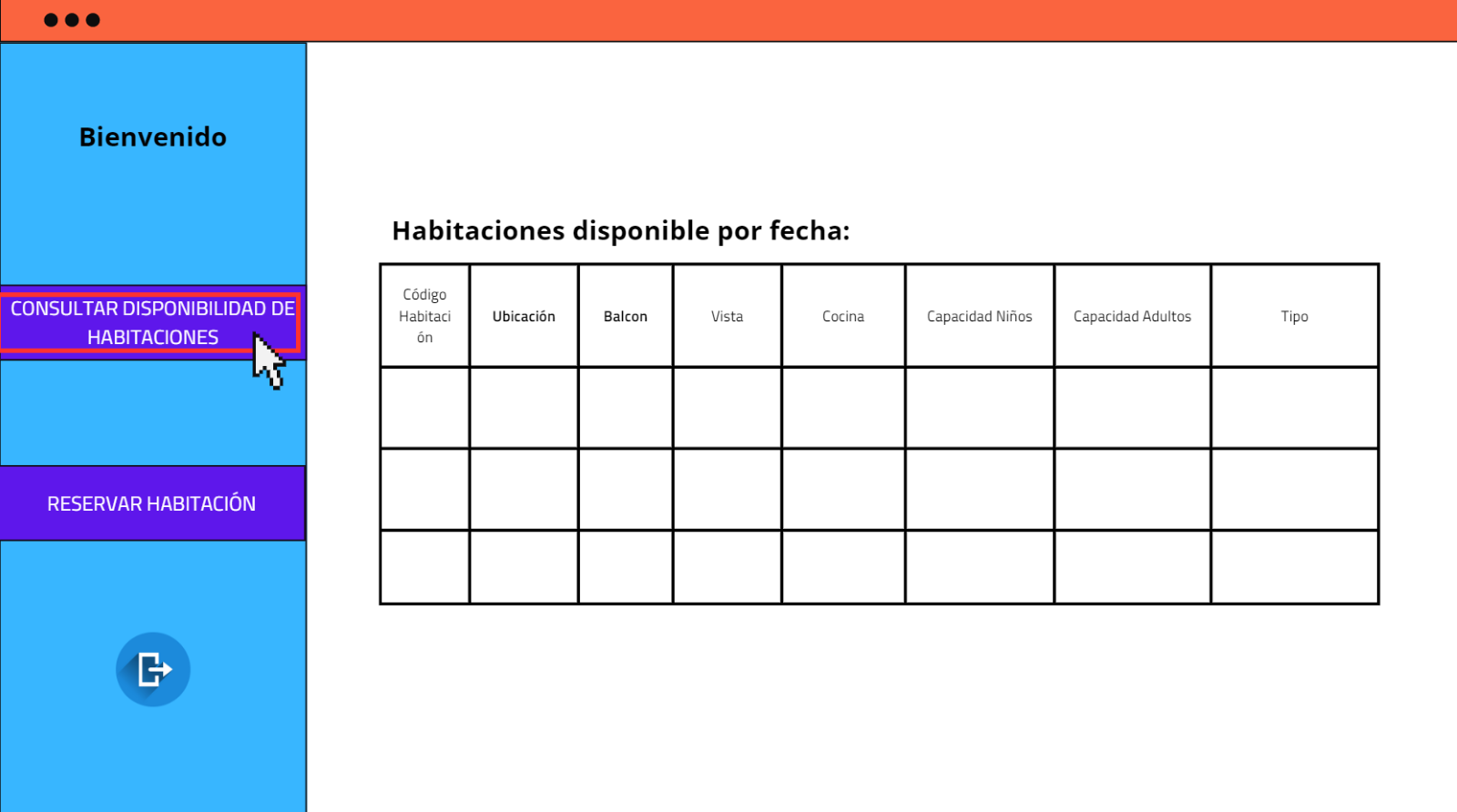
**Login de un usuario de un huésped:**



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



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

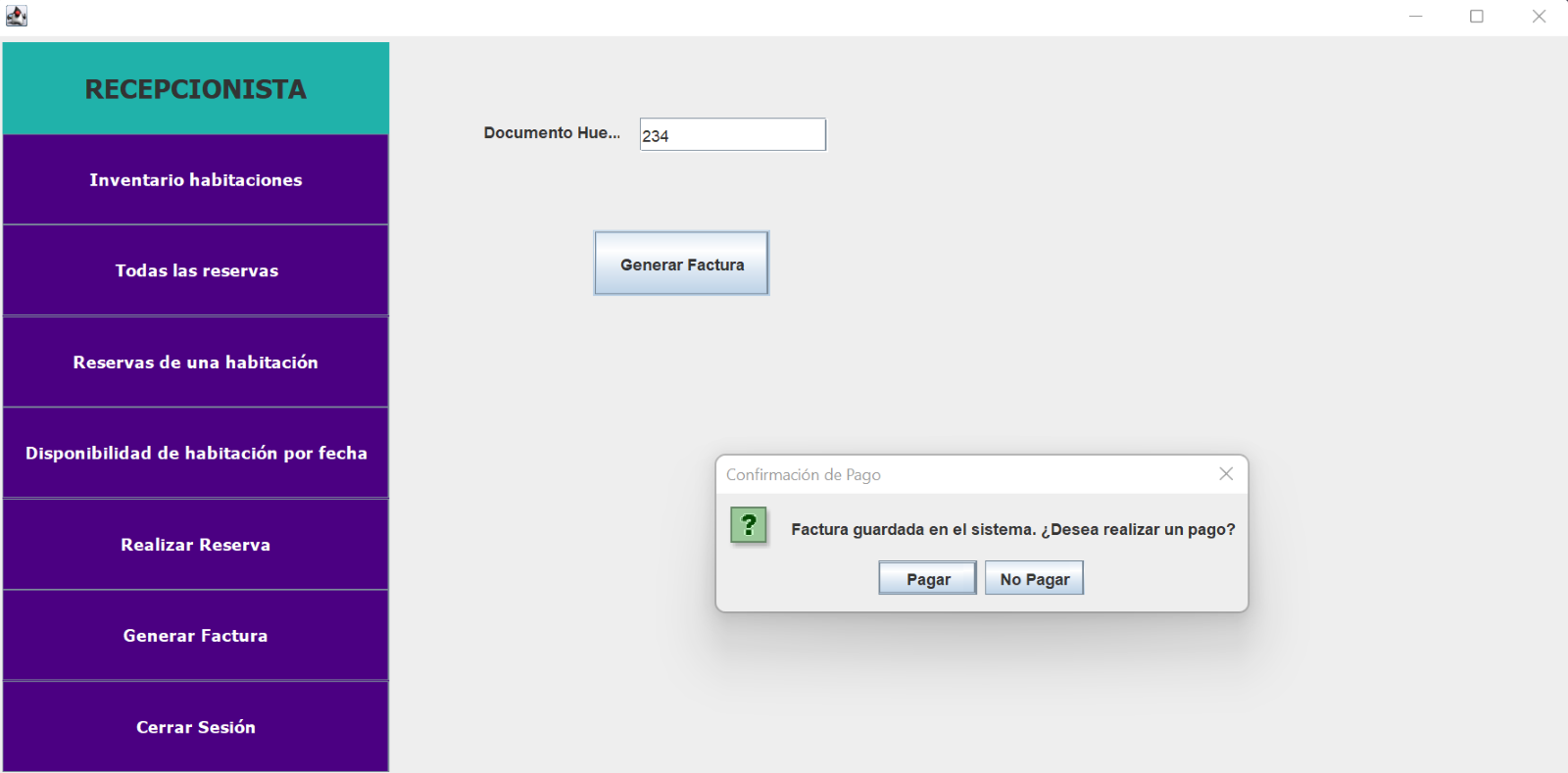


**Segunda opción, reserva de una habitación:**



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

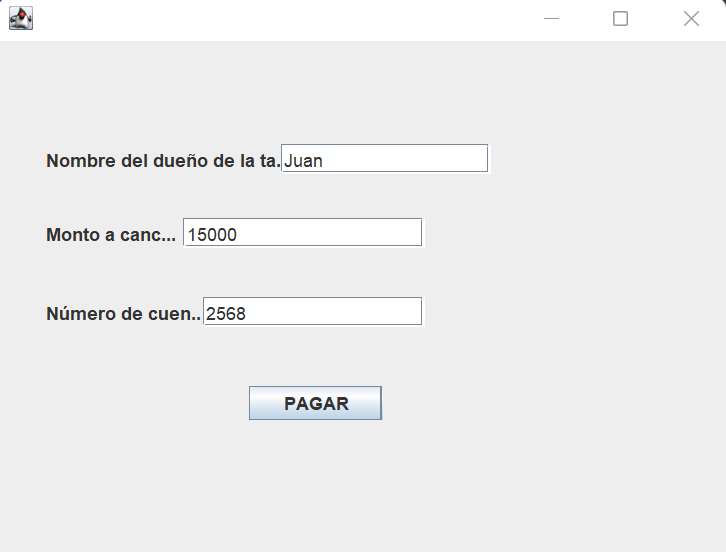
**Selección generar factura:**



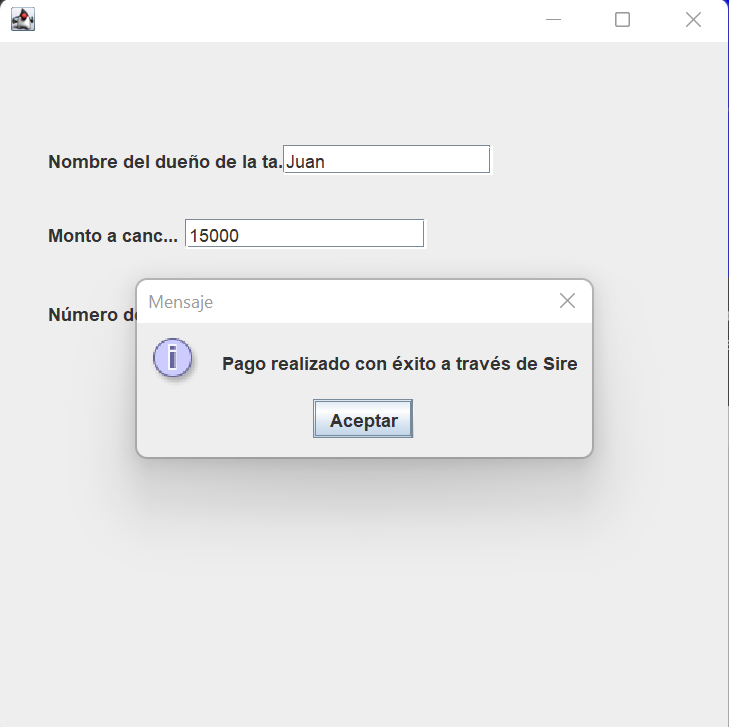
**Selección del método de pago:**



**Página para completar datos:**



**Diálogo con la confirmación del pago:**



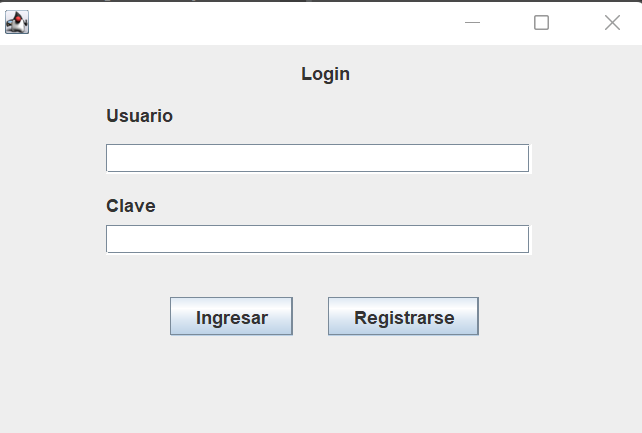
**Aplicación para empleados:**

**Interfaz gráfica**

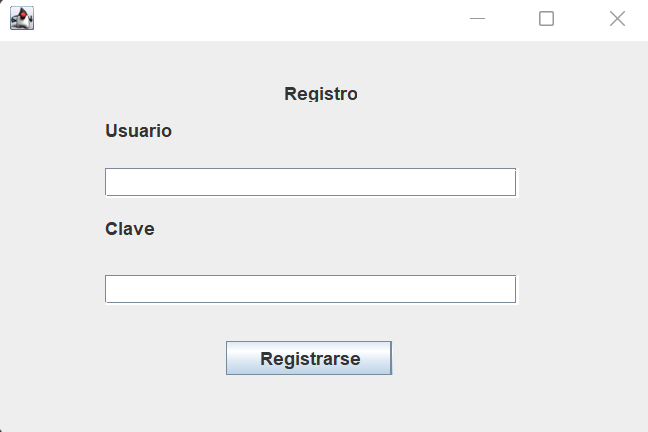
**Login:** En esta parte inicial, se da la opción de ingresar en caso de contar con un usuario ya registrado o, en caso contrario, se registrará el usuario.

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

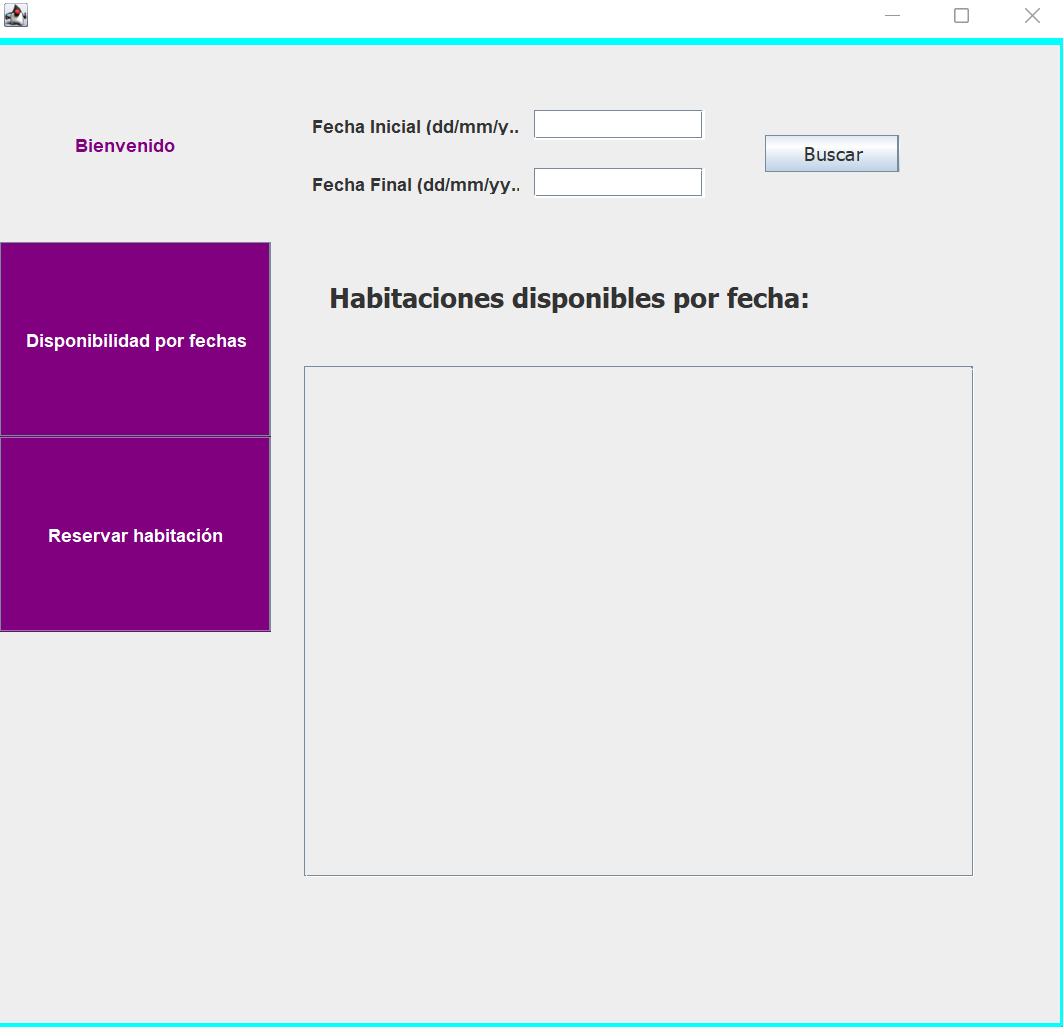
aplicaicón de huéspedes.



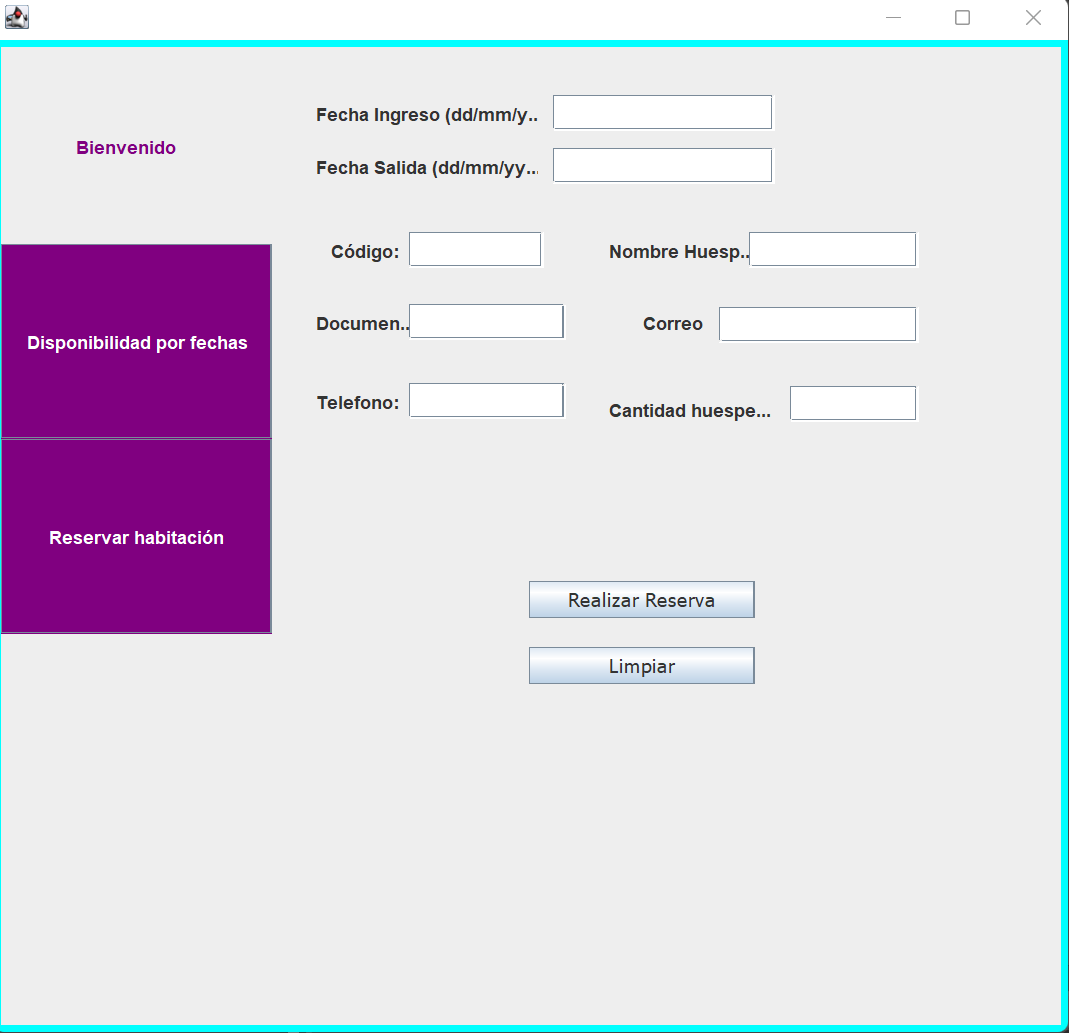
Registro: En esta opción, se podrá ingresar un usuario y contraseña para poder hacer uso de la aplicación de huéspedes.



**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.



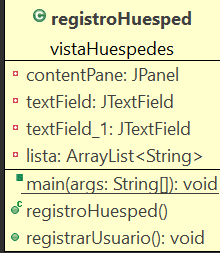
**Ingreso en la opción “Reservar habitación”:** Ahora, luego de revisar la disponibilidad de habitaciones con la anterior opción, el usuario deberá ingresar datos personales, fechas de entrada y salida, la cantidad de huéspedes y el código de la habitación que desea registrar para que se guarde en el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



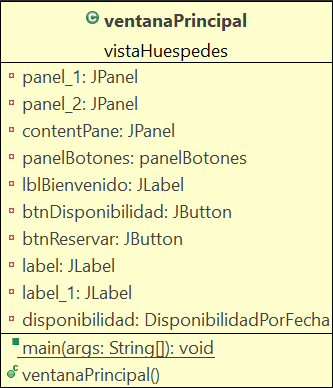
**Diagramas de clase con los cambios implementados:**

**Clases de la aplicación de huéspedes:**

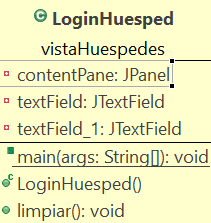
Encargada de registrar el registro de huéspedes, verificando creando el usuario en la plataforma.



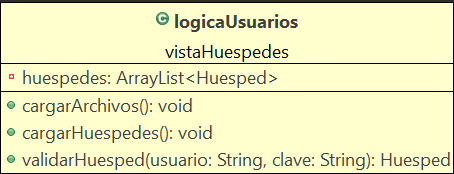
Ventana encargada de soportar los paneles que irán alternando con la selección de los botones. Además, incluye los botones con las funcionalidades disponibles para los huéspedes.



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.



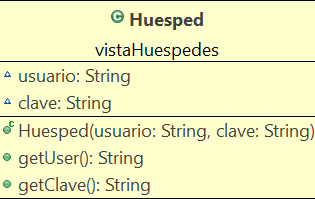
La lógica, es la clase que conecta la carga de Archivos y la carga de huéspedes con la lógica de la clase de los usuarios relacionados con el funcionamiento del hotel. De esta forma, se asegura que los datos estén en tiempo real de la misma plataforma.



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:**

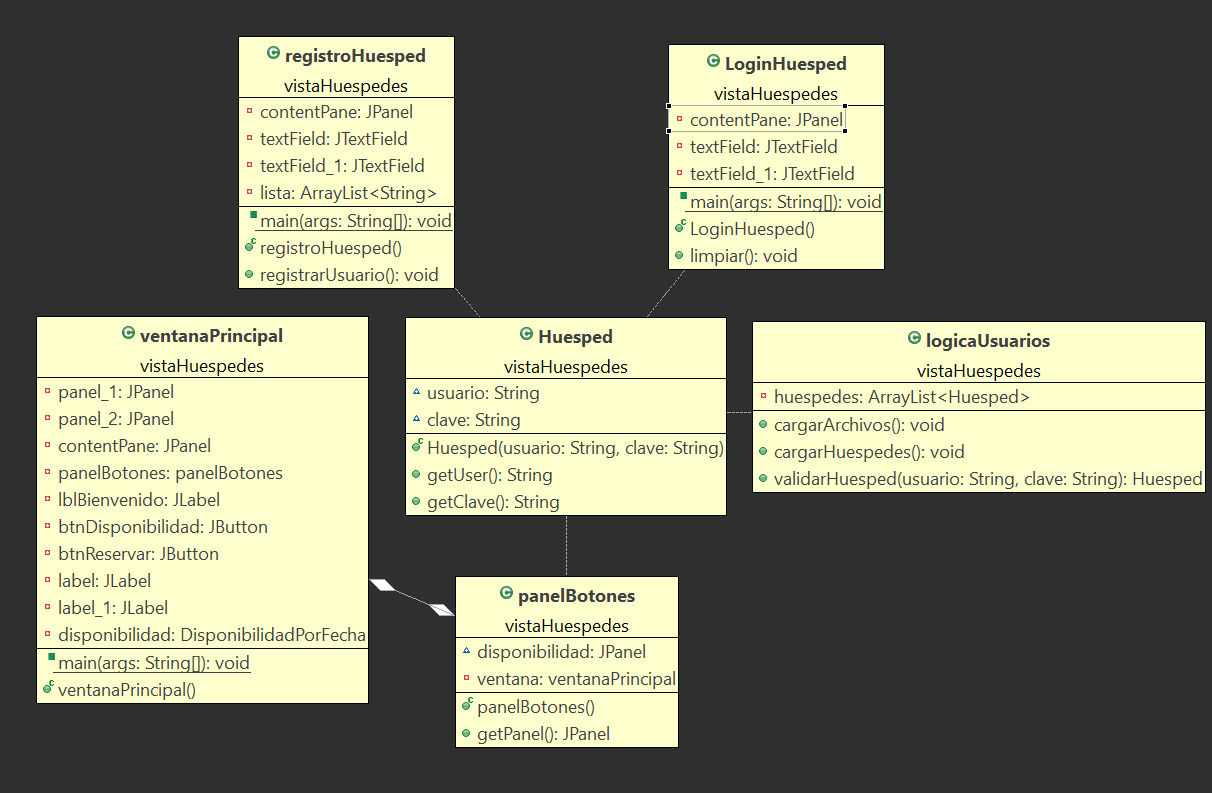
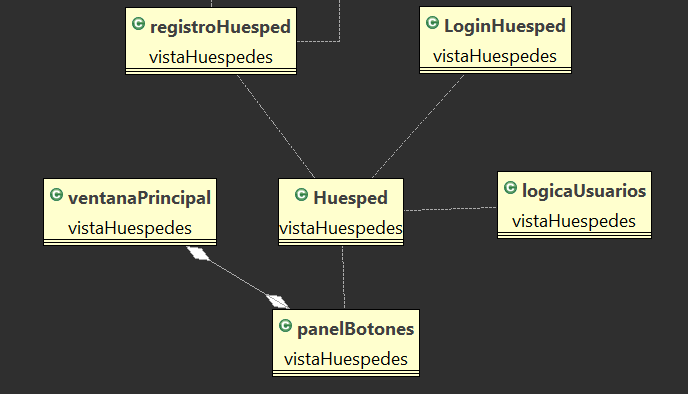


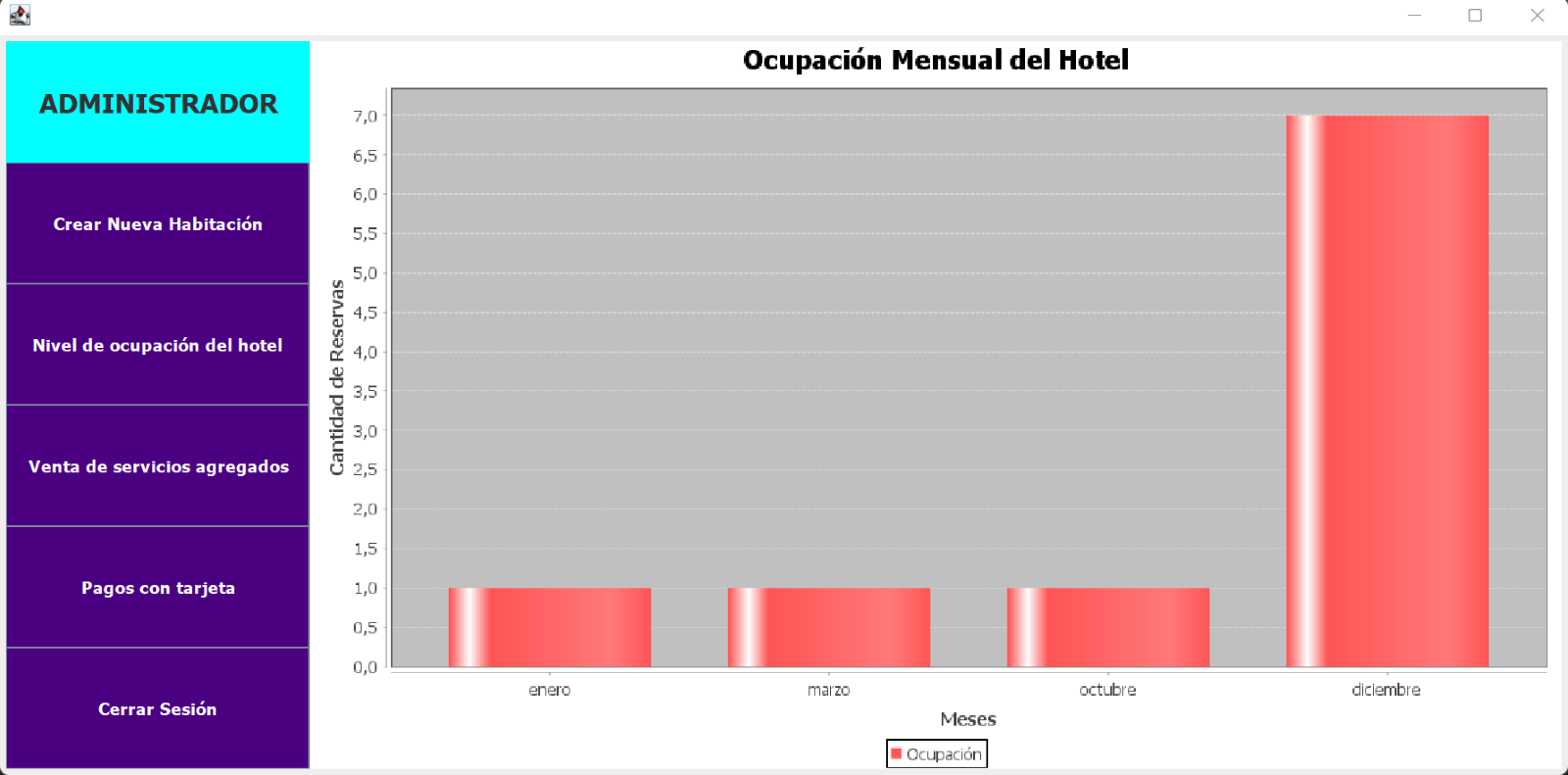
Diagrama de clases de alto nivel para la aplicación de huéspedes:



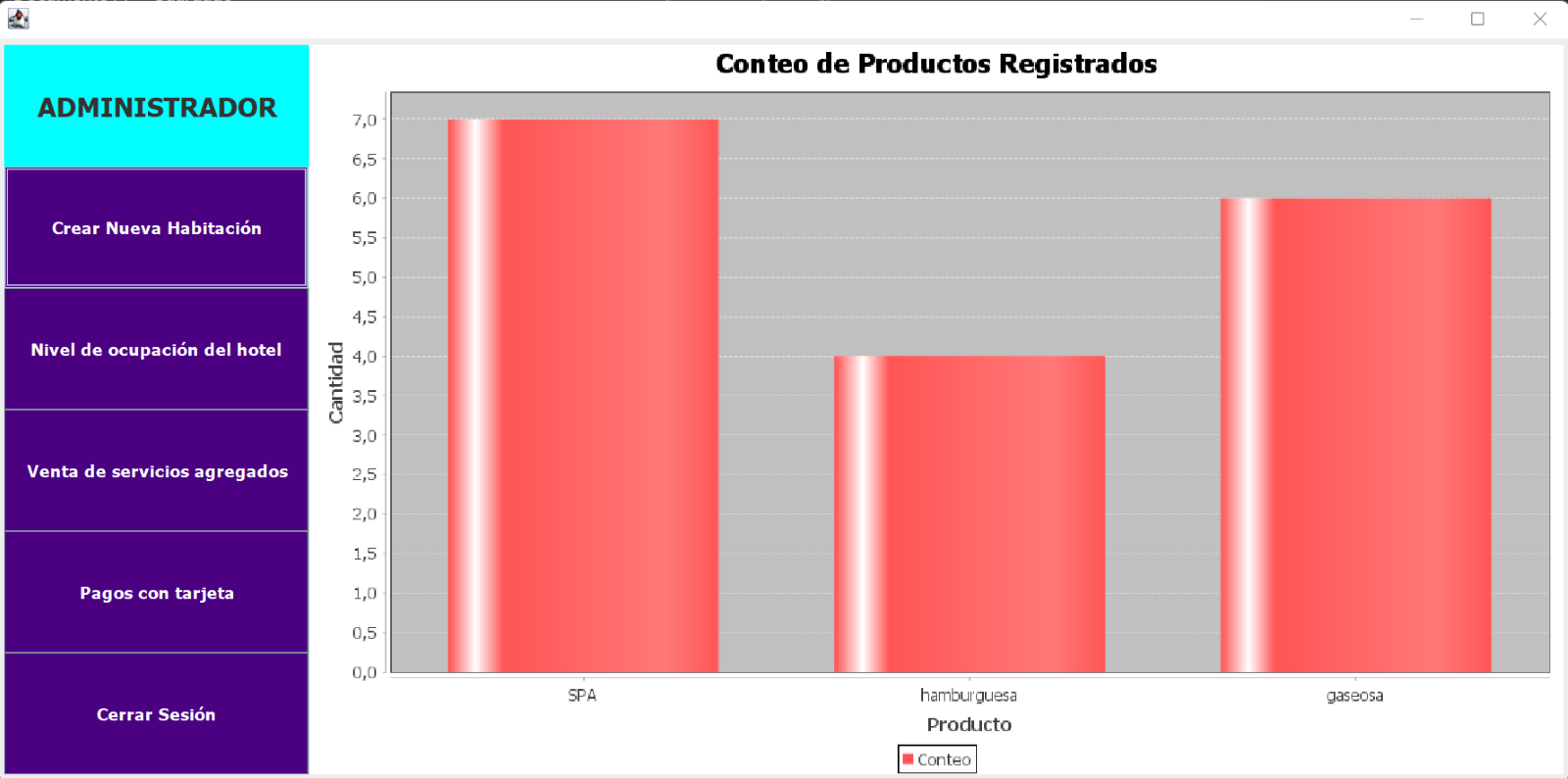
**Adición gráficos informativos menú administrador:**

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

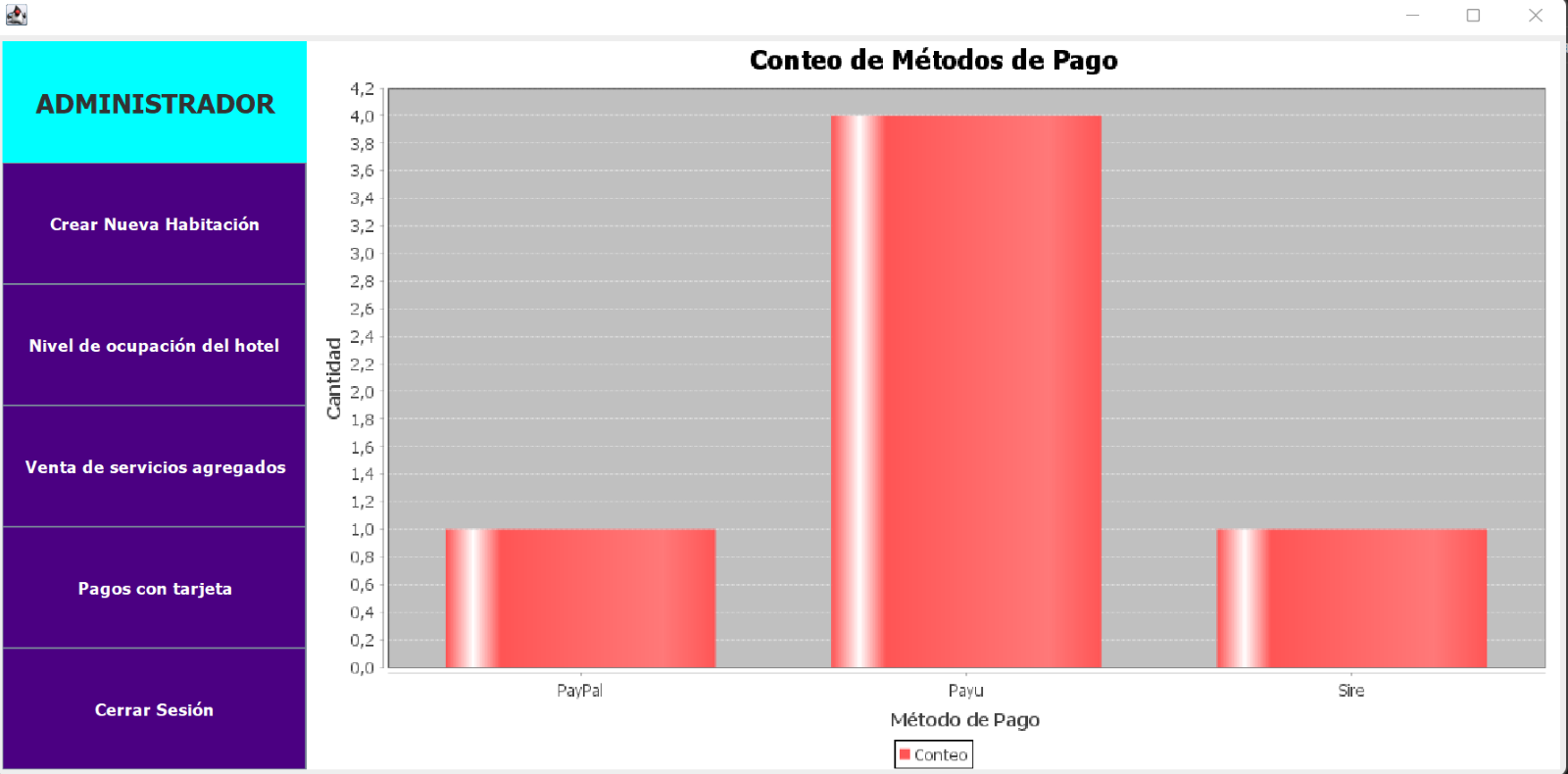
**Información de interés: Usuario: jose1, Contraseña:** 123



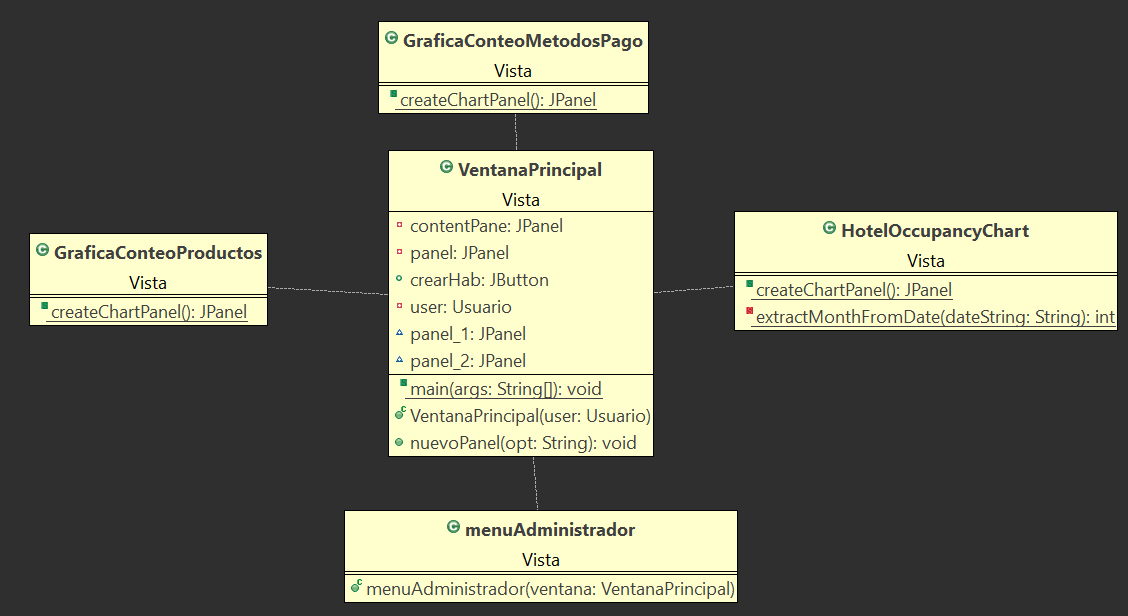
1. **Gráfica venta de servicios en el hotel:**



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



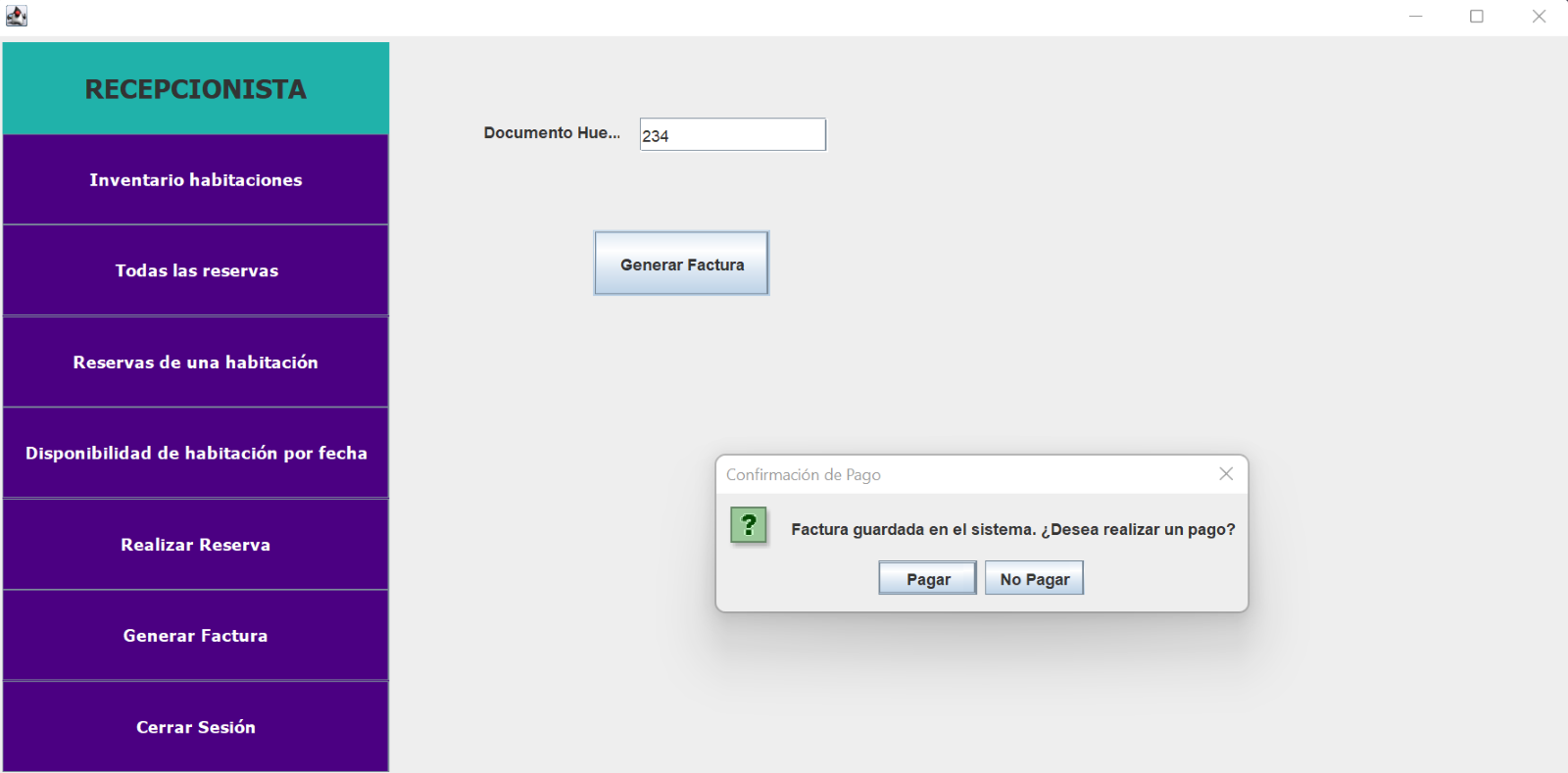
**Diagrama UML relación gráficos con el menú administrador:**



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

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

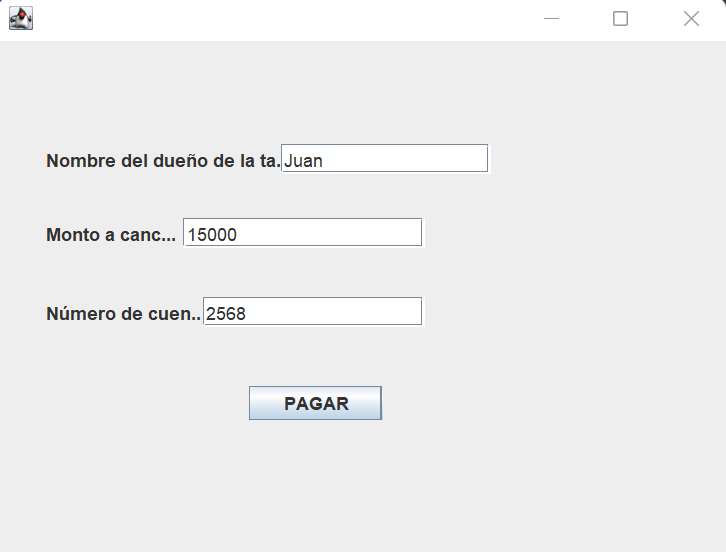
En primer lugar, se genera la factura, en este caso se usa como ejemplo el documento 234 asociado a uno de los huéspedes del hotel.



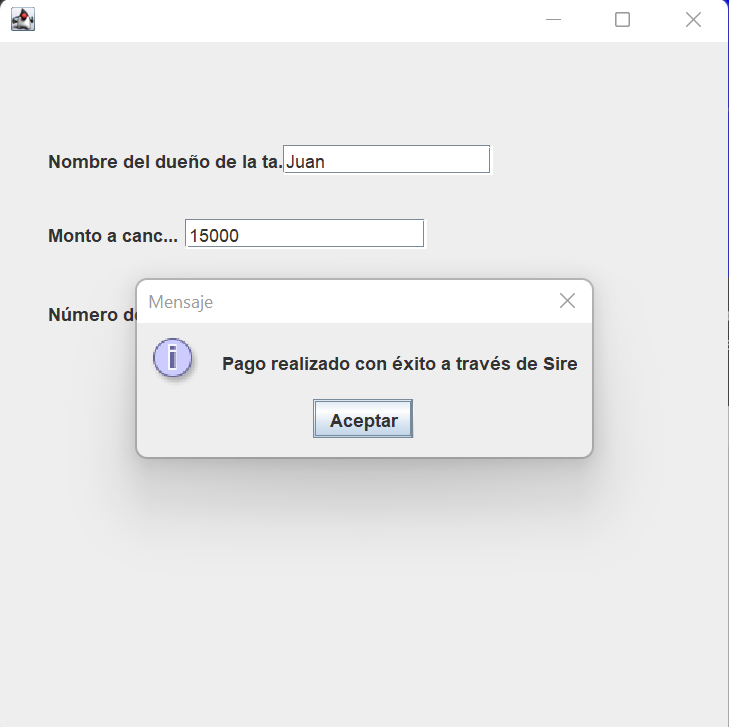
Luego, se selecciona la opción de pagar para que se despliegue el menú de los métodos de pago disponibles:



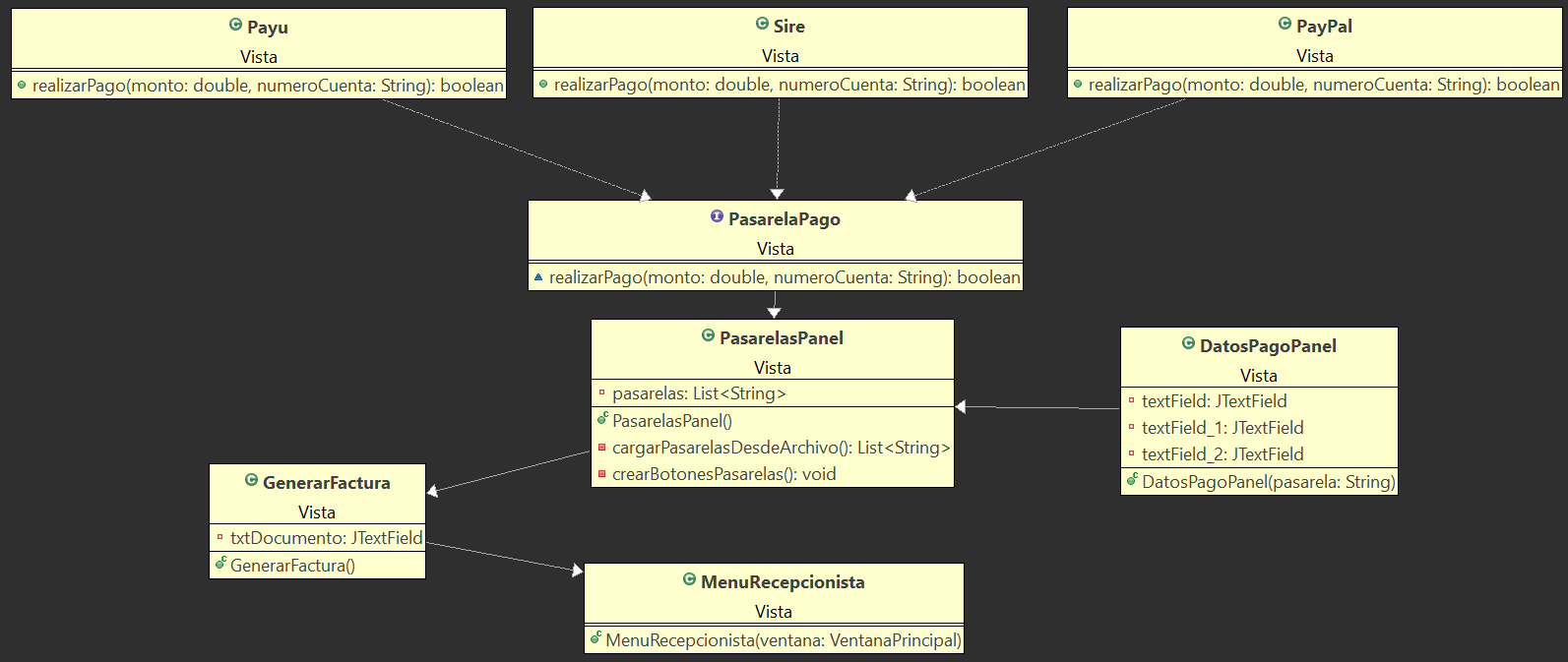
Después, al seleccionar el método de pago, en este caso Sire, se abre una página para completar los datos correspondientes al nombre del titular de la cuenta, la cantidad a cancelar y el número de la tarjeta con la que se realiza el pago. De esta forma, se guardará el pago en el sistema.



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



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



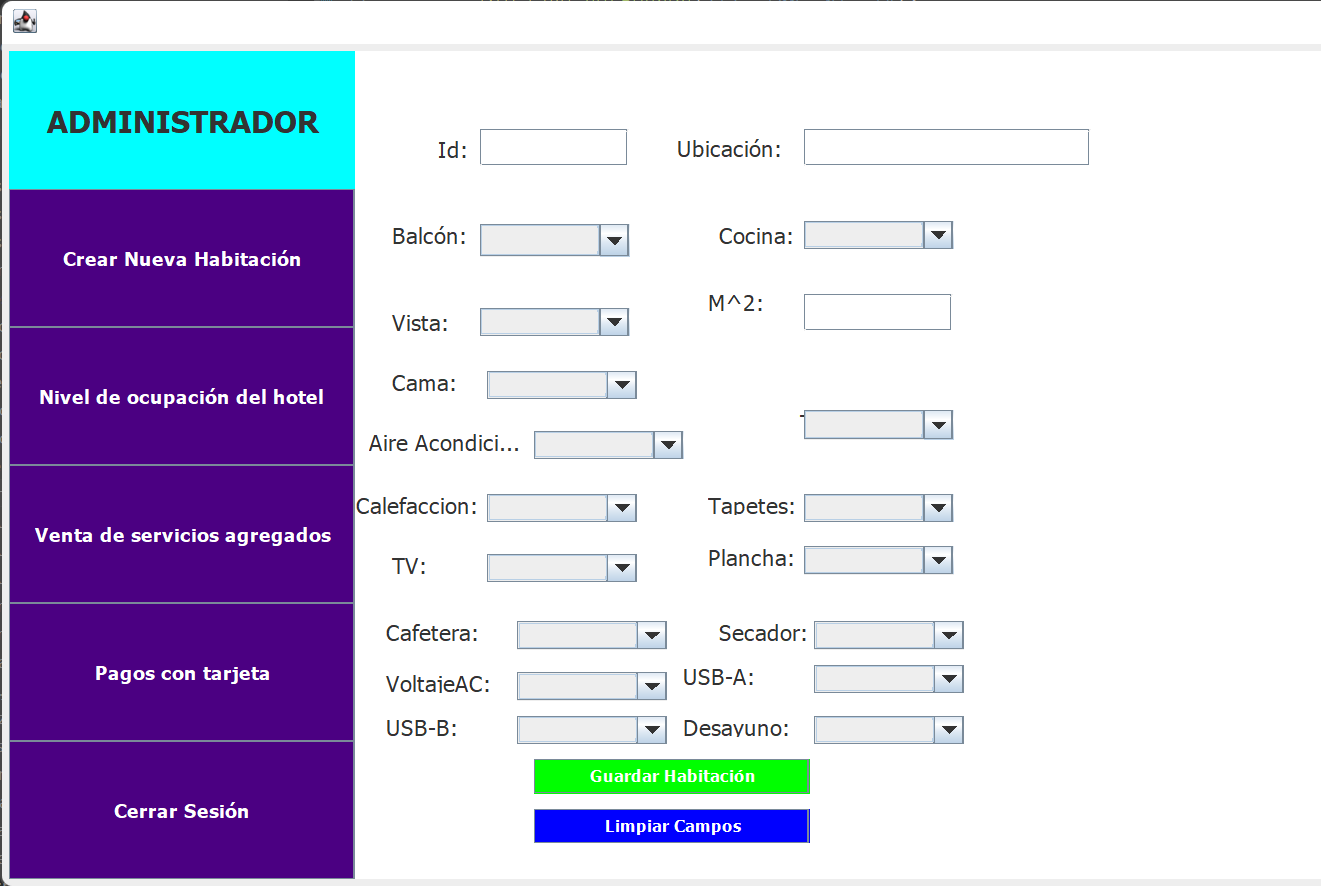
**Colaboración:** Los métodos de pago implementan su propio método de la interfaz PasarelaPago. Así pues, en primer lugar, el menú recepcionista usa el método de GenerarFactura. Luego, PasarelasPanel, que se basa en la selección de alguna de las PasarelasPago que permite la selección de cualquiera de 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.



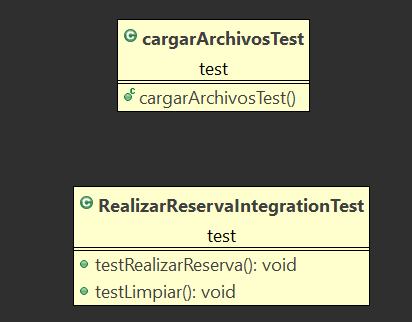
**Adición de elementos en la creación de una habitación desde el menú del administrador:**



**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.