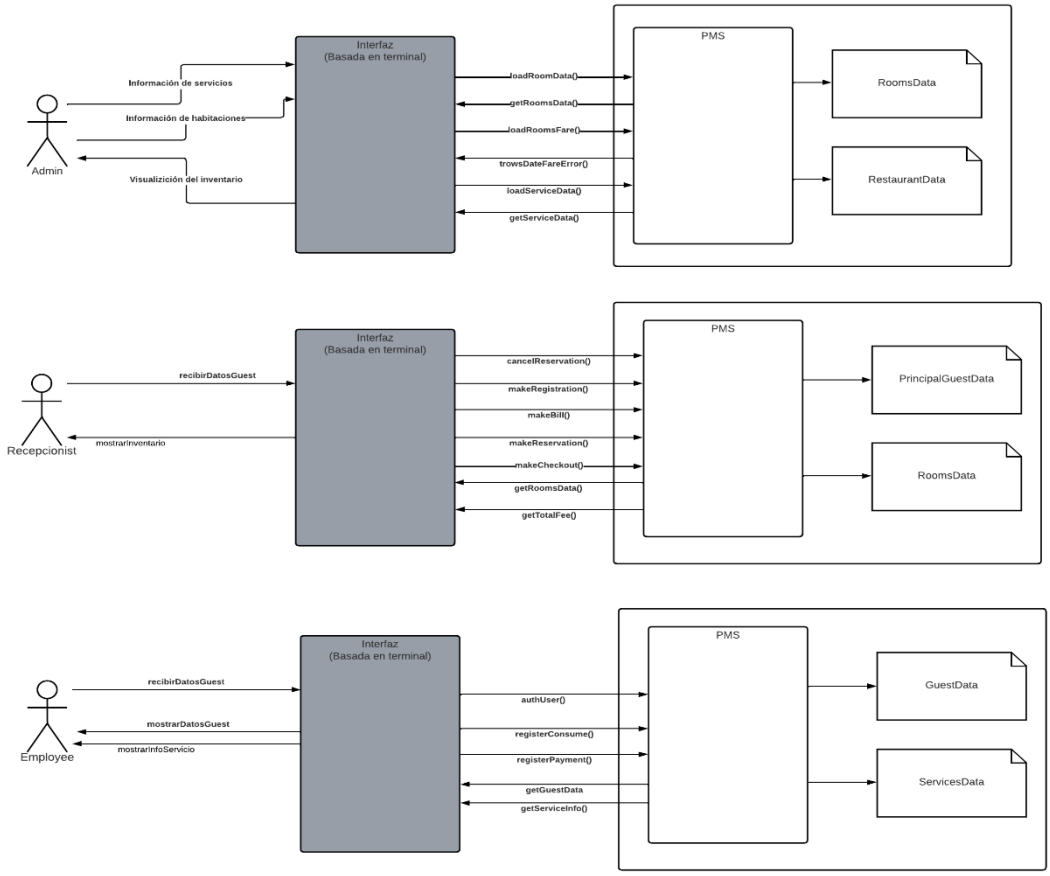
Documentación Property Management System (PMS)

Tabla de contenido

1 Contexto

A continuación, se definen las funcionalidades de alto nivel que la aplicación debe tener para poder satisfacer al usuario independientemente de la interfaz que se decida implementar, es decir, la aplicación no se verá afectada por la interfaz que se use.



En cuanto a la interacción con la interfaz, cada usuario tendrá una opción en común: el inicio de sesión. El resto de las interacciones serán acorde a dicho usuario, por ejemplo, el administrador podrá cargar un archivo con la información de las habitaciones, o añadir una manualmente. Respecto al recepcionista, una de sus interacciones será realizar una reserva, o registrar a los invitados. Finalmente, el último usuario, el empleado, podrá hacer los respectivos cargos de su servicio.

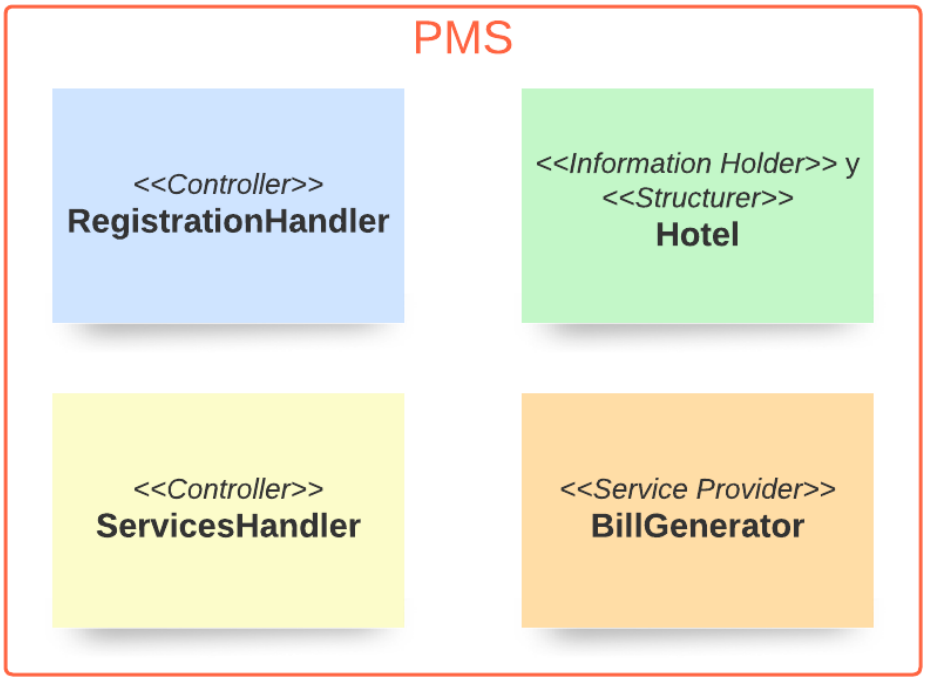
A su vez, la interfaz deberá mostrar una información específica a cada usuario, a excepción del inicio de sesión. Por ejemplo, tanto el administrador como el recepcionista van a poder ver en su interfaz el inventario que hay en el hotel, pero el recepcionista va a ser capaz de ver el informe de la tarifa total para las noches seleccionadas por un grupo de huéspedes. Mientras que, el empleado va a poder ver en su interfaz la información de los servicios y la información de los usuarios registrados en el hotel que pueden hacer consumos de los servicios del hotel. En pocas palabras, cada usuario va

a tener una interacción con la interfaz diferente, ya que cada uno cumple funciones diferentes dentro del sistema.

2 Nivel 1

2.1 Macro roles y estereotipos

1. Se requiere controlar las actividades del hotel, como la gestión de habitaciones (check-in y check-out, reservas, etc.) y también el manejo de los servicios ofrecidos dentro del hotel a los huéspedes, por lo que identificamos dos roles de *Controller*, uno para cada actividad del hotel, puesto que dichas actividades poseen grandes diferencias que hacen necesario dividir la lógica de estas en diferentes roles, estos roles serán *RegistrationHandler* y *ServicesHandler* respectivamente.
2. Así mismo se identifica que debe existir un rol que se encargue de orquestar las relaciones entre diferentes partes de la aplicación, por lo que se define a Hotel como *Structurer*.
3. Puesto que se debe manejar datos de habitaciones y tarifas, así como del servicio de restaurante, se decide destacar un rol encargado de la contenencia de toda esta información. El rol Hotel deberá contener también el estereotipo *Information holder* con el cual tendrá responsabilidades relacionadas al manejo de la información del Hotel. Se decide manejar únicamente un rol para ambos estereotipos pues se considera pertinente mantener centralizadas estas dos funciones.
4. Finalmente se modela el servicio de generación de facturas, este será el rol *billGenerator* y tendrá el estereotipo de *Service Provider*, se modela a este rol con este estereotipo ya que tendrá una funcionalidad específica, y será utilizado por otros componentes



2.2 Responsabilidades

Continuando con las tareas que se realizarán, se toman en cuenta las responsabilidades que cada uno de los componentes tendrá asociada. Por lo tanto, en la siguiente tabla se muestran cada una de las responsabilidades que asume cada componente declarado anteriormente

Tabla 1: Asignación de responsabilidades

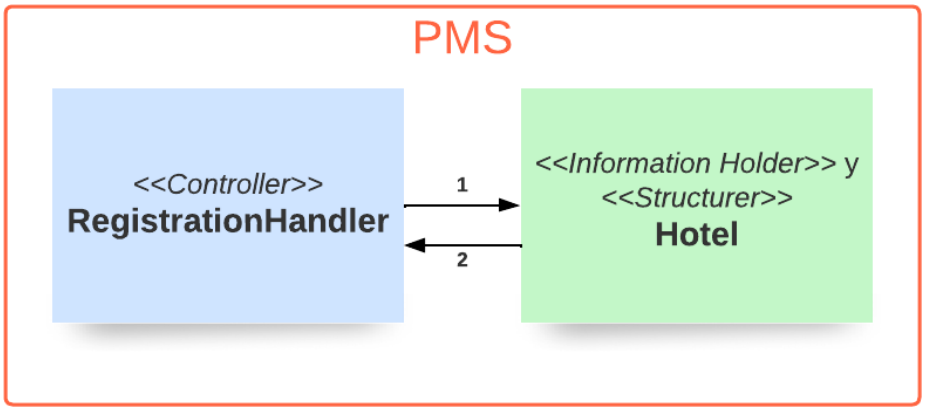
#	Responsabilidad	Componente
1	Crear una reserva	RegistrationHandler
2	Hacer Check-in de un huésped y/o su grupo	
3	Hacer Check-out de un huésped y/o su grupo	
4	Cancelar una reserva	
5	Almacenar habitaciones	Hotel
6	Almacenar información de servicios	
7	Almacenar información reservas	
8	Almacenar información registros	
9	Cargar información servicios	
10	Cargar información habitaciones	
11	Generación de facturas de servicios	BillGenerator
12	Generación de factura de cobro por habitación	
13	Informar tarifa total para las noches seleccionadas	
14	Registrar consumo	ServicesHandler
15	Registrar pagos (cargar a la habitación o pagar en el momento)	

2.3 Colaboraciones

Continuando, se definen las colaboraciones que hay entre los distintos componentes ya definidos.

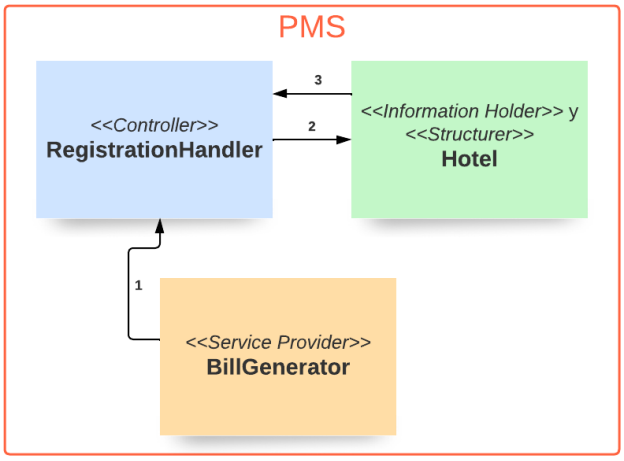
- **Hacer Check-in:** el controlador *RegistrationHandler* al recibir los datos del nuevo registro por interfaz:

    1. Le indica a *Hotel* que hay que almacenar información del usuario sobre el nuevo registro, y debe modificar la información de las habitaciones que se encuentra en *Hotel*
    2. *Hotel* debe entregar información sobre la reserva que hizo ese usuario, además de guardar la información del huésped principal y/o su grupo



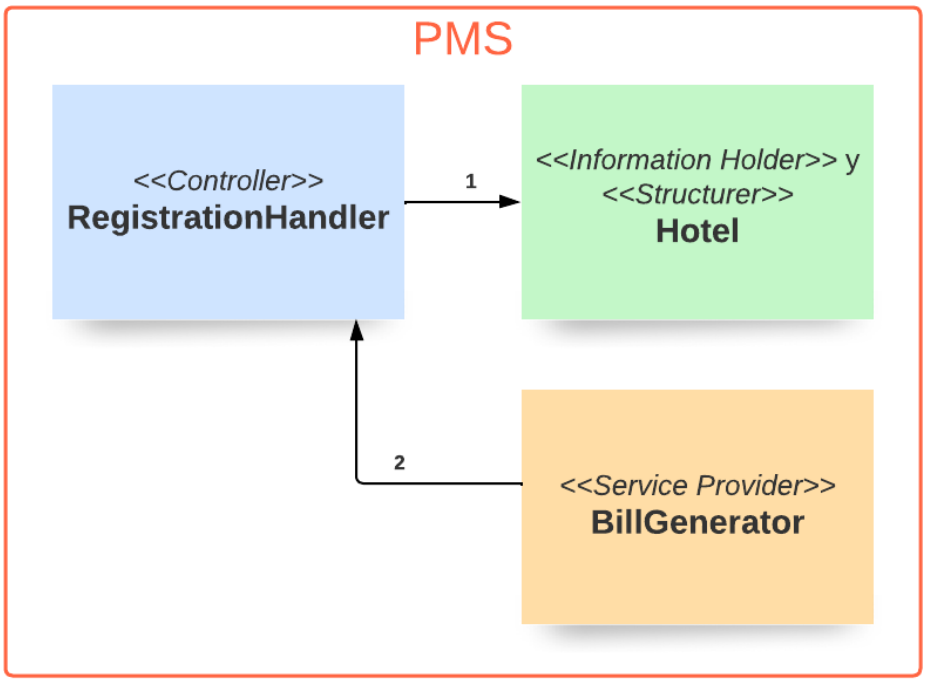
- **Reservar habitaciones:** el controlador *RegistrationHandler* al recibir los datos de la nueva reserva:

    1. Debe pedirle al service provider *BillGenerator que calcule* e informe a *RegistrationHandler* sobre la tarifa total de las noches seleccionadas en la reserva.
    2. Le indica a *Hotel* que hay que almacenar información sobre la nueva reserva y modificar información de disponibilidad de las habitaciones.
    3. *Hotel* debe entregarle información sobre las habitaciones disponibles sobre las cuales el huésped puede hacer la reserva.



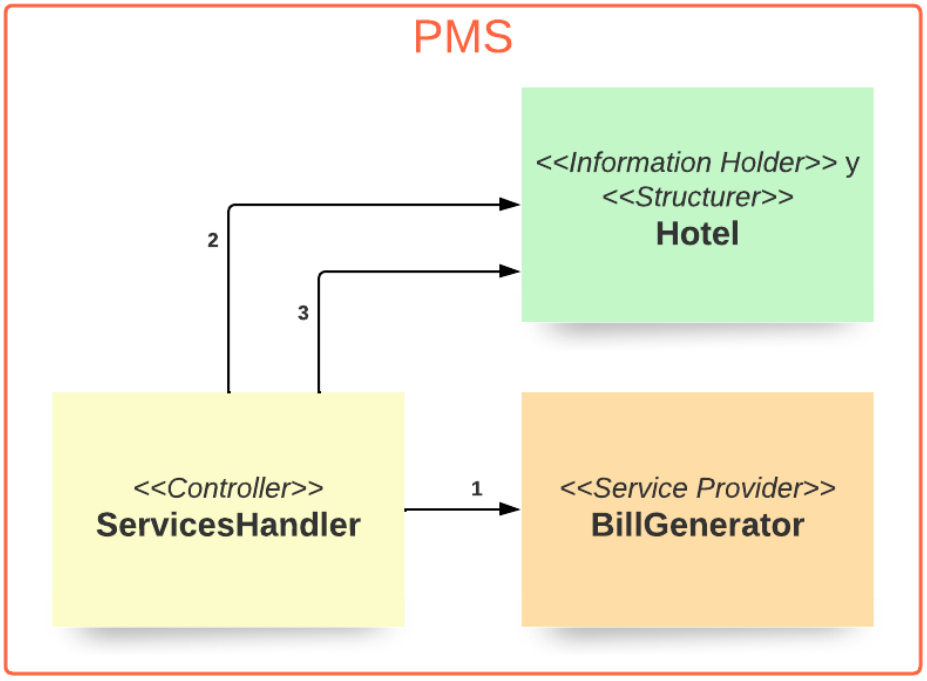
- **Hacer check-out:** El cliente se dirige a la recepción a pagar el valor de su reserva y los servicios consumidos, con el fin de salir del hotel. Para ello, el controlador *RegistrationHandler:*

    1. Le indica al *Hotel* que almacene la nueva información sobre la habitación que acaba de quedar disponible.
    2. Le pregunta a *BillGenerator* la factura de cobro por la habitación, igualmente también le pregunta por la generación de la factura de los servicios consumidos. Todos los servicios que se consumieron aparecen en la factura, así ya hayan sido pagados, sin embargo, si ya se pagaron, no se vuelven a pagar en el check-out.

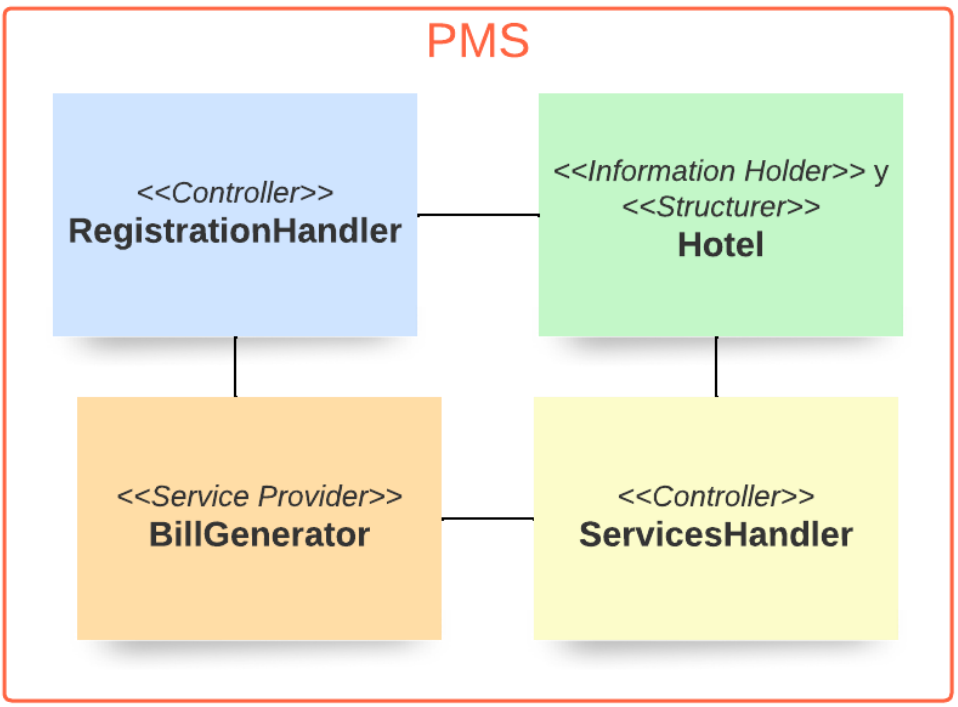


- **Consumir un servicio del hotel:** El controlador *ServicesHandler*, debe encargarse de registrar los consumos de un huésped dentro del hotel, por lo que:
    1. Le indica a *BillGenerator* que vaya actualizando la factura de los servicios consumidos. Se debe tener en cuenta si el servicio consumido se paga en el momento o se carga a la habitación del huésped para pagar al hacer check-out.

2. Le indica al hotel que le entregue la información disponible sobre los servicios que da el hotel. Cabe aclarar que la información de los servicios contiene: El menú del restaurante, el tipo de pago del servicio y el costo del mismo.
3. Le pregunta al Hotel si el huésped que quiere hacer consumo del servicio ya está registrado o no en el hotel. Si no hay registro del huésped, no se puede hacer consumo del servicio.



En las colaboraciones que hay entre componentes, es evidente que existe un estilo de control delegado, ya que el controlador *RegistrationHander* se encarga de hacer la lógica para las funcionalidades fundamentales como registrar huéspedes o reservar habitaciones. Sin embargo, se apoya de los componentes BillGenerator y servicesHandler para hacer las funcionalidades adicionales que implica hacer un registro en la aplicación, como generar facturas, o llevar a cabo cada uno de los consumos que hace el huésped registrado. Igualmente, se puede evidenciar que las colaboraciones son fuertemente conectadas, ya que para que pueda haber una reserva, se debe conocer el inventario que contiene el hotel, así mismo, para que pueda generarse una factura, se debe conocer la información de los servicios y los registros que guarda hotel.

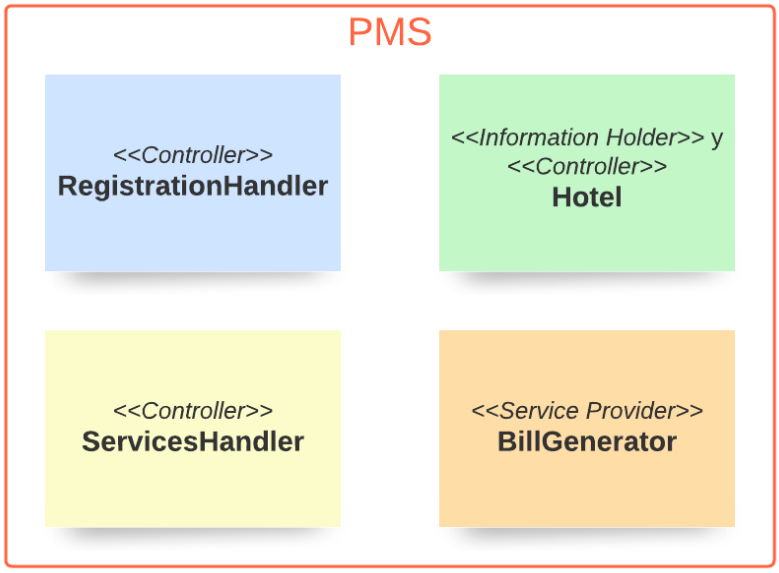


3 Nivel 2

Como parte del proceso iterativo, se toma la decisión de cambiar uno de los estereotipos de Hotel, esto debido a que encontramos responsabilidades que habían sido olvidadas en la primera iteración, adicional a eso, para la responsabilidad de *Informar tarifa total para las noches seleccionadas* se hace un cambio respecto al rol que la ejecutará, pues se considera óptimo que el que la maneje sea *registrationHandler*. Las responsabilidades nuevas que se le asignan a Hotel son las siguientes:

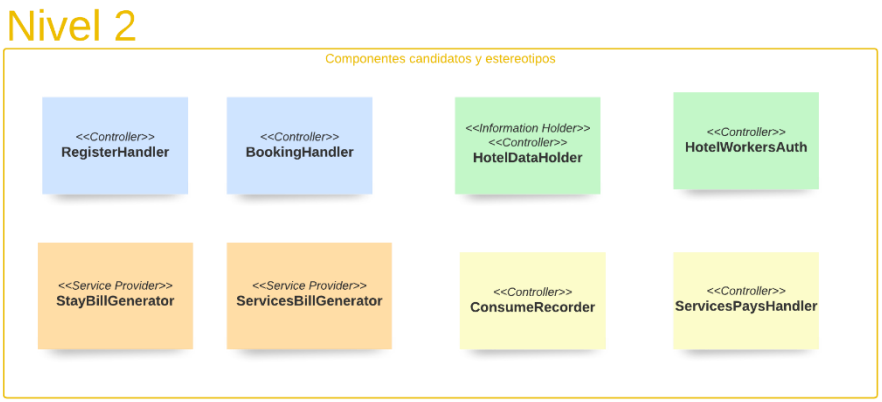
#	Responsabilidad	Componente
1	Autenticación de empleados	Hotel
2	Generar archivos log con el historial de huéspedes	
3	Informar tarifa total para las noches seleccionadas	RegistrationHandler

Con este cambio identificado, el rol *Hotel* queda de la siguiente manera:



3.1 Componentes candidatos y estereotipos

A partir de los roles identificados en la iteración anterior, estos roles se disgregan en subcomponentes más pequeños que sean acordes a sus roles padres pero que tengan un nivel mayor de especificidad.



3.2 Responsabilidades

#	**Responsabilidad**	**Componente**
1	Verificar disponibilidad de habitaciones	*BookingHandler*
2	Generar y guardar reserva en el hotel	
3	Cancelar reserva	
4	Informar tarifa de la reserva	
5	Cambiar información del uso de la habitación	*RegisterHandler*
6	Tomar datos	
7	Mostrar y cerrar factura de la habitación	
8	Crear factura de estadía	*StayBillGenerator*
9	Calcular costo total de la estadía	
10	Crear factura del servicio	*ServicesBillGenerator*

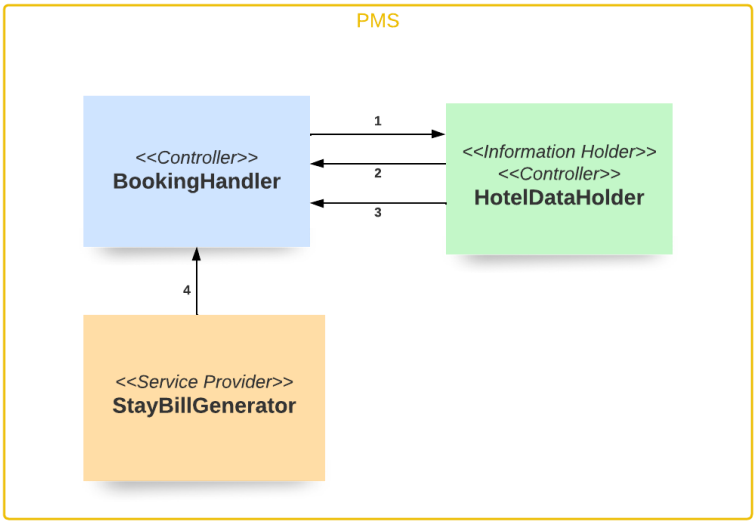
11	Calcular costo del servicio	
12	Registrar el consumo para el huésped	*ConsumeRecorder*
13	Registrar el consumo para la habitación	
14	Asignar si se va a la cuenta final o al pago inmediato	
15	Mostrar y cerrar factura del servicio de pago inmediato	*ServicesPayHandler*
16	Carga de información de las habitaciones	*HotelDataHolder*
17	Carga y configuración de tarifas dinámicas para habitaciones	
18	Consultar inventario	
19	Carga de tarifas y menú del restaurante	
20	Guardar información de las reservas y registros realizados	
21	Inicio de sesión para el usuario	*HotelWorkersAuth*
22	Registro de nuevos empleados	
23	Enrutamiento a la interfaz correspondiente	

En este nivel, se puede evidenciar como la cantidad de responsabilidades aumenta respecto al nivel 1, ya que, al subdividir en más componentes, es más fácil ver como tienen responsabilidades más específicas.

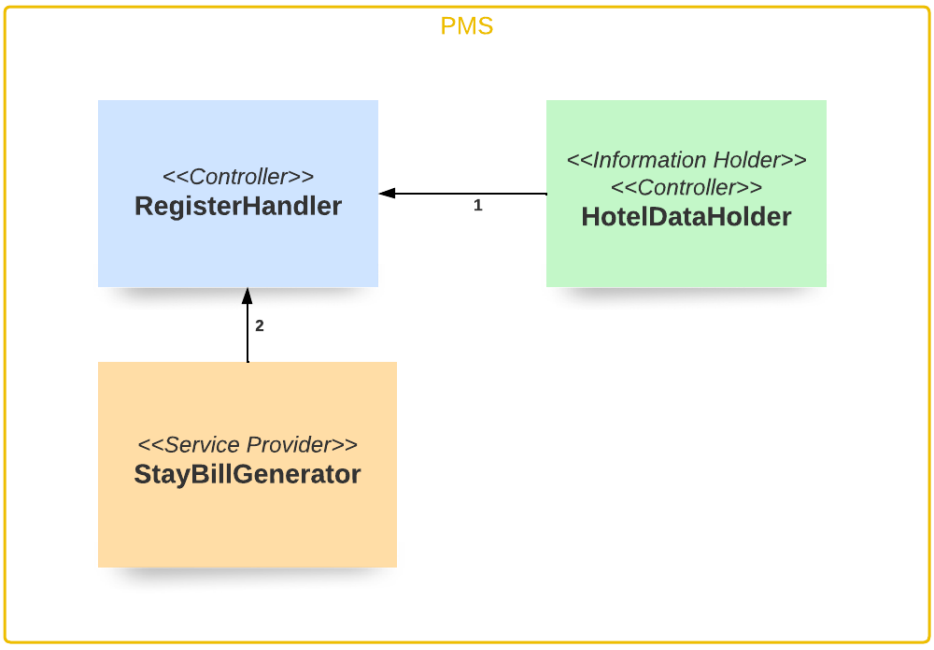
3.3 Colaboraciones

Teniendo en cuenta que ahora hay más especificidad en los componentes, así mismo las colaboraciones cambian y serán más descriptivas para el proceso. Por lo que, a continuación, se muestran las nuevas colaboraciones entre los nuevos componentes.

- **Reservar habitaciones:** El controlador *BookingHandler*, al recibir la información del usuario para reservar:
    1. Le indica a *HotelDataHolder* que le proporcione información del inventario disponible, es decir la información de las habitaciones disponibles para hacer la reserva.
    2. Le indica a *HotelDataHolder* que debe almacenar los nuevos datos de la reserva que se acabó de hacer.
    3. Le indica a *HotelDataHolder* que actualice la disponibilidad a ocupado de la habitación que se acabó de reservar
    4. Le indica a *StayBillGenerator* que guarde la información de la tarifa de la nueva reserva

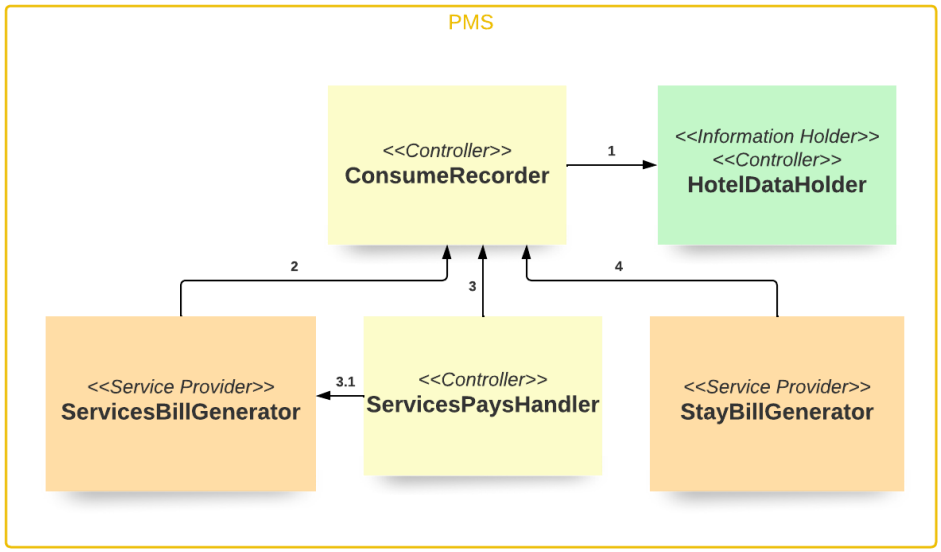


- **Registrar huéspedes:** El controlador *RegisterHandler*, al recibir información de los nuevos huéspedes que va a alojar en el hotel:
    1. Le indica a *HotelDataHolder* que guarde la información sobre el nuevo registro hecho.
    2. Le indica a *StayBillGenerator* que cree la factura de la estadía, con el fin de que se almacene el consumo de los servicios y la estadía de los nuevos huéspedes.

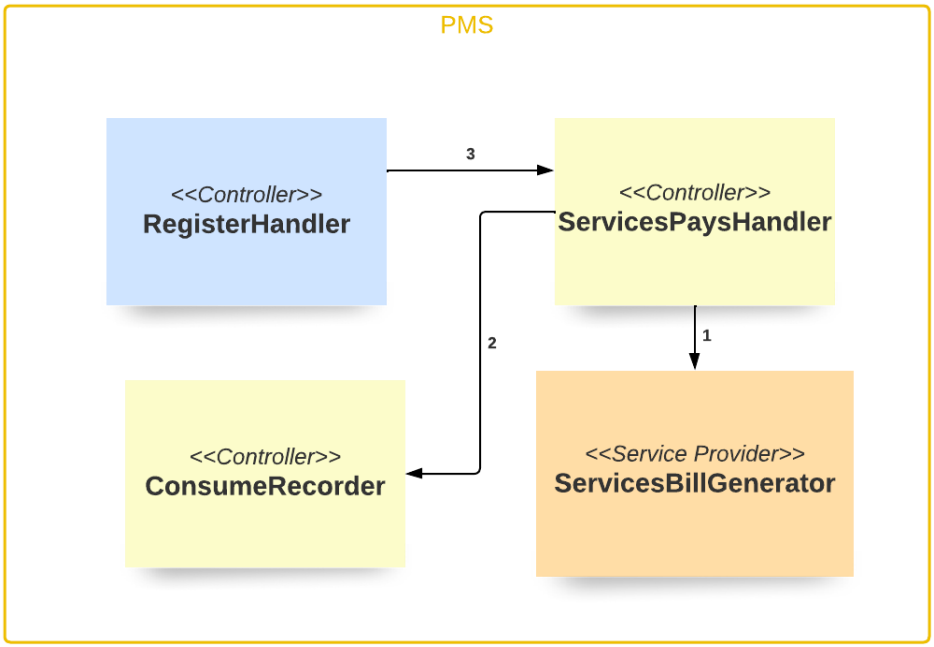


- **Consumir servicio:** el controlador *ConsumeRecorder* necesita dejar registrado que servicios se consumieron por parte de los huéspedes, para ello:
    1. Le indica a *HotelDataHolder* que le proporcione la información de los servicios que hay disponibles en el hotel.
    2. Le indica ServicesBillGenerator que genere una nueva factura que registre el servicio que se va a consumir y calcule su valor.
    3. Le indica a ServicesPayHandler que realice un cobro de la factura generada a partir de ServicesBillGenerator.

4. Si el huésped quiere pagar después, Le indica a *StayBillGenerator* que registre un nuevo consumo a la factura de estadía ya creada en el registro.

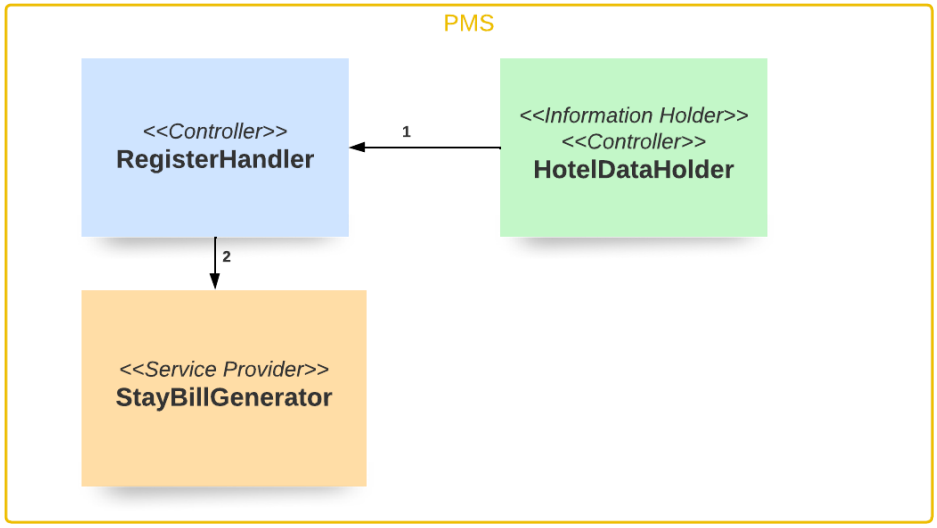


- **Pagar un servicio:** El controlador *ServicesPayHandler*
    1. Le indica a *ServicesBillGenerator* que le calcule el costo del servicio específico.
    2. Una vez calculado el costo, le pide la factura a *ConsumeRecorder* para hacer la cancelación de la factura.
    3. Guarda la información del pago y de la orden en el registro, para lo cual le indica a *RegisterHandler* que guarde el servicio como un servicio que ya fue pagado.



- **Hacer check-out:** El controlador *RegisterHandler*, al recibir la orden de realizar el check-out de un registro:
    1. Llama al *HotelDataHandler* para que actualice la información de la habitación (la libere)

2. Llama a *StayBillGenerator* para que devuelva la factura con el valor del hospedaje y de los servicios que fueron cargados a la habitación.



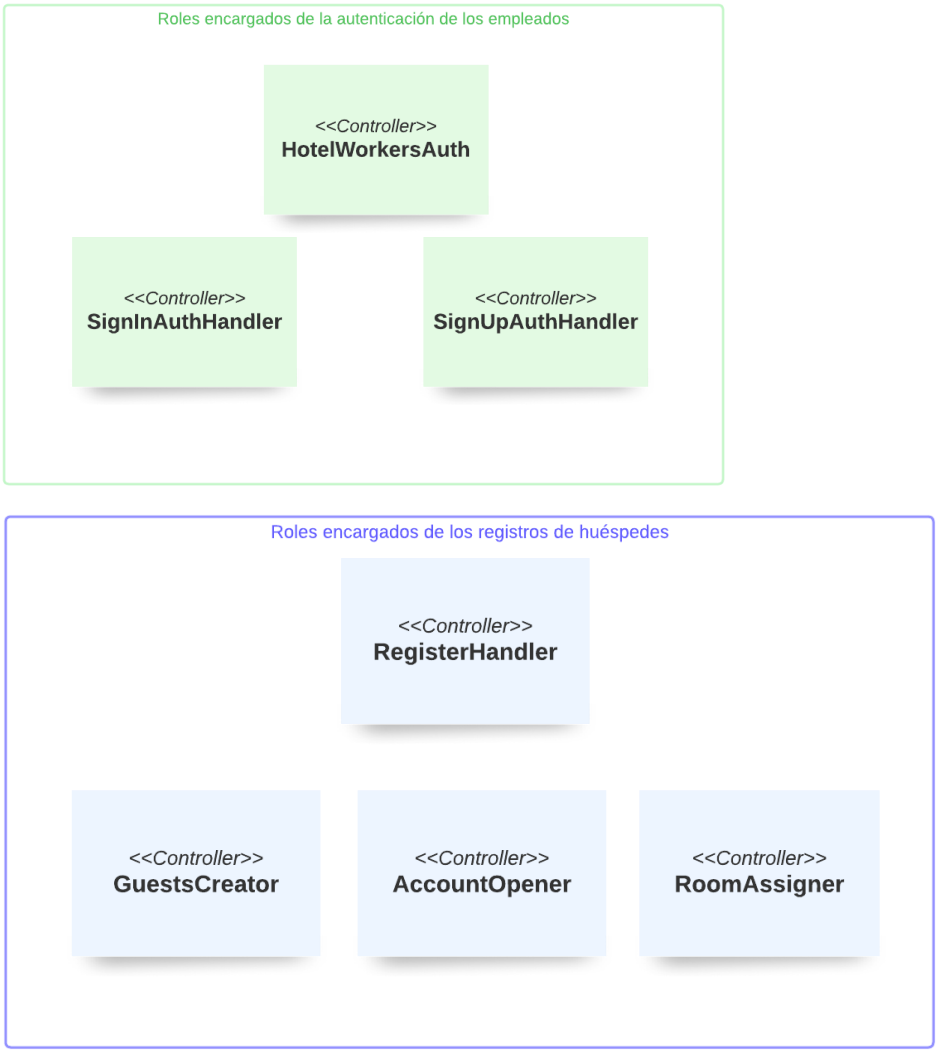
Es evidente que todas las colaboraciones definidas tienen un estilo de control delegado, ya que la lógica de cada funcionalidad importante se divide en diferentes controladores, por ejemplo, para hacer el registro y reserva, se utilizan otros componentes para que ayuden a completar la funcionalidad que implica hacer estas dos importantes funcionalidades. Así mismo, las colaboraciones están fuertemente conectadas, puesto que, al ser un estilo de control delegado, implica que una colaboración necesite de la otra para poder cumplir un objetivo más grande. Un ejemplo, es cumplir el objetivo de hacer check-out, para ello, se necesita una fuerte colaboración entre el *HotelDataHandler y el StayBillGenerator* para poder cancelar los servicios consumidos por el huésped.

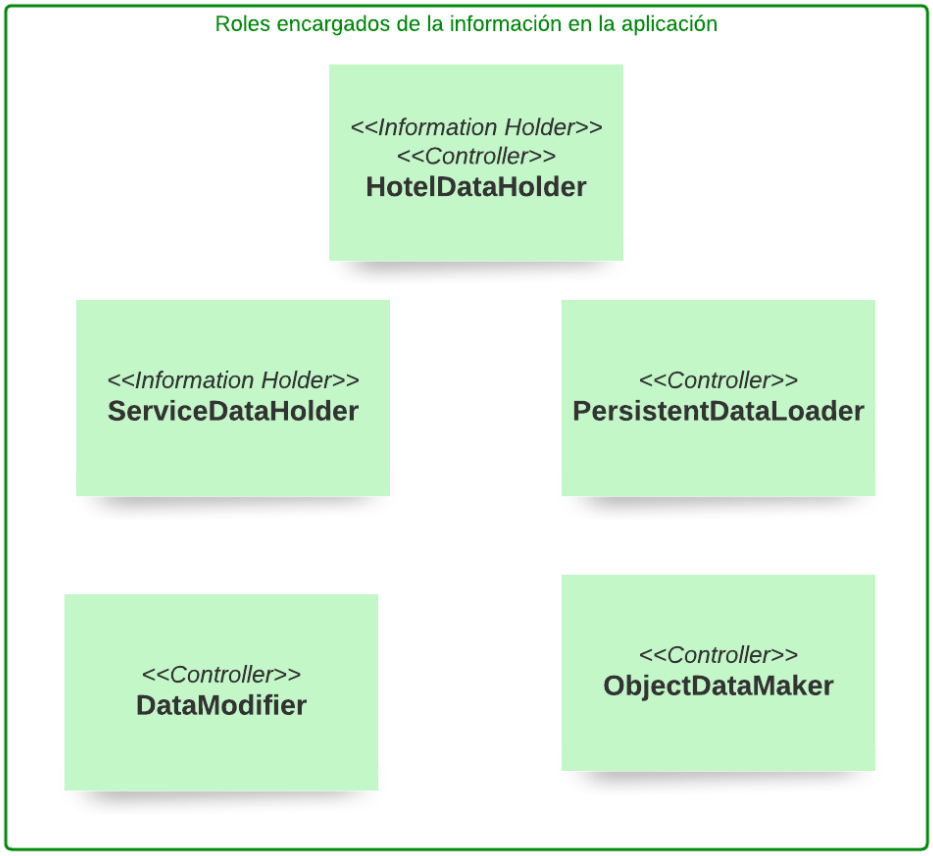
4 Nivel 3

Como objetivo de esta tercera iteración, se realizará un desglosamiento más profundo de los elementos del diseño para lograr un acercamiento a elementos que tengan cercanía a clases y métodos implementables

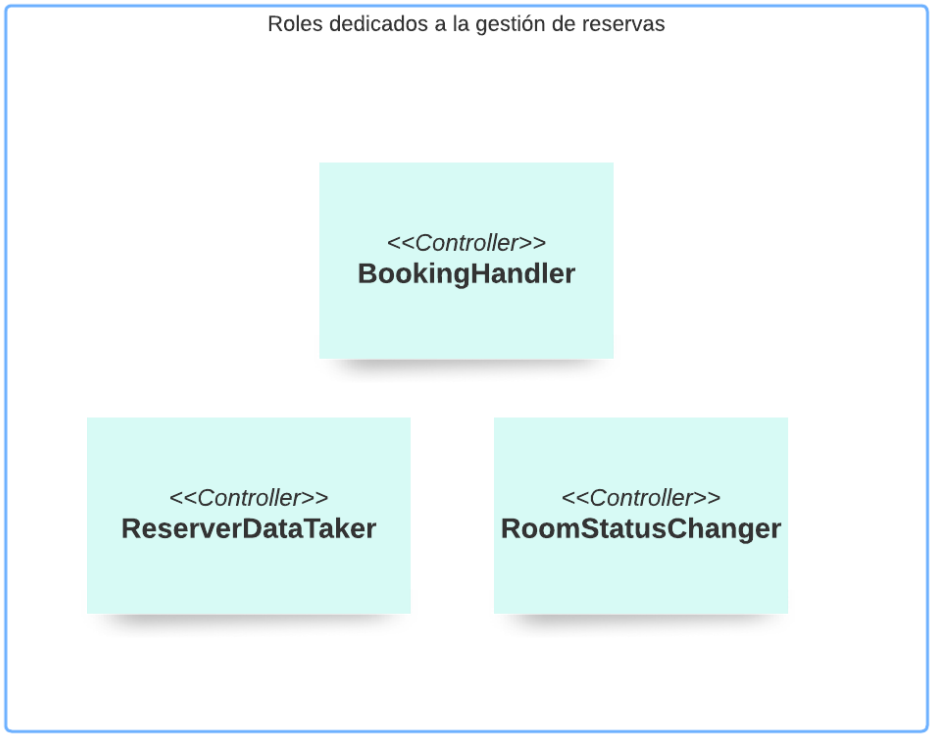
4.1 Componentes y roles

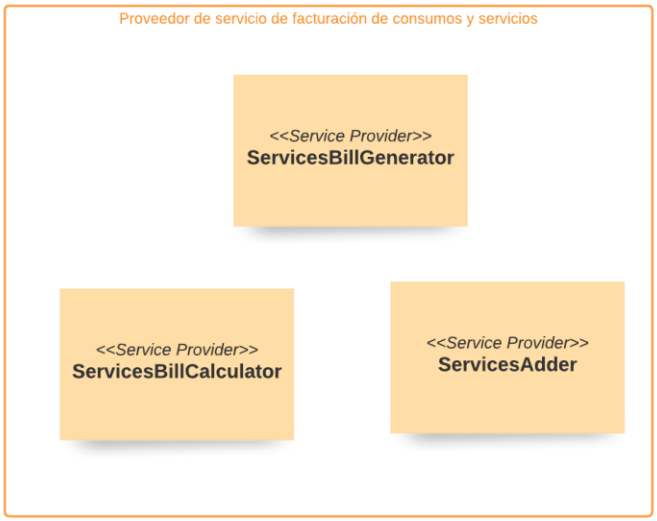
Se parte de la base de los roles designados en la iteración anterior para subdividir estos en representadores de actividades que se realizarán en la aplicación.

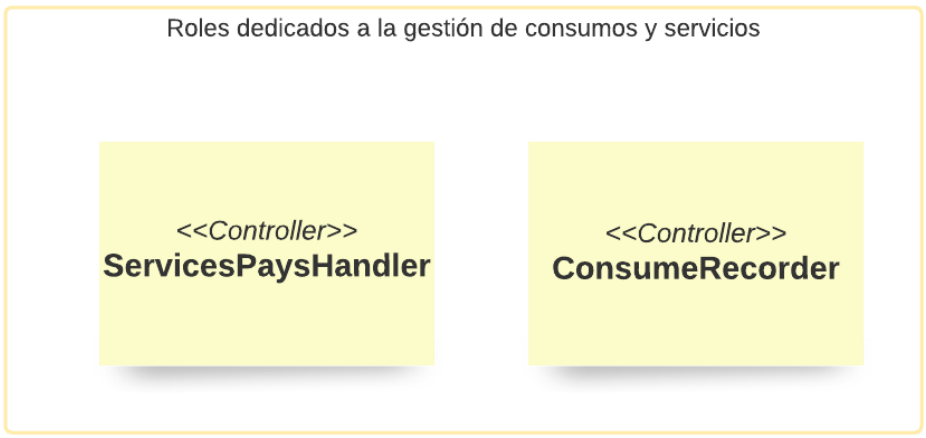

Roles encargados de la autenticación de los empleados
<<Controller>>
HotelWorkersAuth
<<Controller>>
SignInAuthHandler
<<Controller>>
SignUpAuthHandler
Roles encargados de los registros de huéspedes
<<Controller>>
RegisterHandler
<<Controller>>
GuestsCreator
<<Controller>>
AccountOpener
<<Controller>>
RoomAssigner


Roles encargados de la información en la aplicación
<<Information Holder>>
<<Controller>>
HotelDataHolder
<<Information Holder>>
ServiceDataHolder
<<Controller>>
PersistentDataLoader
<<Controller>>
DataModifier
<<Controller>>
ObjectDataMaker


Proveedor de servicio de facturación del hospedaje
<<Service Provider>>
StayBillGenerator
<<Service Provider>>
StayBillCalculator
<<Service Provider>>
ServiceAdder
<<Service Provider>>
RoomAdder


Roles dedicados a la gestión de reservas
<<Controller>>
BookingHandler
<<Controller>>
ReserverDataTaker
<<Controller>>
RoomStatusChanger


Proveedor de servicio de facturación de consumos y servicios
<<Service Provider>>
ServicesBillGenerator
<<Service Provider>>
ServicesBillCalculator
<<Service Provider>>
ServicesAdder


Roles dedicados a la gestión de consumos y servicios
<<Controller>>
ServicesPaysHandler
<<Controller>>
ConsumeRecorder

4.2 Responsabilidades

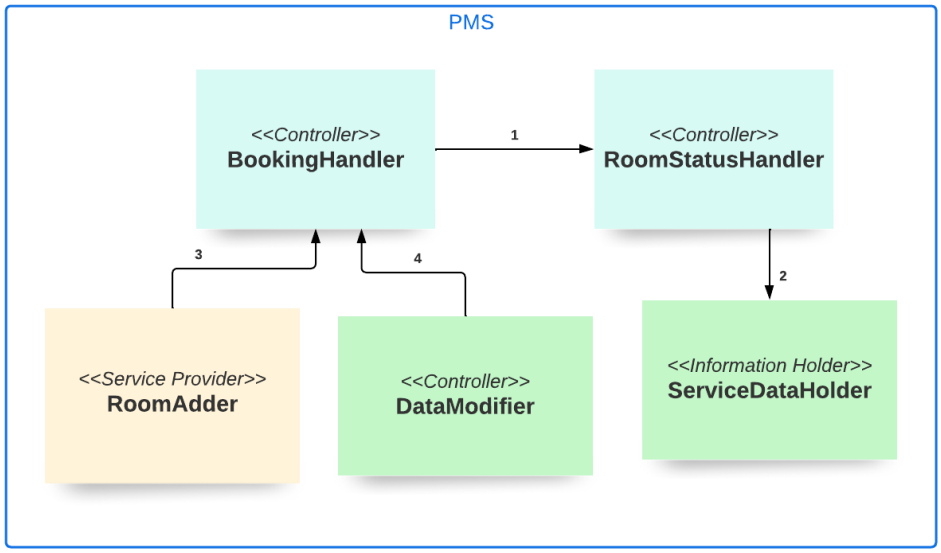
Teniendo en cuenta que hay más granularidad en cuanto a la definición de componentes, así mismo hay más especificaciones de las responsabilidades que cada uno toma. Por lo tanto, la siguiente tabla muestra las responsabilidades adquiridas de cada componente nuevo definido.

#	Responsabilidades	Componente
1	Verificar disponibilidad de habitaciones	*RoomStatusHandler*
2	Generar reserva	*BookingHandler*
3	Guardar reserva	
4	Cancelar reserva	
5	Informar tarifa de la reserva	
6	Cambiar status habitación	*RoomStatusHandler*
7	Cargar nueva cuenta de habitación	*AccountHandler*
8	Tomar datos huésped principal	*GuestsCreator*
9	Tomar datos grupo	
10	Mostrar factura de la habitación	*StayBillGenerator*
11	Cerrar factura de la habitación	
12	Crear factura de estadía	
13	Calcular costo del servicio	*ServicesBillCalculator*
14	Registrar consumo para el huésped	*ConsumeRecorder*
15	Registrar el consumo para la habitación	
16	Asignar pago a cuenta final	
17	Realizar pago inmediato	*ServicesPaysHandler*
18	Cerrar factura de pago inmediato	*ServicesBillGenerator*
19	Mostrar factura de pago inmediato	
20	Cargar información de las habitaciones	*PersistentDataLoader*
21	Agregar información de una habitación manualmente	*ObjectDataMaker*
22	Configuración de tarifas dinámicas para habitaciones	*DataModifier*
23	Consultar inventario	*ServiceDataHolder*
24	Carga de menú del restaurante	*PersistentDataLoader*
25	Carga de tarifas del menú del restaurante	
25	Guardar información de las reservas	*DataModifier*
26	Guardar registros realizados	
27	Inicio de sesión de usuario	*SignInAuthHandler*
28	Registro de nuevos empleados	
29	Enrutamiento a la interfaz	*HotelWorkersAuth*

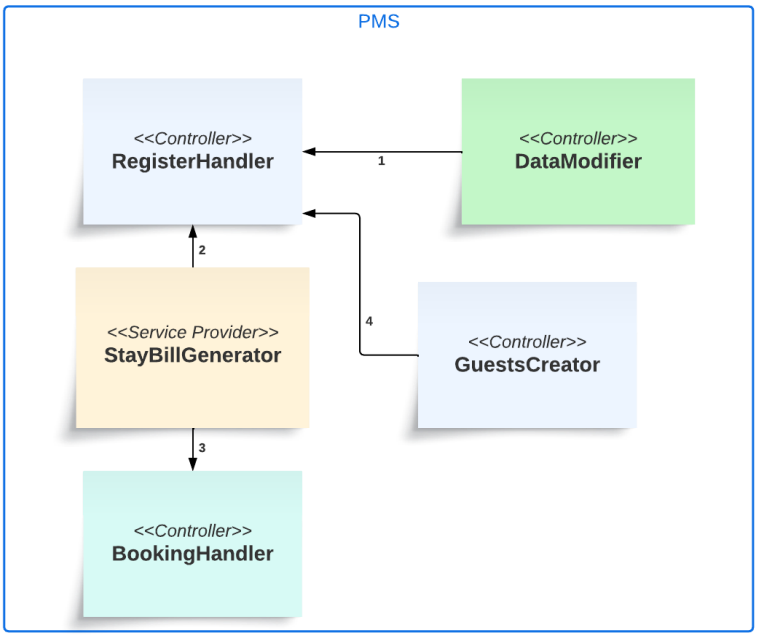
4.3 Colaboraciones

- **Reservar habitaciones:** El controlador *BookingHandler*, al recibir la información del usuario como los días a reservar (fecha inicial, fecha final) y los datos del usuario (nombre, email, DNI, numero celular):
    1. Le pide a *RoomStatusHandler* la información con la disponibilidad de las habitaciones para reservar.
    2. *RoomStatusHandler* le pide a *ServiceDataHolder* la información del inventario disponible para poder consultar las características de cada habitación por específico.

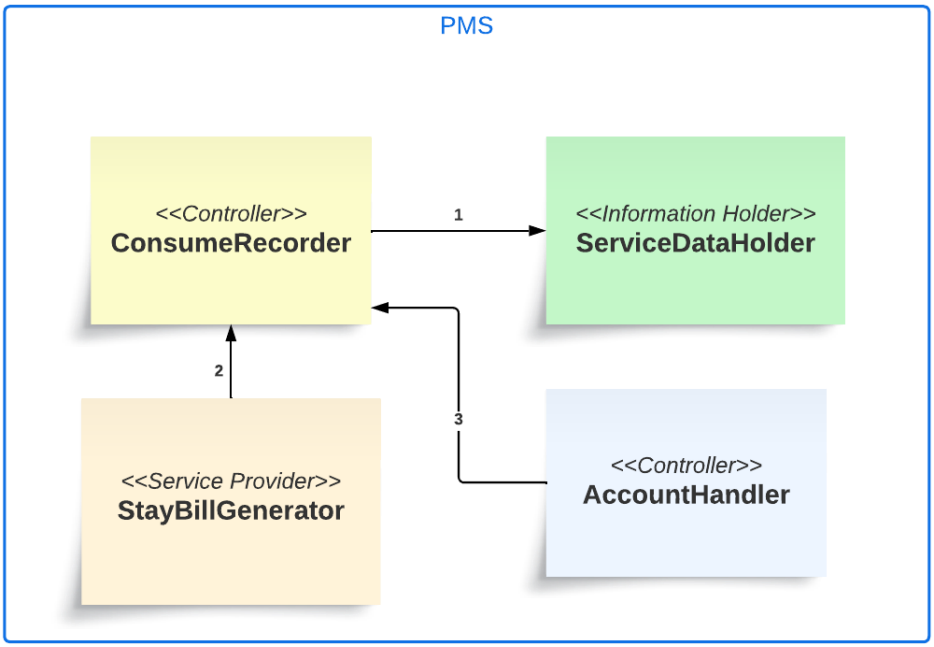
3. Le pide a *RoomAdder* que cambie el status de la habitación a ocupado por los días que se hizo la nueva reserva.
4. Le da a *DataModifier* los datos del nuevo registro realizado para que los guarde.



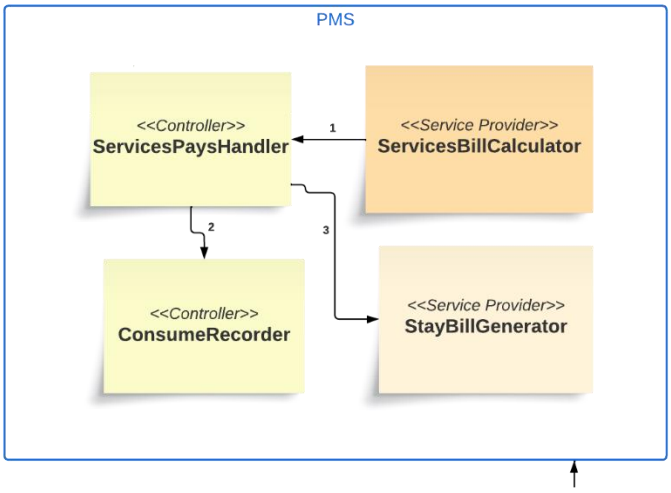
- **Registrar huéspedes:** El controlador RegisterHandler, al recibir información de cada uno de los huéspedes:
    1. Le indica a *DataModifier* que guarde la información sobre el nuevo registro realizado.
    2. Le indica a *StayBillGenerator* que cree la factura de la estadía para poder almacenar el consumo de los servicios y la estadía de los nuevos huéspedes.
    3. *StayBillGenerator* le pide a *BookingHandler* que le informe sobre la tarifa de la reserva que se hizo.
    4. Le pide a *GuestCreator* que tome los datos del huésped principal y de su grupo asociado.



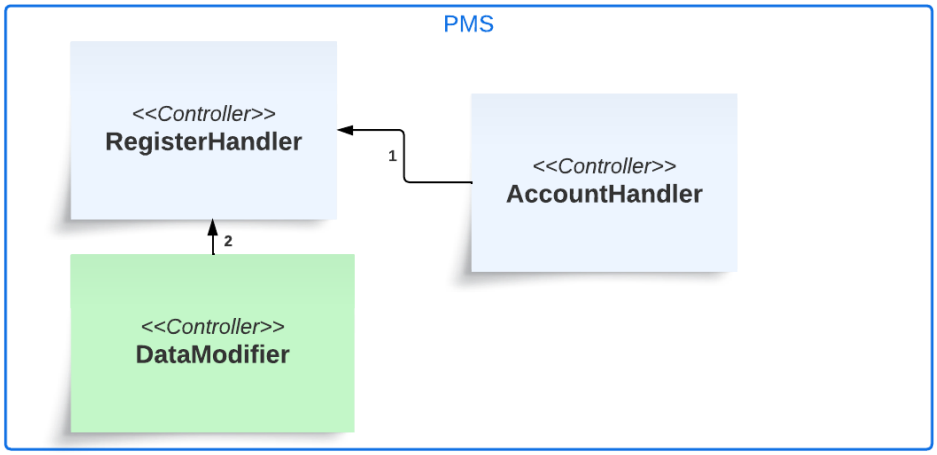
- **Consumir servicios:** el controlador ConsumeRecorder necesita dejar registrado que servicios se consumieron por parte de los huéspedes, para ello:
    1. Le pide *ServideDataHolder* que le informe sobre el inventario de los servicios disponibles en el hotel
    2. Le indica a *ServicesBillGenerator* que haga el cálculo del costo del servicio consumido
    3. Si el huésped decide pagar después, le indica a *AccountHandler* que cargue una nueva cuenta a la habitación en donde se encuentra el huésped.



- **Pagar servicio:** Los huéspedes pueden elegir entre pagar a la salida (al hacer check-out) o pagar inmediatamente, en este caso se hace el pago inmediatamente, por lo que el controlador ServicesPayHandler :
    1. Le indica a *ServicesBillCalculator* que le calcule el costo del servicio específico.
    2. Una vez calculado el costo, le pide que registre el consumo a *ConsumeRecorder* para hacer la cancelación de la factura.
    3. Guarda la información del pago y de la orden en el registro, y le indica a *StayBillGenerator* que guarde el servicio como un servicio que ya fue pagado.



- **Hacer Check-out:** El controlador RegisterHandler, al recibir la orden de realizar el check-out de un registro:
    1. Le pide a *accountHandler* que cierre y guarde la cuenta.
    2. Le pasa a *DataModifier* toda la información del hospedaje para que la guarde.



5 Nivel 4

Continuando con el proceso de diseño, se logra llegar a la máxima granularidad para cada componente definido anteriormente, por lo que en este nivel se procede a descomponer cada componente en diferentes clases.

5.1 HotelDataHolder

Para empezar, es importante recordar las responsabilidades que este componente utiliza:

- Consultar Inventario
- Cargar información de las habitaciones
- Guardar información de las reservas
- Guardar registros realizados
- Agregar información de los registros
- Agregar información de las habitaciones.

5.1.1 Componentes candidatos y estereotipos

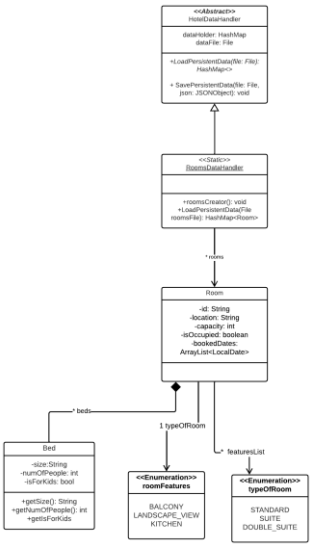
Así mismo, es importante recordar que en el nivel 3 este rol se descompuso en otros 4 componentes, estos cuatro roles en los que se subdividió HotelDataHolder son: ServiceDataHolder, PersistentDataLoader, DataModifier y ObjectDataMaker.

El primer rol llamado ServiceDataHolder, se encarga de almacenar toda la información relacionada a los servicios del hotel. Por ejemplo, almacena el costo de los servicios, que servicios hay disponibles, si el servicio puede llevarse a la habitación o si el servicio se paga por grupo o no. Con el fin de aumentar la granularidad de este rol, ya se especifica en 4 roles específicos, que son RoomsDataHandler, ServicesDataHandler, FoodDataHandler, BookingsDataHandler y RegistrationDataHandler.

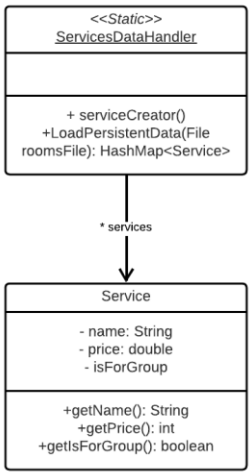
Por otro lado, tenemos a PersistentDataLoader, que se convirtió en un método fundamental para el HotelDataLoader quien es el cabeza de todos los information holders del sistema. Así mismo, DataModifier, se convirtió en un método para las clases FaresDataHandler y ServicesDataHandler que más adelante se van a especificar. Igualmente, el rol ObjectDataMaker se convierte en un método de la clase principal HotelDataHandler.

5.1.2 Responsabilidades

En cuanto a RoomsDataHandler, es una clase que maneja la información general de las habitaciones, como por ejemplo cargar sus características o crear una habitación. Así mismo, esta clase contiene varias clases Room, que a su vez contienen la información específica de una habitación como su id, ubicación, capacidad y demás atributos que son fundamentales para distinguir una habitación. A continuación, se muestra cómo queda desglosado estas nuevas clases.



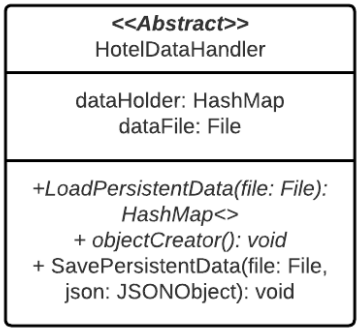
Continuando con la descomposición de ServiceDataHolder, se subdivide en otras dos clases: ServicesDataHandler y Service. Primero, ServicesDataHandler es una clase que se encarga de que el administrador sea capaz de cargar nuevos servicios al hotel y así mismo poder agregar un servicio nuevo por aparte. Segundo, necesitamos de Services con el fin de tener las características genéricas de cualquier tipo de servicio. Principalmente, la clase Services se hace con el fin de que más adelante se puedan guardar las características principales de nuevos servicios que pueda llegar a tener el hotel. Finalmente, estas dos clases quedan de la siguiente manera:



Seguidamente, se sigue descomponiendo ServiceDataHolder, que va a seguir subdividiéndose en FoodDataHandler y Food. En el caso de FoodDataHandler, se guarda información del menú del restaurante, en donde tiene metodos como LoadPersistenData() que se encarga de que el administrador pueda cargar los nuevos datos para el menú como los nuevos platos o bebidas y sus respectivos precios. Por otro lado, también tenemos a Food que es una clase que se encarga de modelar un producto para el restaurante, por ello contiene características como su precio, nombre, y contiene getters para que se pueda consultar el producto modelado para el servicio del restaurante. De esta manera, la relación entre estas dos clases queda de la siguiente forma:

Así mismo, ServiceDataHolder se descompone en la clase BookingsDataHandler que se encarga de guardar las reservar realizadas. Al igual que BookingsDataHandler, se crea una clase Registration DataHandler que se encarga de guardar los registros realizados en el hotel. En este paso, se muestra cómo se modelaron ambas clases:

Finalmente, tenemos a PersistentDataLoader que se convierte en un método para HotelDataLoader llamado LodadPersistentData que se encarga principalmente de cargar los datos que se encuentran en los archivos.



5.1.3 Colaboraciones

Respecto a las colaboraciones entre todos los componentes, se puede evidenciar como se tiene un sistema delegado, puesto que HotelDataHandler tiene el control de toda la información del hotel de manera separada, es decir en cada clase hay almacenamiento de diferentes tipos de datos que componen la información del hotel.

5.2 BookingHandler

Este rol en específico se encarga de manejar toda la lógica de las reservar para el hotel, por lo que tiene las responsabilidades como:

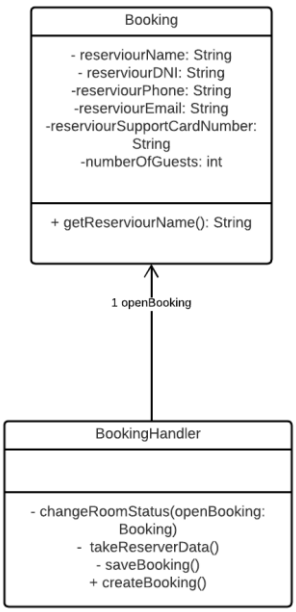
- Generar una reserva
- Guardar la reserva
- Cancelar reserva

5.2.1 Componentes candidatos y estereotipos

A partir de las responsabilidades definidas, se ve la facilidad de crear dos nuevas clases, una llamada Booking, y la otra llamada BookingHandler.

5.2.2 Responsabilidades

En cuanto a la clase Booking, tiene la funcionalidad de hacer una nueva reserva, por lo que para pdoer hacerla, necesita datos de quien va a hacer la reserva como su nombre, su DNI, su email, su número de teléfono y la cantidad de personas que lo acompañan, de tal manera que se traducen como atributos de la clase. Así mismo, Booking tiene un método que se encarga de obtener el nombre de quien hizo la reserva. Por otro lado, tenemos a bookingHandler, que se encarga de hacer toda la lógica asociada a una reserva, como cambiar el estatus a ocupado de las habitaciones que se van a reservar, guardar la reserva en la base de datos del hotel, crear la nueva reserva con los datos recolectados y obtener los datos de la reserva hecha. Todas estas funcionalidades se traducen en métodos, de tal manera que la creación de las clases con sus atributos y métodos queda la siguiente manera:



5.2.3 Colaboraciones

Es importante destacar que este rol tiene una colaboración muy importante con los roles que contienen toda la información del hotel, ya que como se dijo desde el principio, el rol de hacer una reserva se encarga de crear y guardar la serva en el sistema, por lo que implica guardar nuevos datos en el sistema del hotel. Por lo que, de esta manera, los roles entre los information holder y booking están fuertemente conectados entre sí.

6 Diseño final

Para mayor practicidad se adjunta un pdf con el diseño uml completo