Documentación Hotel APP Reservas

Grupo APE

Contenido

[Documentación del Diagrama UML 2](#_Toc188598275)

[Paquete Menu 2](#_Toc188598276)

[Clases principales del paquete: 2](#_Toc188598277)

[Paquete Connectores 2](#_Toc188598278)

[Clases principales del paquete: 2](#_Toc188598279)

[Paquete Model 2](#_Toc188598280)

[Clases principales del paquete: 2](#_Toc188598281)

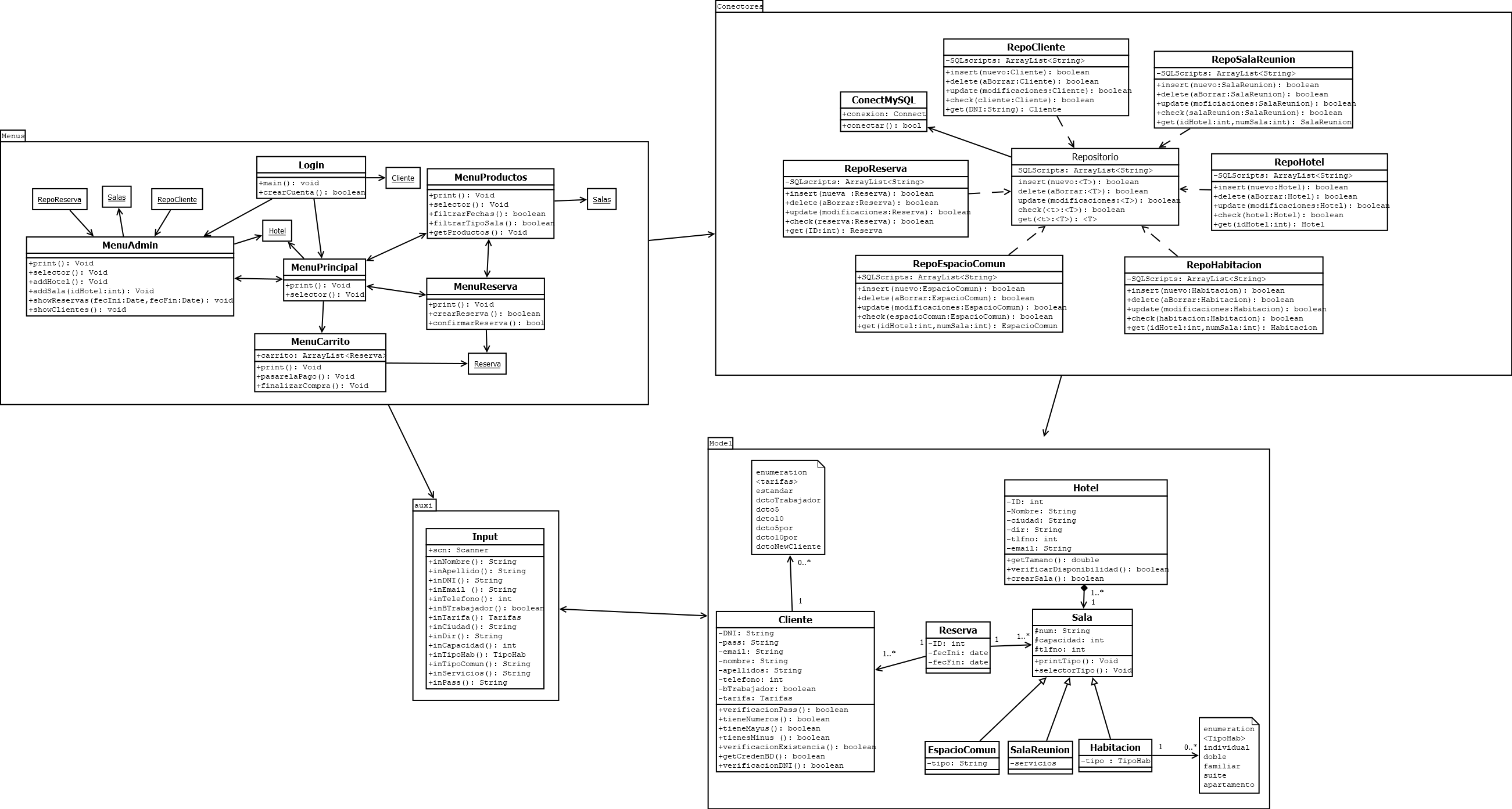
[Paquete Input 3](#_Toc188598282)

[Relaciones 3](#_Toc188598283)

[Conclusión 3](#_Toc188598284)

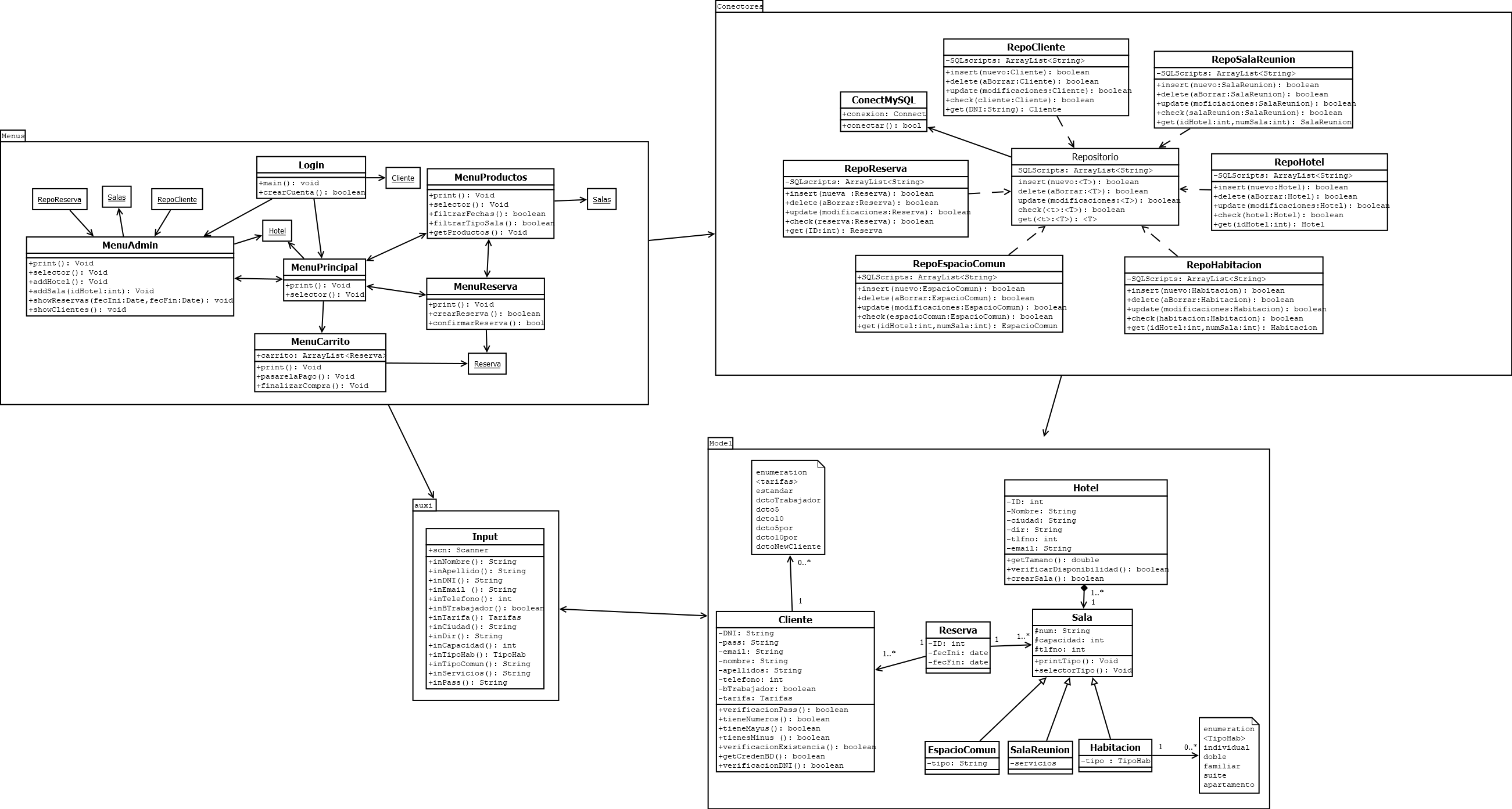
# Documentación del Diagrama UML

Este documento describe el diseño del sistema representado en el diagrama de clases UML proporcionado. Se detalla cada componente del sistema, incluyendo las clases, atributos, métodos y relaciones, con el objetivo de proporcionar una visión clara y completa del diseño.



# Paquete Menu

El paquete Menu contiene las clases responsables de gestionar las interacciones visuales del usuario con el sistema. Estas clases implementan menús específicos para administrar clientes, reservas, productos y otros elementos del sistema.



## Clases del paquete

### Login

Sirve para crear cuentas de cliente o trabajador y autentificar a los ya existentes.

#### Metodos

##### main() : void

Es la función principal de ejecución del programa en primera instancia. Esta función también contiene el control de acceso a la app y la conexión con la base de datos inicial.

##### crearCuenta() : boolean

Es una función que sirve para crear cuentas de cliente o trabajador en caso de no tener. Devolverá un boolean dependiendo de si la cuenta se crea de forma correcta o no.

### **MenuAdmin**

Proporciona funcionalidades administrativas, como agregar hoteles y salas, además de mostrar reservas y clientes.

#### Metodos

##### print() : void

Esta función contiene todo lo que tiene que ver con imprimir el menú de administración.

##### selector() : void

En esta función se contendrá el selector del menú donde se ejecutará la operación de switch para elegir la opción.

##### addHotel() : void

Este método permitirá a los administradores crear hoteles y añadir salas (al menos una).

##### addSala( idHotel:int ) : void

Este método servirá para añadir a un hotel ya existente salas, obligando siempre a añadir un tipo específico de sala.

##### showReservas( fecIni:Date , fecFin:Date ) : void

Este método permitirá mostrar la reserva de habitaciones de todos los hoteles entre unas fechas determinadas.

##### showClientes() : void

Este método mostrara todos los clientes y trabajadores que se encuentren en la base de datos.

### MenuPrincipal

Maneja la navegación principal del sistema entre los diferentes hoteles.

#### Metodos

##### print() : void

Esta función imprimirá por pantalla todos los hoteles y permitirá elegir entre ellos.

##### selector() : void

En esta función se contendrá el selector del menú donde se ejecutará la operación de switch para elegir la opción.

### MenuProductos

Muestra las habitaciones del hotel y su disponibilidad más cercana.

#### Metodos

##### print() : void

Esta función imprimirá por pantalla todas las habitaciones del hotel y la fecha mas cercana disponible a partir del dia de hoy y permitirá elegir entre ellos.

##### selector() : void

En esta función se contendrá el selector del menú donde se ejecutará la operación de switch para elegir la opción.

##### filtrarFechas() : boolean

Este método modifica la consulta para hacer que las habitaciones mostradas solo sean aquellas que cumplan las condiciones de inicio y fin. En caso de que se pueda devolverá true y en caso contrario false.

##### filtrarTipoSala() : boolean

Este método modifica la consulta para solo mostrar el tipo de sala querido. En caso de que se pueda devolverá true, en caso contrario false.

##### getProductos() : void

Este método recogerá de cada hotel cuales son las salas disponibles. Y lo devolverá en pantalla junto a su fecha de disponibilidad mas cercana.

### MenuReserva

Gestiona la creación y confirmación de reservas.

#### Metodos

##### print() : void

Este método imprime las características de la habitación a reservar.

##### crearReserva() : boolean

Pedira la información de la reserva y comprobara que está disponible en esas fechas. Devolvera false en caso de que no se pueda reservar en esas fechas.

##### confirmarReserva() : boolean

En caso que la reserva pueda ser creada y el cliente este de acuerdo, se grabara en la base de datos la información de la reserva. En caso de que por cualquier motivo no se pueda grabar devolverá false. Ademas, este método introducirá en la clase MenuCarrito.carrito la reserva.

### MenuCarrito

Gestiona el carrito de reservas y el proceso de pago. Y finaliza la ejecución del programa.

#### Atributos

##### +carrito:ArrayList<Reserva>

Este atributo guardará todas las reservas hechas por el cliente durante la ejecución de la app y permitirá mostrar toda la información sobre ellas. DEBE DE SER PUBLICO Y ESTATICO.

#### Metodos

##### print() : void

Imprimira toda la información del usuario, las reservas que haya seleccionado y el precio de las mismas individualmente y de forma total.

##### pasarelaPago() : boolean

En caso de tener una reserva en el carrito permitirá introducir los datos de pago y finalizar la compra. En caso de que de error devolver false.

##### finalizarCompra() : void

Una vez la compra ha sido exitosa saldrá del programa, en caso de que el usuario no tenga nada en el carrito o el pago sea infructuoso, eliminara la reserva hecha y/o saldrá del programa.

## Paquete Connectores

El paquete `Connectores` se encarga de la interacción con la base de datos mediante clases repositorio. Cada repositorio gestiona una entidad específica del sistema, proporcionando métodos para insertar, actualizar, eliminar y consultar datos.

### Clases principales del paquete:

- \*\*ConectMySQL:\*\* Clase responsable de gestionar la conexión a la base de datos.  
- \*\*RepoCliente, RepoHotel, RepoSala, RepoReserva, etc.:\*\* Proveen métodos para la manipulación de las entidades correspondientes en la base de datos.

## Paquete Model

El paquete `Model` contiene las clases que representan las entidades principales del sistema y sus relaciones. Estas clases incluyen atributos y métodos específicos para modelar la lógica del negocio.

### Clases principales del paquete:

- \*\*Cliente:\*\* Representa a los usuarios del sistema. Incluye atributos como DNI, nombre, email y métodos para validar datos y gestionar reservas.  
- \*\*Hotel:\*\* Representa a los hoteles del sistema, con atributos como ID, nombre, ciudad y métodos para la creación de salas.  
- \*\*Sala:\*\* Clase generalizada para modelar salas, que puede especializarse en subtipos:  
 - \*\*SalaReunion:\*\* Incluye atributos específicos como servicios.  
 - \*\*Habitacion:\*\* Incluye el atributo `tipo` para diferenciar entre individual, suite, etc.  
 - \*\*EspacioComun:\*\* Representa espacios compartidos, con atributos como tipo.

## Paquete Input

El paquete `Input` se encarga de gestionar la entrada de datos del usuario. La clase `Input` incluye métodos para capturar información como nombres, fechas, tarifas, tipos de habitaciones, entre otros.

## Relaciones

El sistema utiliza relaciones para modelar la interacción entre las diferentes clases:  
- \*\*Asociación:\*\* Por ejemplo, entre `Hotel` y `Sala`, donde un hotel puede tener múltiples salas.  
- \*\*Generalización:\*\* La clase `Sala` actúa como clase padre de `SalaReunion`, `Habitacion` y `EspacioComun`.  
- \*\*Dependencia:\*\* Clases como `MenuAdmin` dependen de los repositorios para acceder a la base de datos.

## Conclusión

El diseño representado en el diagrama UML permite gestionar de manera eficiente las funcionalidades del sistema, asegurando una clara separación de responsabilidades entre los componentes. Este documento proporciona una guía detallada para entender la estructura y lógica del sistema.