PROYECTO 3 – DISEÑO

Juan David Obando – 202123148

César Avellaneda - 202214746

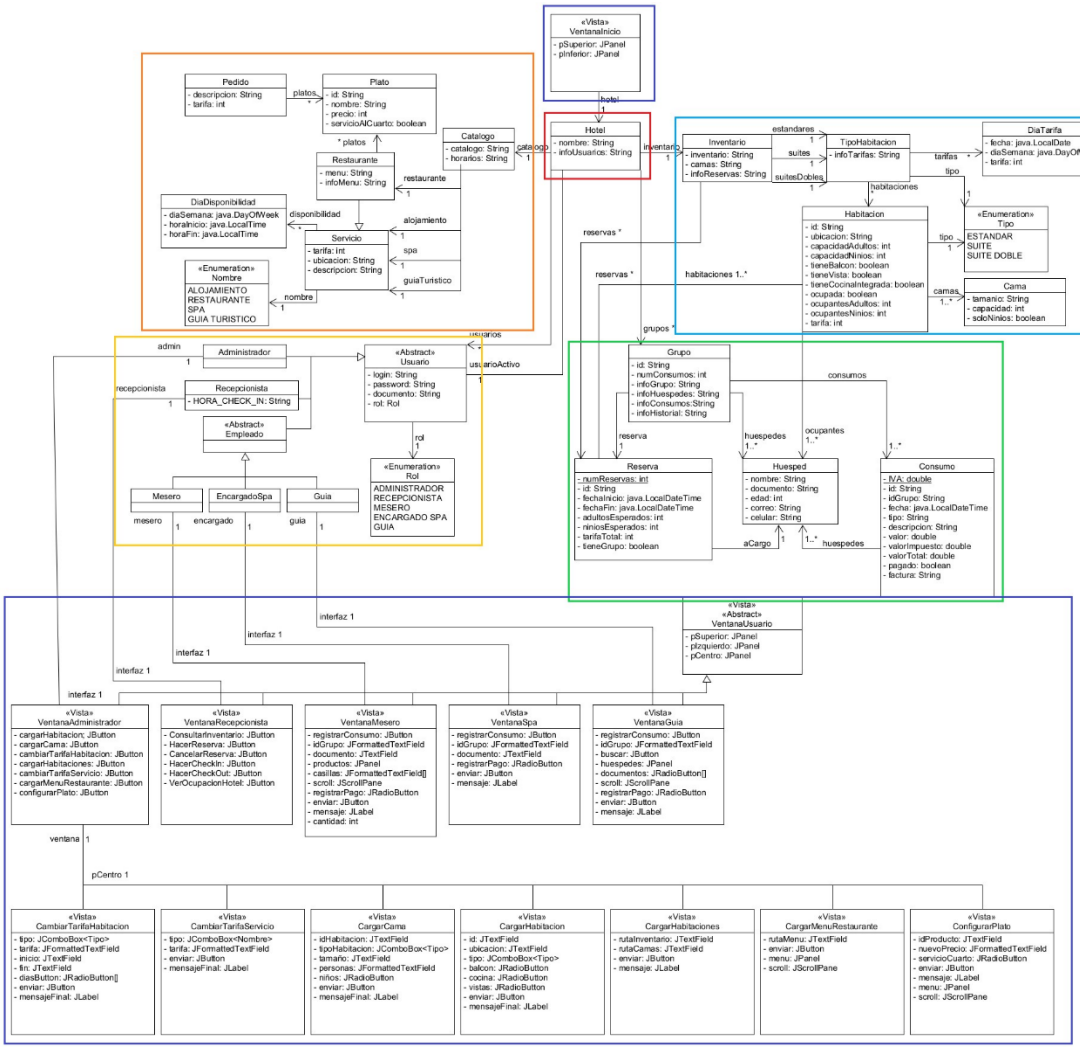
Sara Sofía Cárdenas - 202214907

1. CONTEXTUALIZACIÓN

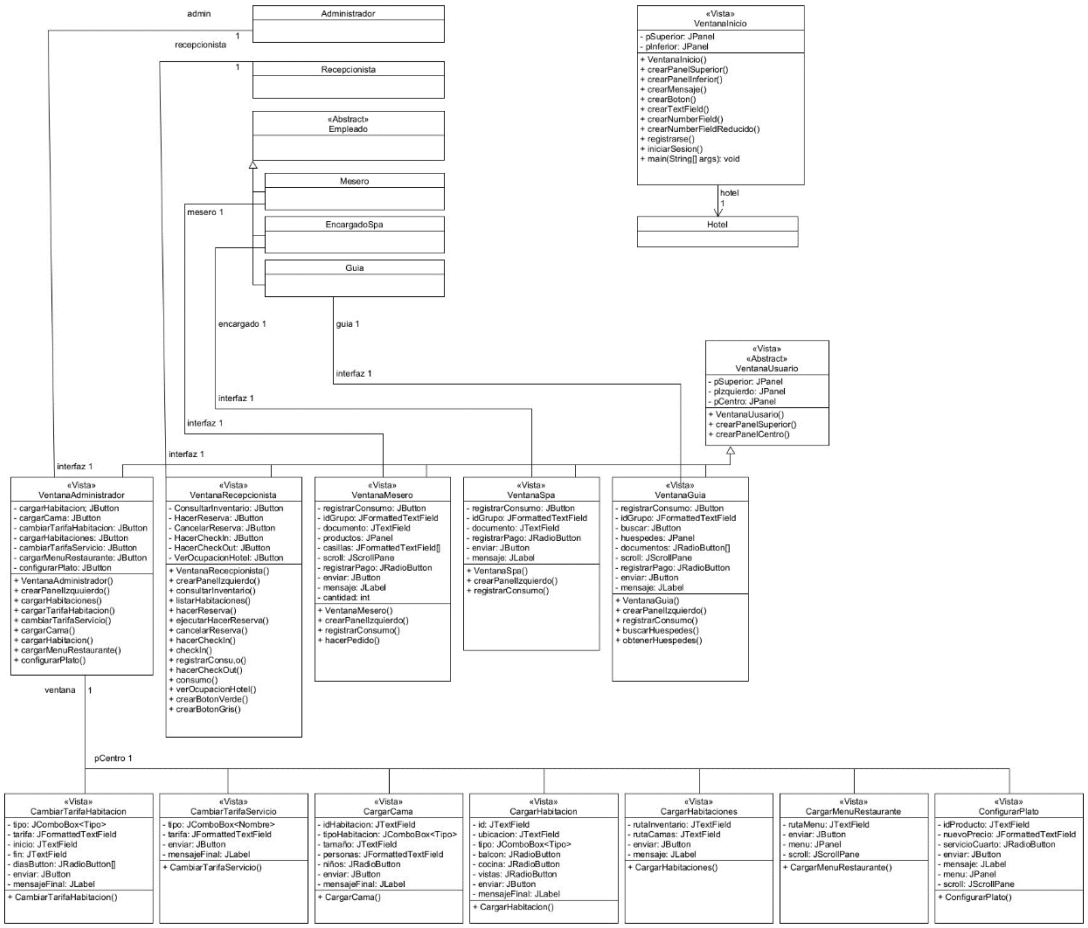
A continuación, se presenta el modelo de dominio

En él existen 6 macrocomponentes, nombrados en sentido horario: el Hotel (rojo), el Inventario (turquesa), los Huéspedes (verde), los Usuarios (amarillo), el Catálogo de servicios (naranja) y la Interfaz gráfica (azul).

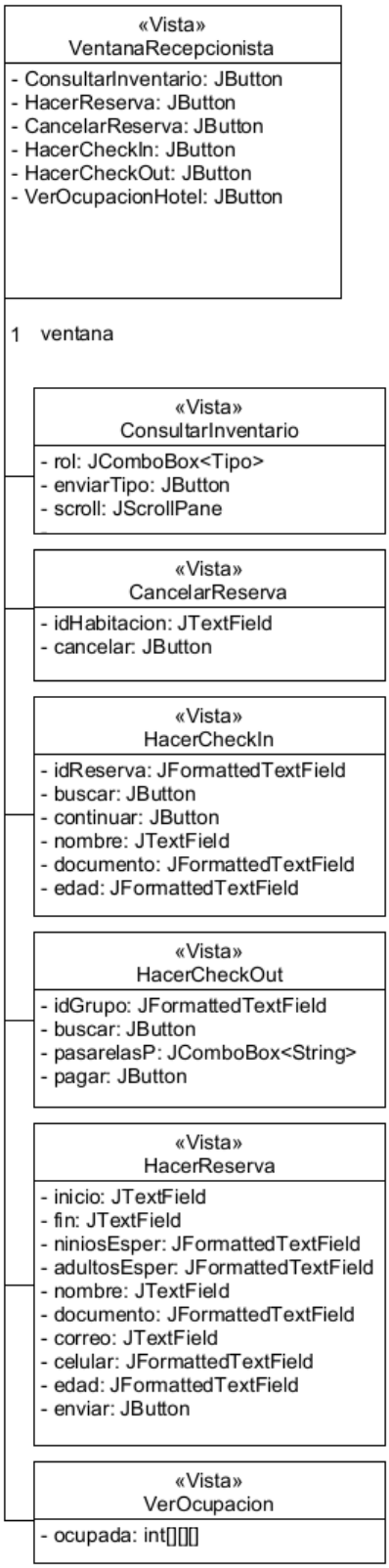
Cada uno de ellos corresponde a un paquete o a varios, en el caso de la interfaz



En concordancia con lo anterior, este es el diagrama de clases de la interfaz gráfica del anterior proyecto



Antes de iniciar el diseño de los nuevos requerimientos, se optó por desagregar las funcionalidades de la interfaz del recepcionista, ya que anteriormente todas se ubicaban en VentanaRecepcionista. Por ello, cada función se convirtió en un panel, como en el caso de la vista del administrador

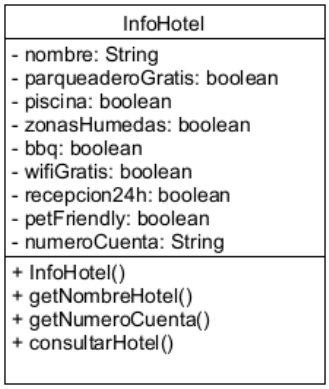


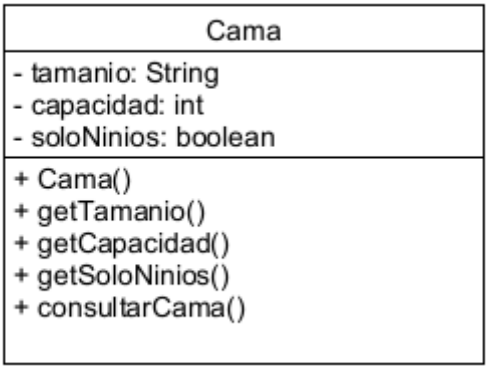
2. CARACTERÍSTICAS DE LAS HABITACIONES Y DEL HOTEL

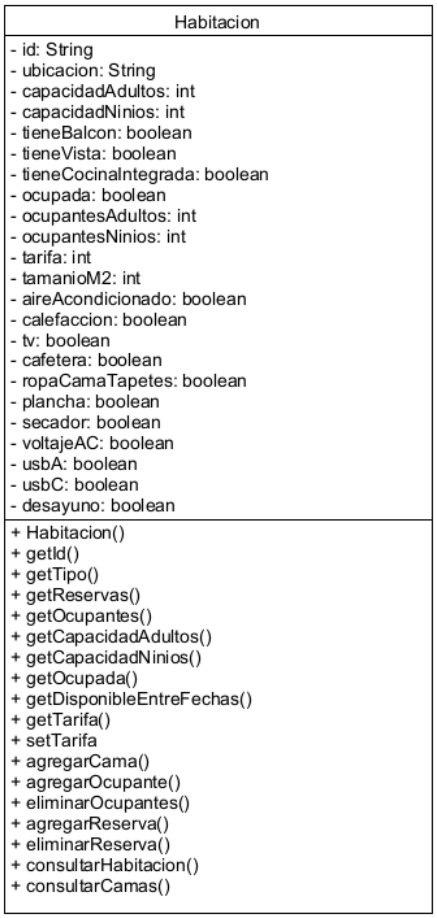
2.1. Componentes

Respecto al modelo, se creó una nueva clase en el Inventario llamada InfoHotel, que contiene los nuevos atributos del hotel. Sus métodos son en su totalidad getters.

Para las habitaciones, se amplió la clase Habitación para soportar los nuevos atributos. De esta manera, se añadieron más getters. Como las habitaciones se componen de Camas, esta clase fue extendida para mostrar su información.

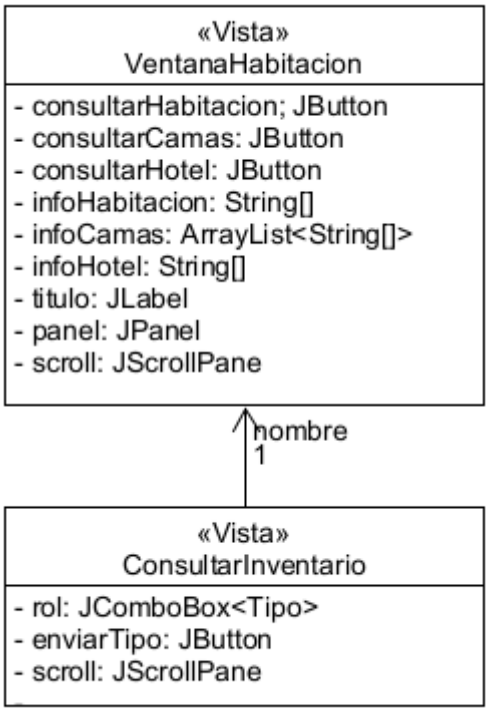






Respecto a la interfaz, se cambió un JDialog por una nueva clase que extiende de JFrame llamada VentanaHabitacion, con el fin de mostrar de forma más organizada las nuevas características de las habitaciones, la información de las camas y los

atributos del hotel. Esta clase extiende de VentanaUsuario para reutilizar la forma de la interfaz.



2.2. Responsabilidades

N°	Componente	Responsabilidad
1	InfoHotel	Guardar la información del hotel
2	InfoHotel	Proveer la información del hotel
3	Habitación	Guardar la información de la habitación
4	Habitación	Proveer la información de la habitación
5	Habitación	Guardar sus respectivas camas
6	Cama	Guardar la información de las camas
7	Cama	Proveer la información de las camas
8	ConsultarInventario	Mostrar el inventario de las habitaciones
9	VentanaHabitacion	Mostrar la información del hotel, la habitación seleccionada y sus camas

2.3. Colaboraciones

Para mostrar esta información, tuvieron que hacerse cambios en el modelo (las clases Inventario y Recepcionista) para conducir la información hasta la vista. De igual forma hubo modificaciones en la vista (JFrame Vista Recepcionista y JPanel ConsultarInventario) para desplegar la nueva ventana y hacerle llegar los datos requeridos. Por ello, en realidad estos métodos no tienen lógica adicional.

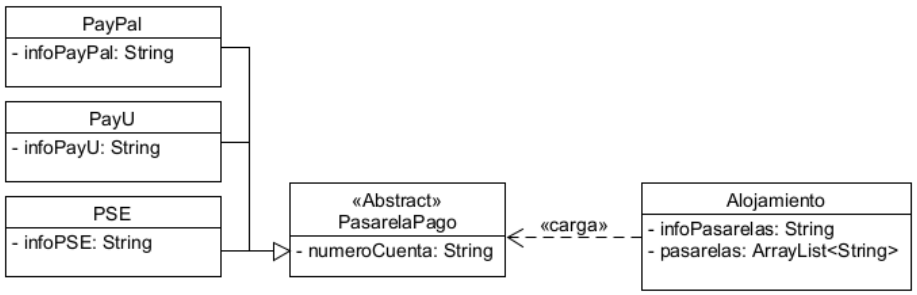
3. PAGOS CON PASARELA DE PAGOS

3.1. Componentes

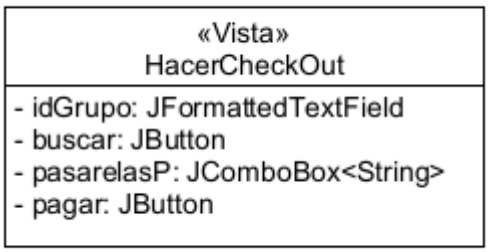
Decidimos crear una clase abstracta PasarelaPago para modela todas las pasarelas de pagos que extiendan de ella, y de esta manea hacer la carga dinámica de clases usando la herencia. Las clases concretas son PayPal, PayU y PSE, pero podrían añadirse fácilmente otras pasarelas.

Cada pasarela necesita el número de cuenta del hotel y crea o actualiza su archivo de persistencia. PayPal y PayU usan archivos .txt, mientras que PSE usa .csv, con el fin de mostrar que sus implementaciones pueden ser distintas.

Se creó entonces una nueva clase Alojamiento que extiende de Servicio con el fin de modelar este servicio, el cual es responsable de cargar también las pasarelas usadas en el check-out y en el pago de las reservas



Para el ambiente gráfico, en el panel hacerCheckOut del recepcionista, se extiende un JComboBox para mostrar las pasarelas disponibles, al igual que se lanza un JOptionPane con el fin de recopilar los datos necesarios para la transacción. De igual forma, con otro JOptionPane se notifica el éxito o no del pago del check-out



3.2. Responsabilidades

N°	Componente	Responsabilidad
1	Alojamiento	Guardar la información del servicio
2	Alojamiento	Cargar los nombres de las pasarelas
3	PasarelaPago	Modelar una pasarela abstracta

4	PasarelaPago	Hacer una transacción genérica
5	PayPal, PayU y PSE	Cargar archivo de persistencia
6	PayPal, PayU y PSE	Registrar una transacción
7	HacerCheckOut	Hacer check-out
8	HacerCheckOut	Mostrar pasarelas disponibles
9	HacerCheckOut	Pagar el check-out con una pasarela

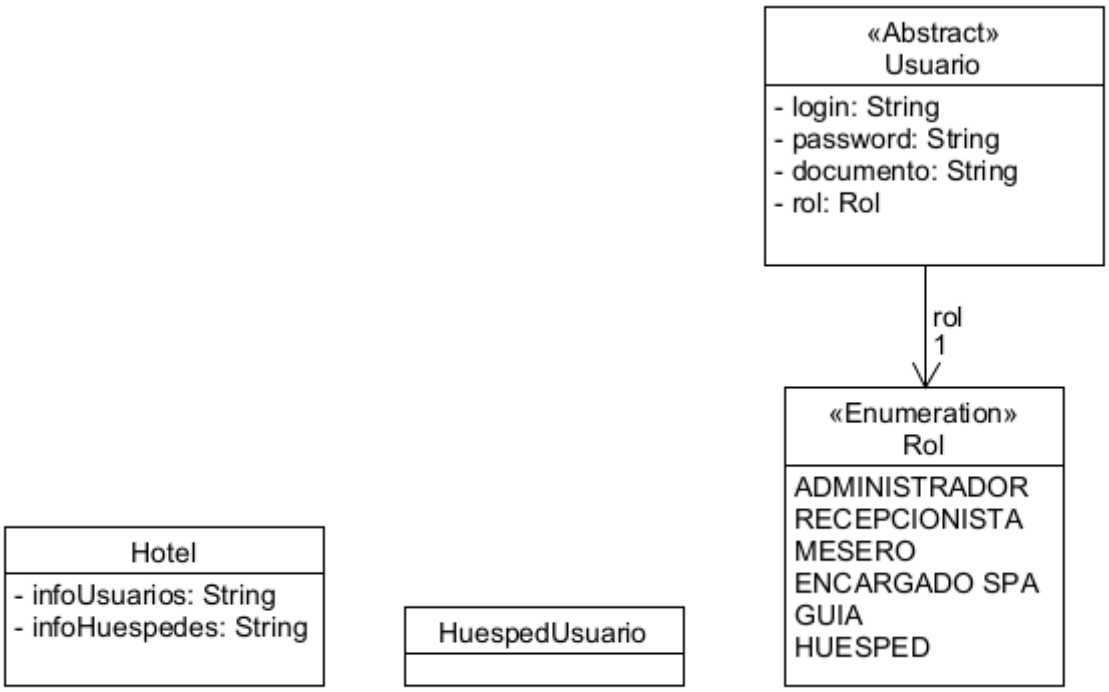
3.3. Colaboraciones

Alojamiento se encarga de cargar los nombres de las pasarelas, mientras PasarelaPago permite modelar los canales de pago. Cuando en HacerCheckOut se tienen todos los datos, se carga una clase dinámicamente con los nombres almacenados por Alojamiento, y de esta manera, se invoca el registro de la transacción y el pago del check-out.

4. APLICACIÓN PARA HUÉSPEDES

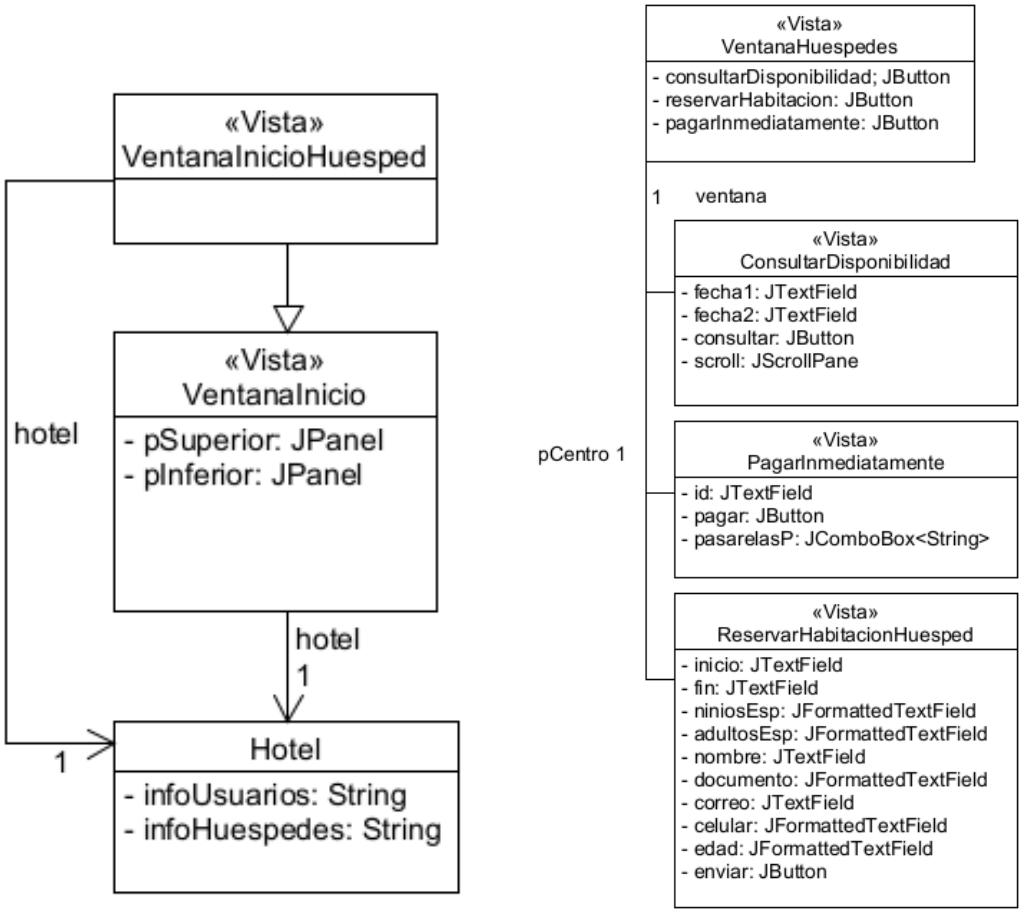
4.1. Componentes

Ahora el Hotel debe cargar los huéspedes desde un archivo, almacenado en infoHuespedes. De igual forma, es necesaria una clase HuespedUsuario que extienda de Usuario para ser un agente que interactúa con el sistema. Por ello, uno de los nuevos Roles es ser Huésped.



Para la nueva aplicación, se necesita un nuevo punto de entrada. VentanaInicioHuesped cumple esta función y la de registrar o autenticar un huésped, extentiendo de VentanaInicio para reutilizar sus funciones. VentanaHuesped cumple

la función de mostrar los requerimientos del huésped. De esta manera, los páneles necesarios son ConsultarDisponibilidad, PagarInmediatamente y ReservarHabitacionHuesped



4.2. Responsabilidades

N°	Componente	Responsabilidad
1	Hotel	Cargar los huéspedes registrados
2	HuespedUsuario	Modelar un huésped en la nueva app
3	Rol	Modelar un nuevo rol para el huésped
4	VentanaInicioHuesped	Ser punto de entrada de la nueva app
5	VentanaInicioHuesped	Registrar y autenticar un huésped
6	VentanaHuesped	Mostrar los requerimientos del huésped
7	ConsultarDisponibilidad	Mostrar la disponibilidad de habitaciones entre dos fechas
8	PagarInmediatamente	Permitir el pago de una reserva antes del check-in
9	PagarInmediatamente	Conectarse a una pasarela de pago
10	ReservarHabitacion Huesped	Hacer una reserva en un rango de fechas

4.3. Colaboraciones

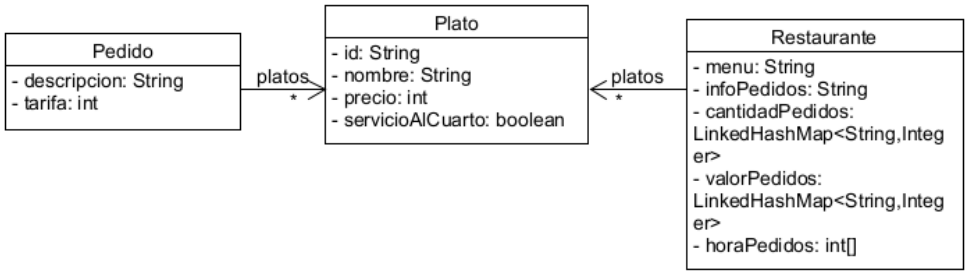
El Hotel carga los huéspedes. Cuando el huesped inicia sesión en VentanaInicioHuesped, VentanaHuesped se despliega para mostrar las opciones disponibles. Allí el huésped puede consultar la disponibilidad de las habitaciones, hacer una reserva o pagarla inmediatamente. Si la paga inmediatamente, el sistema se conecta a una pasarela de pagos, la cual registra el pago y la transacción.

5. REPORTES Y GRÁFICAS DEL RESTAURANTE

5.1. Componentes

Ahora es necesario recopilar información sobre el valor de los pedidos, los productos seleccionados y la frecuencia de estas actividades. Por ello, se añadieron atributos en Restaurante para guardar esta información a la hora de hacer pedidos y añadir productos

De igual manera, la persistencia tuvo cambios. En el menú se guarda el total de items comprados de cada producto y el dinero recibido por cada uno de ellos. Además, se creó un archivo llamado infoPedidos para agrupar los totales por día



Para la interfaz, se creó una VentanaEstadisticas que extiende de VentanaUsuario, con la finalidad de mostrar varios tipos de gráficas respecto a las ventas, las facturas y pedidos del restaurante.

Se tienen 5 gráficas: cantidad de items vendidos por cada producto, valor recibido por cada producto, ventas por horarios del día, cantidad de facturas por día y el valor de todas las facturas por día.

Decidimos hacer uso de la librería XChart para las gráficas, principalmente de barras y de torta, las cuales se muestran en la aplicación. También se adjunta una tabla con los datos en detalle

5.2. Responsabilidades

N°	Componente	Responsabilidad
1	Restaurante	Recopilar información sobre los pedidos
2	Restaurante	Hacer la persistencia de los datos sobre los pedidos
3	Pedido	Brindar información de los platos esocgidos y la tarifa total
4	VentanaEstadisticas	Mostrar las estadísticas del restaurante con XChart y tablas en Swing

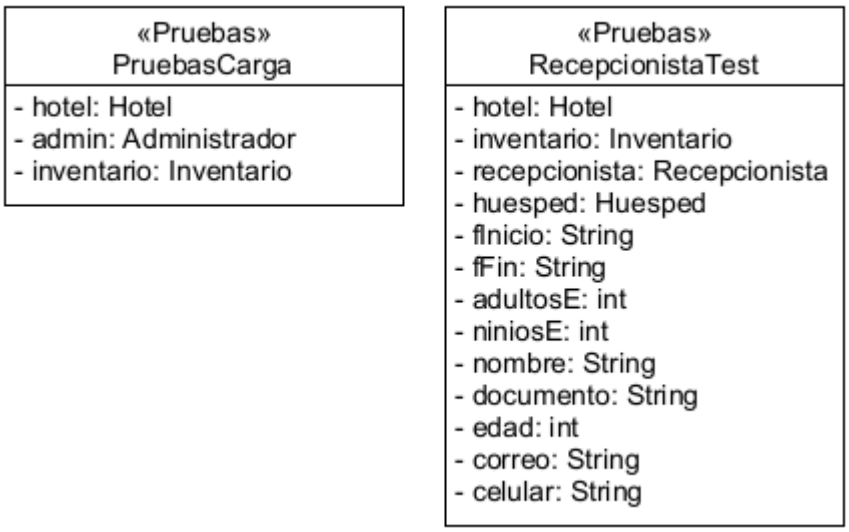
5.3. Colaboraciones

En Pedido se guarda información de los productos y la tarifa total. La información general de los pedidos es almacenada por Restaurante. Posteriormente desde VentanaAdministrador se despliega VentanaEstadisticas, y de esta manera se muestra la información gráficamente. El acceso a las estructuras de datos es por métodos estáticos de Restaurante.

6. PRUEBAS AUTOMÁTICAS

6.1. Componentes

Se creó un paquete llamado pruebas para agrupar todas las pruebas automáticas de la creación de reservas y la carga de archivos en persistencia. En él se crearon las clases PruebasCarga y RecepcionistaTest para los propósitos ya mencionados



6.2. Diseño de pruebas y errores

Se utilizó la metodología given-when-then para diseñar las pruebas unitarias y de integracion

Cargar Habitaciones Administrador

given: El hotel está creado

when: El administrador carga las habitaciones

then: El inventario tiene todas las habitaciones cargadas

Cargar Menú Administrador

given: El hotel y el restaurante está creado

when: El administrador carga el menú

then: El restaurante tiene todos los platos cargados

Cargar Usuarios Hotel

given: El hotel está creado

when: El hotel se inicializa y se le permite cargar usuarios

then: Los usuarios están disponibles y pueden iniciar sesión

Cargar Huéspedes Hotel

given: El hotel está creado

when: El hotel se inicializa y se le permite cargar huéspedes

then: Los huéspedes están disponibles y pueden iniciar sesión

Test Reserva

given: El hotel y el inventario está creado

when: Se crea una reserva

then: La reserva es creada con los datos suministrados

Test Consumo

given: El hotel y el inventario está creado

when: Se crea un consumo del alojamiento

then: El consumo es creado con los datos de la reserva

Test Check Out

given: El hotel está creado

when: Se inicia el check-out con un grupo

then: Se actualiza el estado del grupo y ya no pertenece al hotel

Test Get Consumo

given: El hotel está creado

when: Se busca un consumo

then: Se encuentra un consumo pagado o no

Cabe resaltar que los errores corresponden en su totalidad a IOException, CsvException y excepciones chequeadas en el caso de las reservas y su manejo

6.3. Responsabilidades

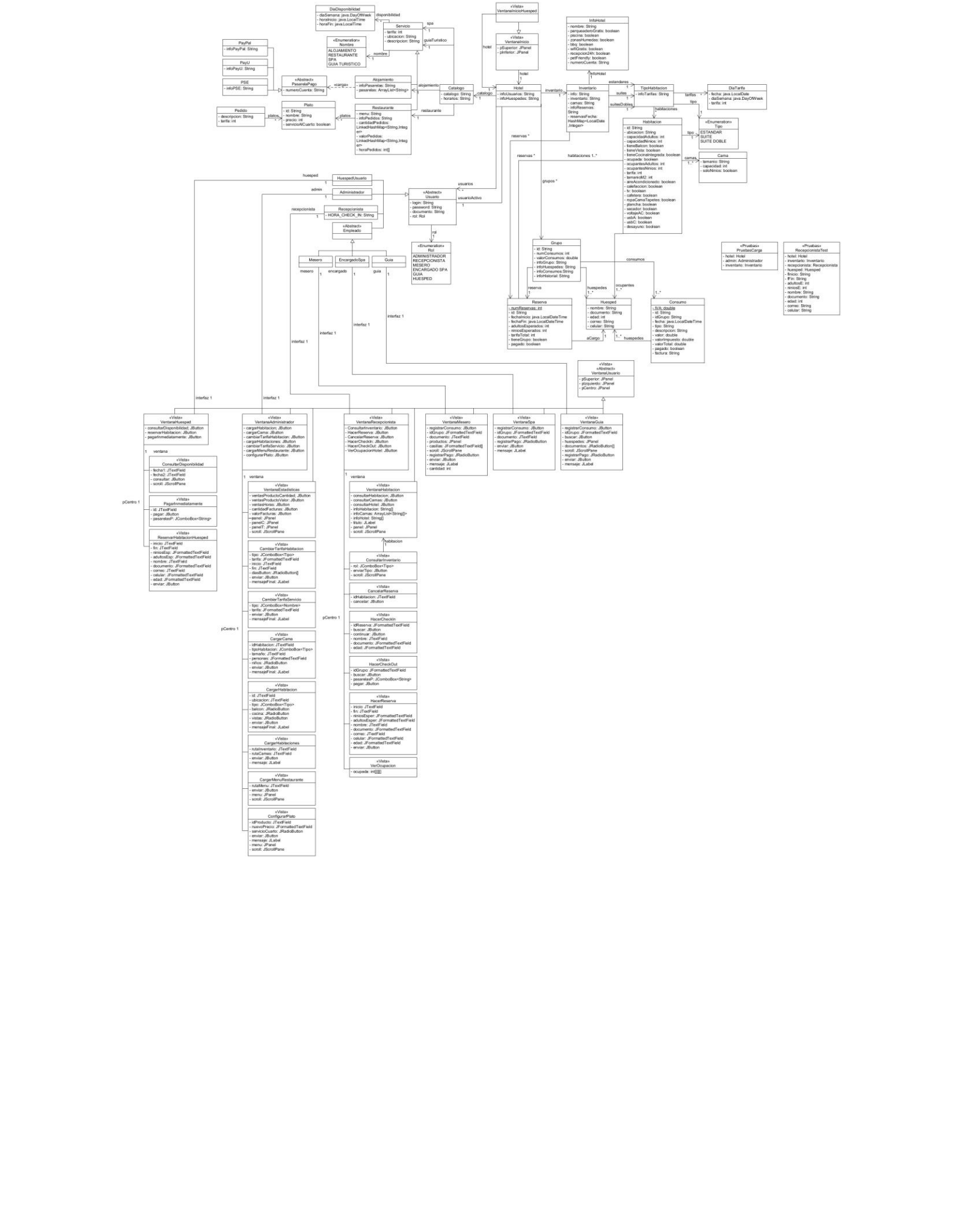
N°	Componente	Responsabilidad
1	PruebasCarga	Prueba los posibles caminos que pueden tomar las funciones encargadas de cargar datos.
2	RecepcionistaTest	Prueba las diferentes funciones que tiene la clase recepcionista.

6.4. Colaboraciones

Las pruebas se conectan con las clases Hotel, Inventario, Catálogo, Restaurante y Administrador para la carga de datos. De igual forma, se conectan con Reserva, Inventario, Huesped y Recepcionista para la creación de reservas.

7. DISEÑO FINAL

A continuación, se presenta el modelo de dominio y el diagrama de clases extendido de toda la aplicación


DiaDisponibilidad
- diaSemana: java.DayOfWeek
- horaInicio: java.LocalTime
- horaFin: java.LocalTime
disponibilidad
Servicio
- tarifa: int
- ubicacion: String
- descripcion: String
spa
guiaTuristico
«Enumeration»
Nombre
ALOJAMIENTO
RESTAURANTE
SPA
GUIA TURISTICO
nombre
«Vista»
VentanaInicioHuesped
«Vista»
VentanaInicio
- pSuperior: JPanel
- pInferior: JPanel
hotel
InfoHotel
- nombre: String
- parqueaderoEnGratis: boolean
- piscina: boolean
- zonasHumedas: boolean
- bbq: boolean
- wifiGratis: boolean
- recepcion24h: boolean
- petFriendly: boolean
- numeroCuenta: String
PayPal
- infoPayPal: String
PayU
- infoPayU: String
PSE
- infoPSE: String
«Abstract»
PasarelaPago
- numeroCuenta: String
«carga»
Alojamiento
- infoPasarelas: String
- pasarelas: ArrayList<String>
alojamiento
Catalogo
- catalogo: String
- horarios: String
catalogo
Hotel
- infoUsuarios: String
- infoHuespedes: String
Inventario
- info: String
- inventario: String
- camas: String
- infoReservas: String
- reservasFecha: HashMap<LocalDate ,Integer>
estandares
suites
suitesDobles
TipoHabitacion
- infoTarifas: String
tarifas
tipo
habitaciones
DiaTarifa
- fecha: java.LocalDate
- diaSemana: java.DayOfWeek
- tarifa: int
Pedido
- descripcion: String
- tarifa: int
platos
Plato
- id: String
- nombre: String
- precio: int
- servicioAlCuarto: boolean
platos
Restaurante
- menu: String
- infoPedidos: String
- cantidadPedidos: LinkedHashMap<String,Integer>
- valorPedidos: LinkedHashMap<String,Integer>
- horaPedidos: int[]
restaurante
Habitacion
- id: String
- ubicacion: String
- capacidadAdultos: int
- capacidadNinios: int
- tieneBalcon: boolean
- tieneVista: boolean
- tieneCocinaIntegrada: boolean
- ocupada: boolean
- ocupantesAdultos: int
- ocupantesNinios: int
- tarifa: int
- tamanioM2: int
- aireAcondicionado: boolean
- calefaccion: boolean
- tv: boolean
- cafetera: boolean
- mqaCamaTapetes: boolean
- plancha: boolean
- secador: boolean
- voltajeAC: boolean
- usbA: boolean
- usbC: boolean
- desayuno: boolean
«Enumeration»
Tipo
ESTANDAR
SUITE
SUITE DOBLE
Cama
- tamanio: String
- capacidad: int
- soloNinios: boolean
reservas *
habitaciones 1..*
huesped
HuespedUsuario
admin
Administrador
recepcionista
Recepcionista
- HORA_CHECK_IN: String
«Abstract»
Empleado
«Abstract»
Usuario
- login: String
- password: String
- documento: String
- rol: Rol
usuarios
usuarioActivo
rol
«Enumeration»
Rol
ADMINISTRADOR
RECEPCIONISTA
MESERO
ENCARGADO SPA
GUIA
HUESPED
Mesero
EncargadoSpa
Guia
mesero
encargado
guia
Grupo
- id: String
- numConsumos: int
- valorConsumos: double
- infoGrupo: String
- infoHuespedes: String
- infoConsumos:String
- infoHistorial: String
grupos *
consumos
reserva
huespedes
1..*
ocupantes
1..*
Reserva
- numReservas: int
- id: String
- fechaInicio: java.LocalDateTime
- fechaFin: java.LocalDateTime
- adultosEsperados: int
- ninosEsperados: int
- tarifaTotal: int
- tieneGrupo: boolean
- pagado: boolean
Huesped
- nombre: String
- documento: String
- edad: int
- correo: String
- celular: String
aCargo
huespedes
Consumo
- IVA: double
- id: String
- idGrupo: String
- fecha: java.LocalDateTime
- tipo: String
- descripcion: String
- valor: double
- valorImpuesto: double
- valorTotal: double
- pagado: boolean
- factura: String
«Pruebas»
PruebasCarga
- hotel: Hotel
- admin: Administrador
- inventario: Inventario
«Pruebas»
RecepcionistaTest
- hotel: Hotel
- inventario: Inventario
- recepcionista: Recepcionista
- huesped: Huesped
- finicio: String
- fFin: String
- adultosE: int
- ninosE: int
- nombre: String
- documento: String
- edad: int
- correo: String
- celular: String
«Vista»
«Abstract»
VentanaUsuario
- pSuperior: JPanel
- pIzquierdo: JPanel
- pCentro: JPanel
interfaz 1
«Vista»
VentanaHuesped
- consultarDisponibilidad: JButton
- reservarHabitacion: JButton
- pagarInmediatamente: JButton
1 ventana
«Vista»
ConsultarDisponibilidad
- fecha1: JTextField
- fecha2: JTextField
- consultar: JButton
- scroll: JScrollPane
«Vista»
PagarInmediatamente
- id: JTextField
- pagar: JButton
- pasarelaP: JComboBox<String>
«Vista»
ReservarHabitacionHuesped
- inicio: JTextField
- fin: JTextField
- ninosEsp: JFormattedTextField
- adultosEsp: JFormattedTextField
- nombre: JTextField
- documento: JFormattedTextField
- correo: JTextField
- celular: JFormattedTextField
- edad: JFormattedTextField
- enviar: JButton
pCentro 1
«Vista»
VentanaAdministrador
- cargarHabitacion: JButton
- cargarCama: JButton
- cambiarTarifaHabitacion: JButton
- cargarHabitaciones: JButton
- cambiarTarifaServicio: JButton
- cargarMenuRestaurante: JButton
- configurarPlato: JButton
«Vista»
VentanaEstadisticas
- ventasProductoCantidad: JButton
- ventasProductoValor: JButton
- ventasHoras: JButton
- cantidadFacturas: JButton
- valorFacturas: JButton
- panelF: JPanel
- panelC: JPanel
- panelT: JPanel
- scroll: JScrollPane
«Vista»
CambiarTarifaHabitacion
- tipo: JComboBox<Tipo>
- tarifa: JFormattedTextField
- inicio: JTextField
- fin: JTextField
- diasButton: JRadioButton[]
- enviar: JButton
- mensajeFinal: JLabel
«Vista»
CambiarTarifaServicio
- tipo: JComboBox<Nombre>
- tarifa: JFormattedTextField
- enviar: JButton
- mensajeFinal: JLabel
«Vista»
CargarCama
- idHabitacion: JTextField
- tipoHabitacion: JComboBox<Tipo>
- tamaño: JTextField
- personas: JFormattedTextField
- nRos: JRadioButton
- enviar: JButton
- mensajeFinal: JLabel
«Vista»
CargarHabitacion
- id: JTextField
- ubicacion: JTextField
- tipo: JComboBox<Tipo>
- balcon: JRadioButton
- cocina: JRadioButton
- vistas: JRadioButton
- enviar: JButton
- mensajeFinal: JLabel
«Vista»
CargarHabitaciones
- rutaInventario: JTextField
- rutaCamas: JTextField
- enviar: JButton
- mensaje: JLabel
«Vista»
CargarMenuRestaurante
- rutaMenu: JTextField
- enviar: JButton
- menu: JPanel
- scroll: JScrollPane
«Vista»
ConfigurarPlato
- idProducto: JTextField
- nuevoPrecio: JFormattedTextField
- servicioCuarto: JRadioButton
- enviar: JButton
- mensaje: JLabel
- menu: JPanel
- scroll: JScrollPane
«Vista»
VentanaRecepcionista
- ConsultarInventario: JButton
- HacerReserva: JButton
- CancelarReserva: JButton
- HacerCheckIn: JButton
- HacerCheckOut: JButton
- VerOcupacionHotel: JButton
«Vista»
VentanaHabitacion
- consultarHabitacion: JButton
- consultarCamas: JButton
- consultarHotel: JButton
- infoHabitacion: String[]
- infoCamas: ArrayList<String[]>
- infoHotel: String[]
- titulo: JLabel
- panelF: JPanel
- scroll: JScrollPane
habitacion
«Vista»
ConsultarInventario
- rol: JComboBox<Tipo>
- enviarTipo: JButton
- scroll: JScrollPane
«Vista»
CancelarReserva
- idHabitacion: JTextField
- cancelar: JButton
«Vista»
HacerCheckIn
- idReserva: JFormattedTextField
- buscar: JButton
- continuar: JButton
- nombre: JTextField
- documento: JFormattedTextField
- edad: JFormattedTextField
«Vista»
HacerCheckOut
- idGrupo: JFormattedTextField
- buscar: JButton
- pasarelasP: JComboBox<String>
- pagar: JButton
«Vista»
HacerReserva
- inicio: JTextField
- fin: JTextField
- ninosEsper: JFormattedTextField
- adultosEsper: JFormattedTextField
- nombre: JTextField
- documento: JFormattedTextField
- correo: JTextField
- celular: JFormattedTextField
- edad: JFormattedTextField
- enviar: JButton
«Vista»
VerOcupacion
- ocupada: int[][]
«Vista»
VentanaMesero
- registrarConsumo: JButton
- idGrupo: JFormattedTextField
- documento: JTextField
- productos: JPanel
- casillas: JFormattedTextField[]
- scroll: JScrollPane
- registrarPago: JRadioButton
- enviar: JButton
- mensaje: JLabel
- cantidad: int
«Vista»
VentanaSpa
- registrarConsumo: JButton
- idGrupo: JFormattedTextField
- documento: JTextField
- registrarPago: JRadioButton
- enviar: JButton
- mensaje: JLabel
«Vista»
VentanaGuia
- registrarConsumo: JButton
- idGrupo: JFormattedTextField
- buscar: JButton
- huespedes: JPanel
- documentos: JRadioButton[]
- scroll: JScrollPane
- registrarPago: JRadioButton
- enviar: JButton
- mensaje: JLabel