Universidad de los Andes

15 de marzo de 2023

Diseño y Programación Orientada a Objetos

Milton Andrés Mesa Manrique

Proyecto 1 Entrega 2

Documento de Diseño para un Sistema de Administración de Propiedades (PMS)

Se desea realizar una aplicación que se encargue de administrar la información de un Hotel. Antes de empezar con el proceso de diseño, vale la pena volver a enunciar las funcionalidades de alto nivel que el sistema debe estar en capacidad de resolver. Utilizamos como insumo el documento de análisis hecho para la entrega 1.

De este modo, aseguramos que el sistema sea capaz de funcionar adecuadamente sin importar la ruta de implementación ni las decisiones de diseño específicas que se tomen a posteriori.

I. NIVEL 1: SISTEMA COMPLETO:

i. Requerimientos funcionales

Usuario	ID	Requerimiento Funcional	Nota
Administrador	RF1	Iniciar Sesión	
	RF2	Crear habitaciones	Habitaciones tipo estándar, suite y doble suite. Algunas camas son solo para niños.
	RF3	Cargar archivo con información de habitaciones	
	RF4	Cargar tarifas por tipo de cuarto	Si hay duplicados de fecha y tipo, se toma la menor tarifa. Avisa si hay fechas vacías en los próximos 365 días.
	RF5	Cargar menús del restaurante	
	RF6	Cambiar tarifa de los servicios que ofrece el hotel	
Recepcionista	RF7	Iniciar Sesión	
	RF8	Registrar pago hecho	
	RF9	Realizar reserva a nombre de huésped	Las habitaciones seleccionadas se bloquean. Menores de 2 años no necesitan cama.
	RF10	Cancelar reserva a huésped	Las reservas no pueden cancelarse si faltan 48 horas.
	RF11	Registro o *checkout* de un huésped	Un grupo debe registrarse y hacer checkout a tiempo, después de pagar todo
	RF12	Generar facturas	
Empleado	RF13	Iniciar Sesión	
	RF14	Consultar inventario	
	RF15	Registrar servicios consumidos	

ii. Aspectos técnicos y Restricciones

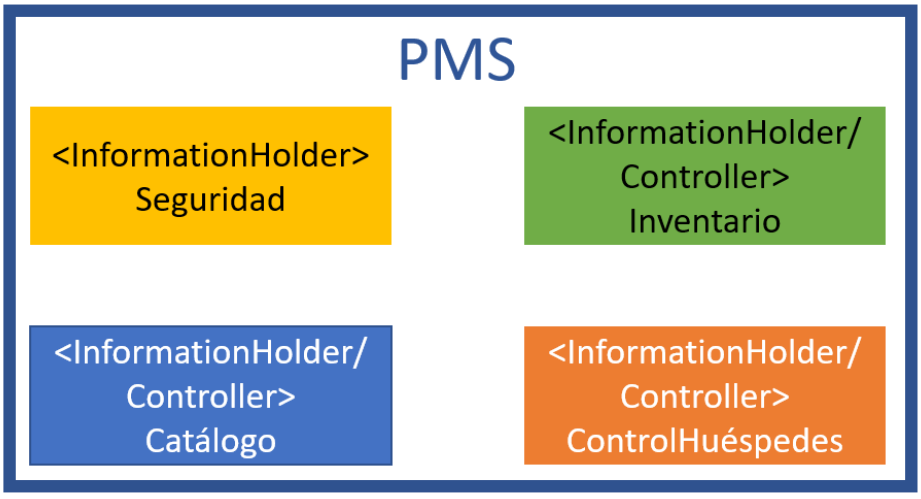
ID	Restricción
R1	Persistencia en archivos
R2	Segregación de usuarios por tipo
R3	Generar historial de un grupo de huéspedes
R4	Interfaz basada en consola

II. NIVEL 2: PRIMERA DESCOMPOSICIÓN:

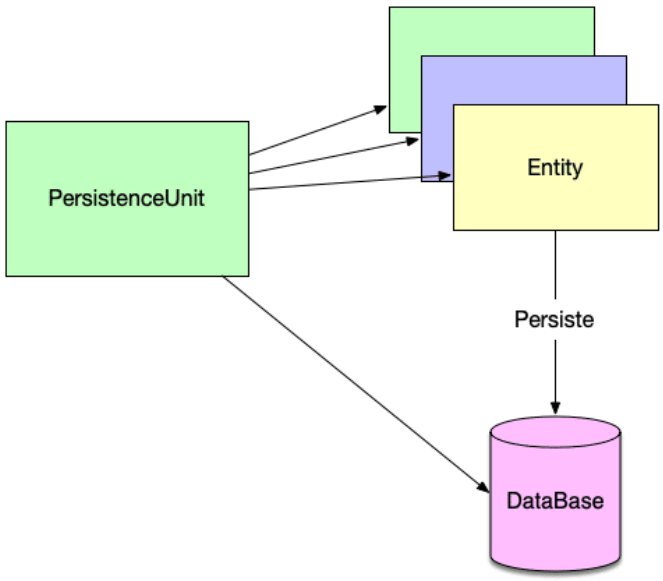
i. Componentes candidatos

Componente	Estereotipo	Descripción	RF Asociados
Seguridad	Information Holder	Mantiene la información de las credenciales de los usuarios y les da acceso especializado según el tipo de usuario. Le avisa a la consola el resultado para que pueda funcionar normalmente. La información de estas credenciales se carga por defecto al iniciar la aplicación.	1, 7, 13
Inventario	Information Holder/Controller	Mantiene la información pertinente del estado de las habitaciones. Permite crear nuevas habitaciones y cargar los datos del mismo. Así mismo, cambiar el estado de una habitación cuando sea necesario, como al hacer reserva o checkout.	2, 3, 4, 8, 9, 10, 11, 12, 14
Cátalogo	Information Holder/Controller	Mantiene la información de todos los servicios que ofrece el hotel incluyendo el restaurante. Permite cambiar las tarifas de estos servicios.	5, 6, 15
ControlHuespedes	Information Holder/Controller	Es un componente que guarda la información del estado actual de todos los huéspedes del hotel. Para cada huésped se tiene la información personal, la habitación reservada, el grupo al que pertenece, los pagos realizados y el saldo pendiente. Dentro de este componente también se encuentra el registro de los servicios consumidos por todos los huéspedes.	8, 9, 10, 11, 12, 15

La figura mostrada a continuación muestra los 4 componentes candidatos y sus respectivos estereotipos.



Adicionalmente, vamos a contar con un 5 componente que, aunque no hace parte del PMS, si funciona de forma paralela al mundo del problema, como la consola. La persistencia nos va a permitir llevar un respaldo del estado de ejecución del programa. En caso de que falle nuestra aplicación, podremos recuperar el estado previo haciendo la carga de datos de la persistencia. Este componente se incluirá de forma breve dentro del diseño. Nos basta con saber que funcionara de la siguiente manera. Es una entidad que se conecta de forma transversal a las demás componentes, guardando el estado de ejecución de las clases en una carpeta de archivos .txt y es capaz de cargar la información de vuelta a las clases en caso de ser necesario.



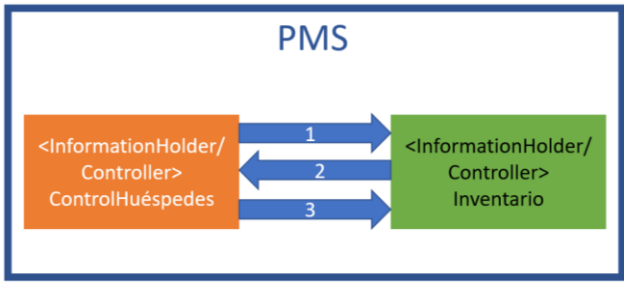
ii. Responsabilidades:

#	Responsabilidad	Componente
1	Conocer las credenciales aceptadas para cada usuario en el sistema.	**Seguridad**
2	Aceptar o denegar acceso a la aplicación, dando el tipo de usuario que es.	
3	Conocer el estado actual de las habitaciones del hotel.	**Inventario**
4	Crear nuevas habitaciones.	
5	Cargar datos de las habitaciones a partir de un archivo	
6	Cambiar el estado de una habitación según reserva, cancelación, registro o checkout de un huésped.	
7	Extraer información de una habitación para generar una factura.	
	Cambiar el saldo asignado a una habitación según el consumo de un huésped.	
8	Cargar las tarifas por tipo de cuarto en una fecha específica.	
9	Cargar los menús del restaurante.	**Catálogo**
10	Conocer el nombre y valor de todos los servicios que ofrece el hotel, incluyendo el restaurante y tarifas de habitaciones.	
11	Permite cargar o cambiar los precios de los servicios que ofrece el hotel.	
12	Registrar que un pago se hizo a nombre de un cliente.	**ControlHuéspedes**
13	Realizar una reserva a nombre de un huésped.	
14	Cancelar reserva a un huésped.	
15	Realizar registro o checkout de un huésped.	
16	Generar factura a nombre de un huésped, habitación o grupo.	
17	Hacer registro de un servicio consumido a nombre de un cliente, habitación o grupo.	

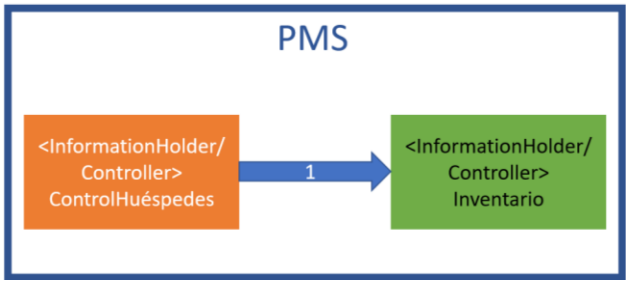
iii. Colaboraciones:

A continuación, vamos a detallar el proceso que deben seguir los requerimientos funcionales que involucren la colaboración entre dos o más componentes. Dejamos por fuera los requerimientos funcionales que solo requieran un componente.

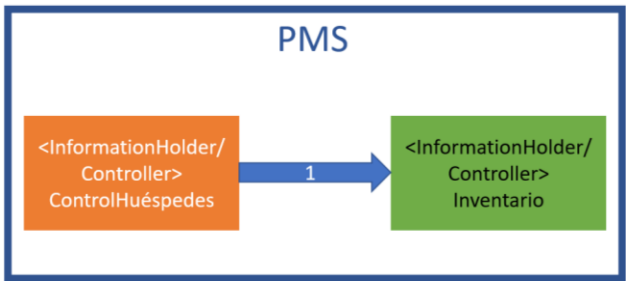
- Realizar reserva a nombre de un huésped:
    1. **ControlHuespedes** le indica al **Inventario** que busque una habitación con los atributos buscados como capacidad y tipo.
    2. **ControlHuespedes** retoma la información de esa habitación, dada por el **Inventario**, y crea un subtotal con el costo de la habitación las noches seleccionadas.
    3. **ControlHuespedes** le pide al **Inventario** que bloquee la habitación para que no pueda ser reservada por otra persona, hasta que el huésped actual cancele la reserva o libere la habitación.



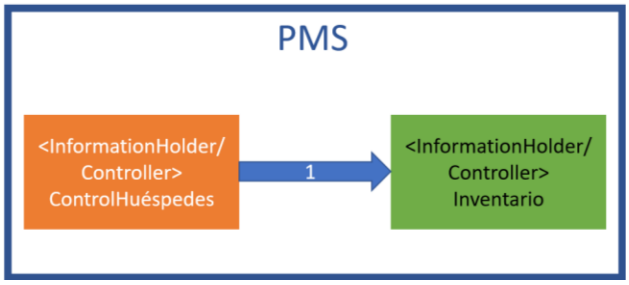
- Cancelar reserva a nombre de un huésped:
    1. **ControlHuespedes** revisa la fecha de la reserva. Si faltan más de 48 horas para la fecha reservada, se puede cancelar. Comprobado esto le indica a **Inventario** que libere la habitación.



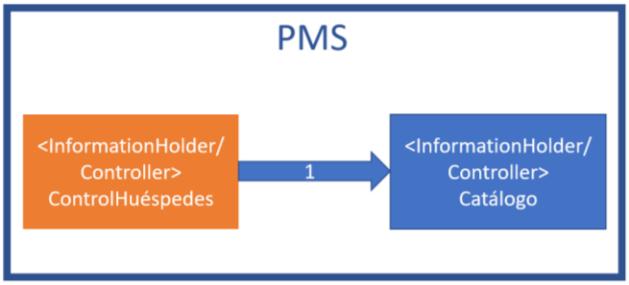
- Registro de un huésped:
    1. **ControlHuespedes** crea a un nuevo huésped con toda la información necesaria. Modifica en el **Inventario** el estado de la habitación donde se va a alojar el huésped. Repite el mismo proceso para cada uno de los miembros del grupo del huésped original.



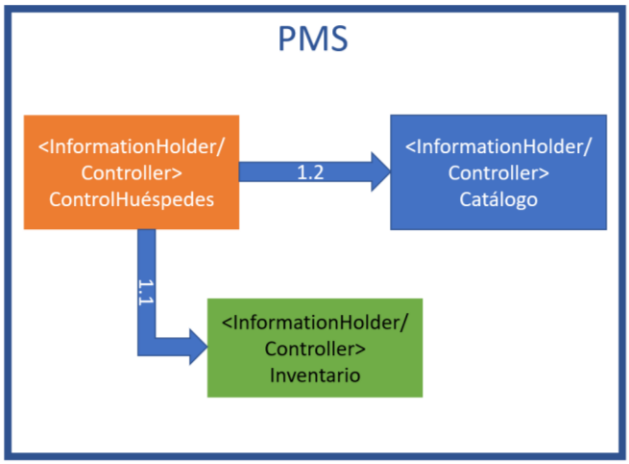
- Checkout de un huésped:
    1. **ControlHuespedes** cambia el estado del huésped creado y lo invalida. Repite el proceso para los demás huéspedes que estén hospedados en la misma habitación. Luego de esto, le indica a **Inventario** que libere el estado de la habitación para que pueda ser reservada por otra persona.



- Generar facturas de los consumos realizados:
    1. **ControlHuespedes** extrae el consumo total del huésped, grupo o habitación, según sea requerido por la consola. Luego debe validar cada uno de los precios para que estén en concordancia con los precios del **Catálogo**.



- Registrar servicio consumido:
    1. **ControlHuespedes** inicia el registro de un consumo de un producto, por un valor, y a nombre de un cliente, grupo o habitación según sea el caso. En el caso de que servicio de alojamiento, debe verificar el precio de las tarifas que están en **Inventario**.
    2. En caso de que el servicio sea otro, debe verificar el precio de las tarifas que hay en **Catálogo**.



III. Nivel 3: Segunda descomposición:

Con los componentes candidatos identificamos continuaremos el proceso de descomposición en clases con atributos y métodos que nos permitan dar resolución de los requerimientos funcionales.

i. Seguridad:

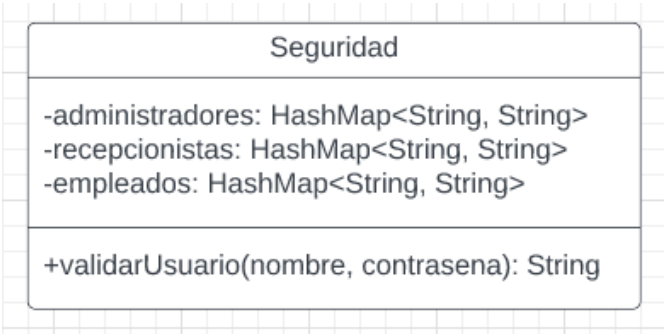
Recordando las responsabilidades asociadas a este componente, tenemos:

- Conocer las credenciales aceptadas para cada usuario en el sistema:
- Aceptar o denegar acceso a la aplicación, dando el tipo de usuario que es.

Las responsabilidades asignadas a este componente de alto nivel fueron de relativa sencillez. Conocer las credenciales se puede modelar por medio de HashMaps<usuario, contrasena> que nos permitirán cumplir con el objetivo de almacenar la información y dividir los tipos de usuario.

La responsabilidad de aceptar o denegar el acceso a la información, la podemos modelar con un método que haga la validación del usuario y retorne un String con el tipo de usuario que logro ingresar al sistema, en caso de que sea aprobado, o un mensaje de error.

Por estas razones, este componente fue modelado como una sola clase que tenga los atributos y el método mencionado anteriormente.



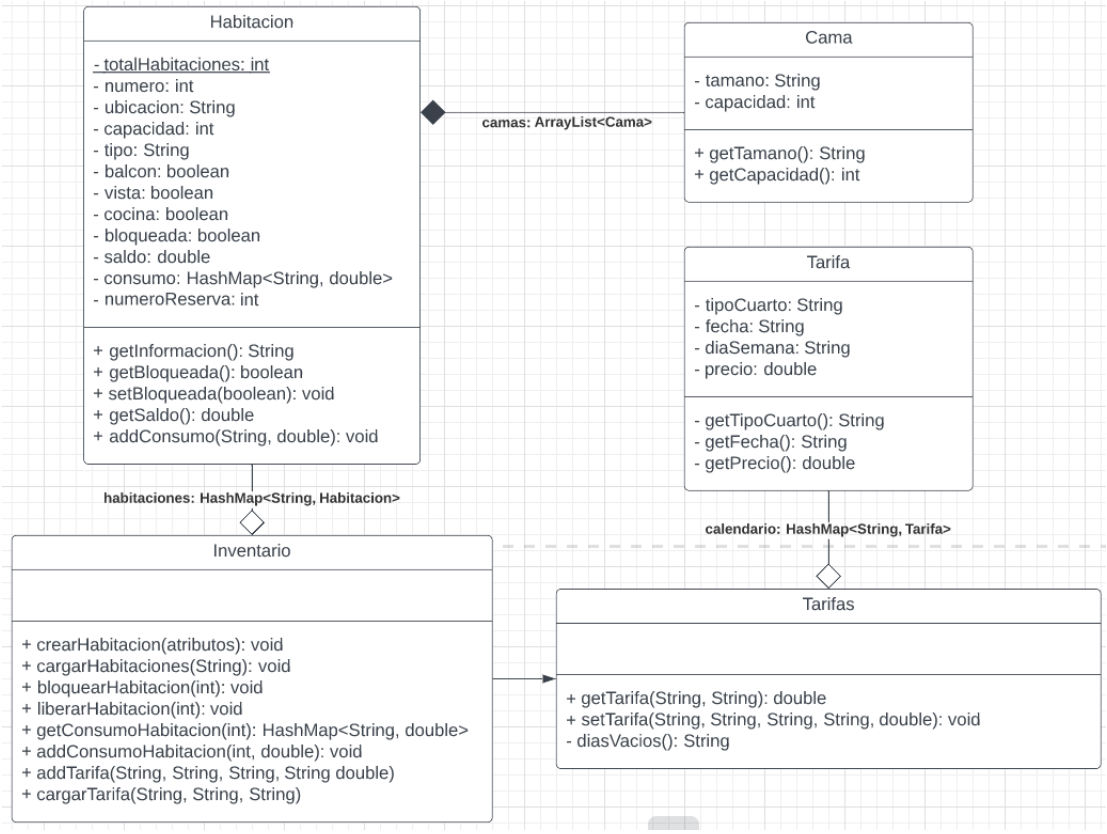
Dado que el componente consiste únicamente de una clase, no cuenta con colaboraciones con el resto del sistema (salvo con la consola, para verificación de usuarios).

ii. Inventario:

Recordando las responsabilidades asociadas a este componente, tenemos:

- Conocer el estado actual de las habitaciones del hotel.
- Crear nuevas habitaciones.
- Cargar datos de las habitaciones a partir de un archivo.
- Cambiar el estado de una habitación según reserva, cancelación, registro o checkout de un huésped.
- Extraer información de una habitación para generar una factura.
- Cambiar el saldo asignado a una habitación según el consumo de un huésped.
- Cargar las tarifas por tipo de cuarto en una fecha específica.

Para el desarrollo de estas responsabilidades se descompuso el componente Inventario en 4 clases. Estas clases garantizan que haya un correcto acceso a las habitaciones del hotel y se puedan gestionar de manera efectiva, teniendo en cuenta las tarifas de las habitaciones y las características de las habitaciones, así como su estado actual, pues es posible saber si una habitación está reservada, ocupada o libre. Encontramos múltiples relaciones de agregación y composición, debido a que uno de los principales motivos para desagregar este componente fue el de distribuir la carga de la información en varias estructuras de datos como ArrayLists y HashMaps que permitan que las responsabilidades estén menos concentradas en un solo punto, y se puedan distribuir en varias clases.



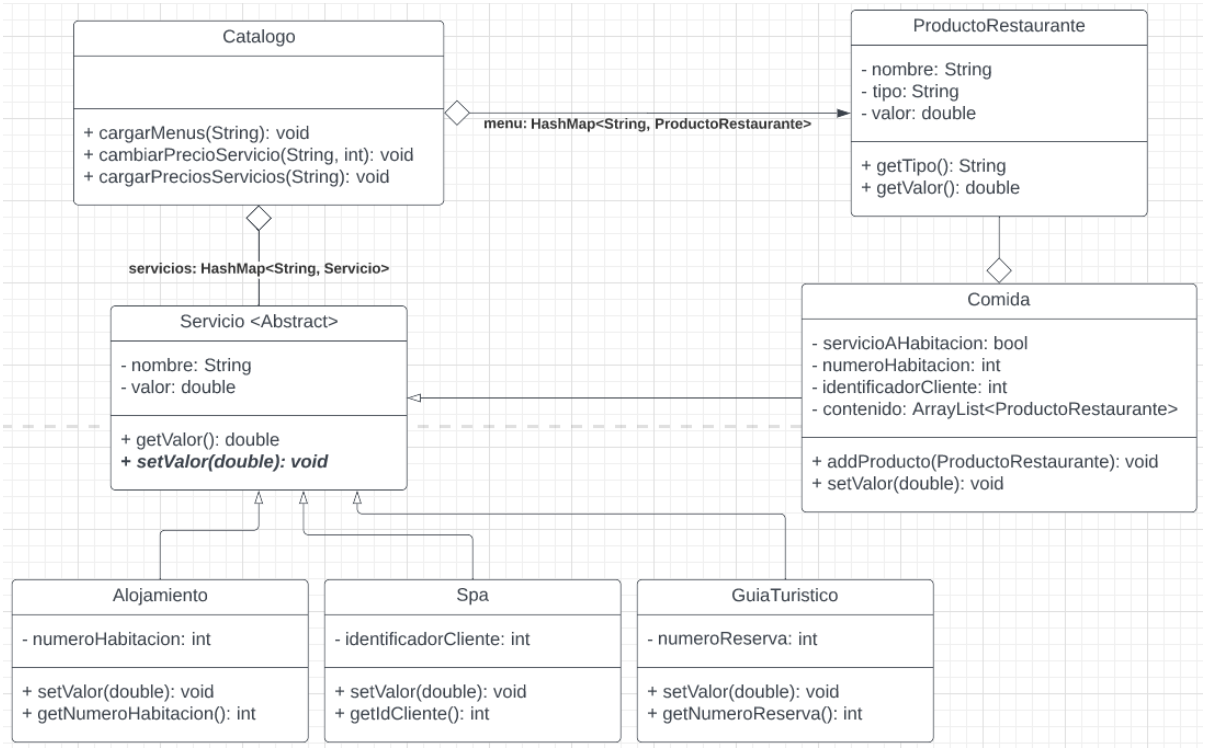
Las colaboraciones se pueden apreciar con facilidad en la anterior imagen. Contamos con un estilo de control centralizado donde Inventario es la clase principal que mediante múltiples métodos controla el estado y comportamiento de las otras clases, que son por lo general Information Holders.

iii. Catálogo:

Nuevamente, vemos las responsabilidades que asociamos a este componente:

- Cargar los menús del restaurante.
- Conocer el nombre y valor de todos los servicios que ofrece el hotel, incluyendo el restaurante.
- Permite cargar o cambiar los precios de los servicios que ofrece el hotel.

Para este componente se buscó generar una abstracción de Servicio para que a partir de allí podamos diseñar varias clases que hereden y representen los diferentes tipos de Servicios que tiene el hotel, que tienen un comportamiento marcadamente diferente entre sí, especialmente en lo relativo a la facturación y el cálculo del valor total. Del mismo modo encontramos relaciones de agregación que nos permiten almacenar datos del estado actual del problema de una forma más organizada.



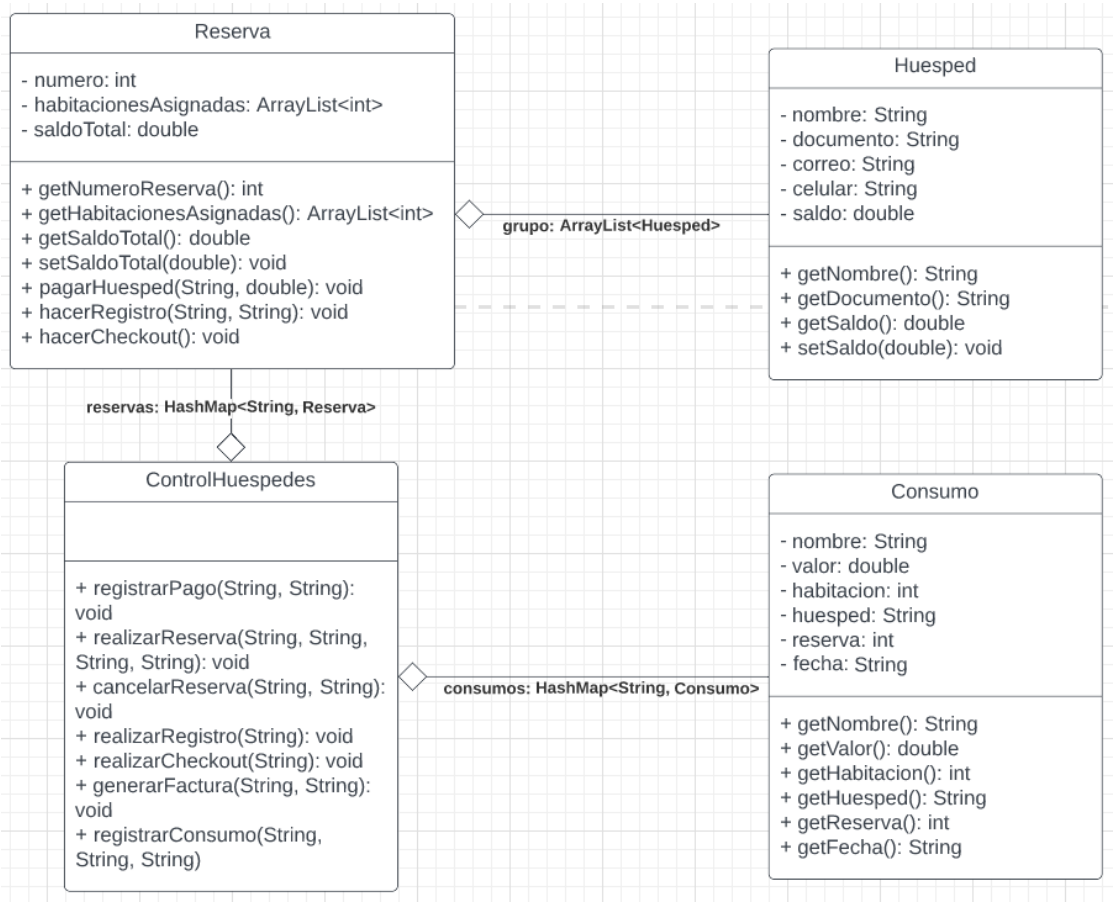
Respecto a las colaboraciones que presenta el diseno presente, es posible observar que la clase Catalogo funciona como un controlador que ejerce un tipo de poder centralizado, ya que esta clase controla los comportamientos de los Servicios y la Comida a través de relaciones de agregación, que modulan el comportamiento.

iv. ControlHuéspedes:

Con este último componente, retomamos las responsabilidades asociadas también:

- Registrar que un pago se hizo a nombre de un cliente.
- Realizar una reserva a nombre de un huésped.
- Cancelar reserva a un huésped.
- Realizar registro o checkout de un huésped.
- Generar factura a nombre de un huésped, habitación o grupo.
- Hacer registro de un servicio consumido a nombre de un cliente, habitación o grupo.

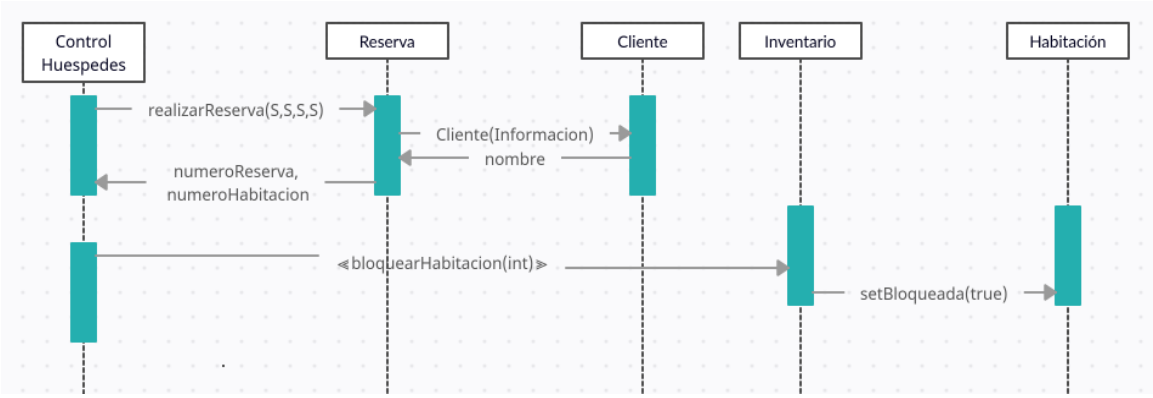
Para este ultimo componente se siguió un proceso similar a los anteriores, donde a partir de una clase central llamada ControlHuespedes se controlan las responsabilidades de todo aquello pertinente a las reservaciones, y un registro de los consumos que hacen los huéspedes en el hotel, con información de alto detalle.



En este caso nos encontramos del mismo modo con un estilo de control centralizado, donde una clase principal regula el comportamiento y los valores de las clases aledañas.

DIAGRAMA DE SECUENCIA DE FUNCIONALIDAD PRINCIPAL: HACER RESERVA

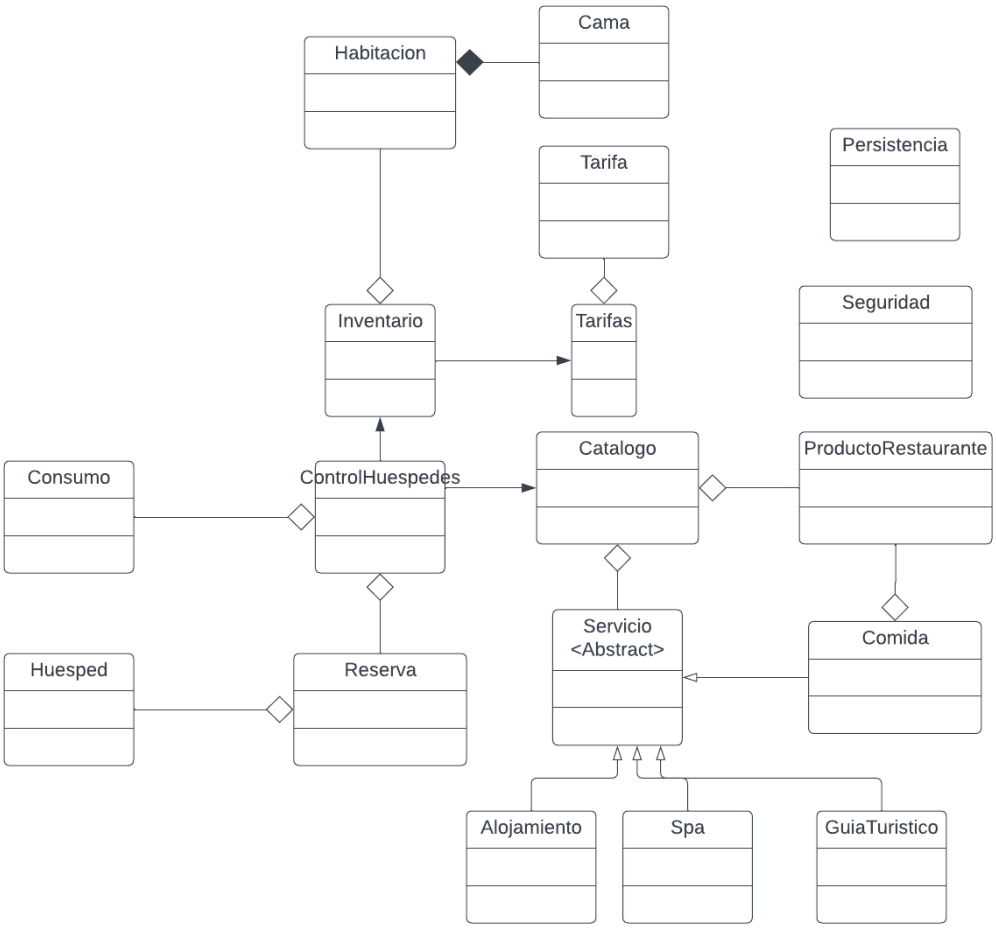
Para hacer una reserva se debe seguir el siguiente flujo de funciones:



IV. Nivel 4: Descomposición final:

En este punto ya hemos identificado los componentes específicos que harán parte de nuestra solución, así como sus atributos, métodos y relaciones. De este modo, se obtiene el diseño final que nos permitirá proceder a la implementación de nuestra solución. Ahora, presentamos el diagrama de clases final del diseño>

Y una forma abreviada del mismo, que nos permite hacer énfasis en la manera en que las colaboraciones se dan entre clases.


Habitacion
Cama
Tarifa
Persistencia
Inventario
Tarifas
Seguridad
Consumo
ControlHuespedes
Catalogo
ProductoRestaurante
Servicio
<Abstract>
Comida
Huesped
Reserva
Alojamiento
Spa
GuiaTuristico