**Proyecto 1: Property Management System (PMS)**

**Jhoan Sebastian Sánchez Suarez**

**Manuela Rivas Gomez**

**Clay Felipe Charry Pinzón**

1. **Contexto del problema**

Se está desarrollando un programa para el manejo de distintas funcionalidades dentro del sistema de un hotel. Para este programa se tienen 3 entes principales que podrán interactuar con el sistema que son, administrados, empleado (recepcionista o empleado regular) y huésped. Con esto en mente, cada ente puede llevar a cabo una serie de acciones y solicitudes en el sistema, siendo el huésped el ente con más restricciones de interacción y el administrador con conocimiento y control total de las diferentes funcionalidades del sistema. Es por esto por lo que el flujo y funcionamiento del sistema se puede entender con base en la figura 1. Esta figura ilustra las interacciones entre los diferentes entes con el sistema de una forma totalmente genérica, sin embargo, le permite al lector del documento entender el flujo del sistema solicitado.

Figura 1. Diagrama de flujo de funcionalidades del sistema. Este es un diagrama de flujo genérico de las interacciones entre entes que pueden interactuar con el sistema solicitado y las interacciones que pueden tener con el mismo.
Figura 1. Diagrama de flujo del funcionamiento e interacciones en el sistema propuesto. Diagramade flujo genérico de las interacciones entre entes que pueden interactuar con el sistema solicitado y las interacciones que pueden tener con el mismo.

Se proponen inicialmente tres tipos de interfaces administradoras con las cuales únicamente empleados y/o el administrador pueden interactuar, es importante recalcar que el huésped no tiene acceso directo a ninguno de estos administradores de funcionalidades, por lo cual es el ente con mayor cantidad de restricciones prácticas dentro del sistema solicitado. Cabe resaltar que el administrador es el único que tiene la potestad de interactuar con el administrador de inventario de habitaciones, ya que debe ser el único habilitado para modificar tarifas de habitaciones, ingresar nuevas habitaciones y modificar los atributos de las mismas y comparte con los empleados el acceso al administrador de servicios ya que, el administrador puede también modificar tarifas e ingresar servicios y el empleado es le encargado de ingresar el consumo de un servicio por parte de los huéspedes. Por último, los empleados son los que principalmente interactúan con el administrados de reservas, puesto que son los entes encargados de registrar las reservas, salidas, entradas y cuentas de los huéspedes en el sistema. El huésped no interactúa directamente con ningún administrador del sistema, solo puede realizarle solicitudes al empleado para poder interactuar con el mismo.

**2. Nivel 1**

**2.1. Componentes candidatos y estereotipos**

1. Al reconocer la necesidad de administrar las habitaciones y poder controlar toda la información referente a estas, es evidente la necesidad de implementar un “Administrador de habitaciones” encargado de almacenar consigo la información referente al inventario de las habitaciones, controlar las tarifas de estas y permitir la creación de nuevas habitaciones. Este administrador cumple con los estereotipos *Information Holder* y *Controller* también para no solo almacenar la información sino también poder modificar ciertas cosas e interactuar con las partes de este.

2. En respuesta a la necesidad funcional del ingreso, factura y control de servicios adicionales que puede ofrecer el hotel se requiere otro componente encargado de interactuar con estos servicios de forma que se debe implementar un “Administrador de servicios” encargado de almacenar consigo la información referente a la facturación correspondiente a cada servicio adicional y consumo de estos por parte de los huéspedes, además modificar la información correspondiente a los servicios por lo cual este adopta el estereotipo de *Information Holder* y *Controller*.

3. En cuanto a la última necesidad funcional solicitada para la ejecución del programa, fue evidente que era necesario generar un “Administrador de huéspedes”. Este está encargado de todas las funcionalidades relacionadas directamente con la interacción con los huéspedes como ingresar servicios adicionales a la cuenta final de estos, generar y cancelar reservas y almacenar la información de estadía de los huéspedes. Es por esto que este administrador cumple con los estereotipos de *Information Holder* y *Controller*.

La figura 2 que se presenta a continuación ilustra los tres componentes candidatos propuestos previamente.



Figura 2. Componentes candidatos con sus respectivos estereotipos. Se ilustran los tres componentes candidatos para llevar a cabo todos los requerimientos funcionales con sus respectivos estereotipos que componen el sistema (PMS).

**2.2. Responsabilidades**

Con la selección y desarrollo de los tres componentes candidatos, se tomaron en cuenta, implícitamente una serie de responsabilidades posibles que se deben ocupar para cumplir con los requerimientos funcionales y no funcionales del sistema solicitado. La tabla 1 presentada a continuación presenta estas responsabilidades de forma explícita y las asigna al componente encargado de asumirlas.

**Tabla de responsabilidades con componentes asociados**

|  |  |  |
| --- | --- | --- |
| **#** | **Responsabilidad** | **Componente** |
| **1** | Cargar habitaciones | **Administrador de habitaciones** |
| **2** | Crear habitación |
| **3** | Modificar tarifa habitación |
| **4** | Ingresar tarifa habitación |
| **5** | Establecer servicio | **Administrador de servicios** |
| **6** | Modificar tarifa servicio |
| **7** | Registrar consumo |
| **8** | Generar factura |
| **9** | Actualizar cuenta final | **Administrador de huéspedes** |
| **10** | Generar factura |
| **11** | Solicitar reserva |
| **12** | Cancelar reserva |
| **13** | Registrar llegada |
| **14** | Registrar salida |
| **15** | Registrar huésped |
| **16** | Registrar grupo |

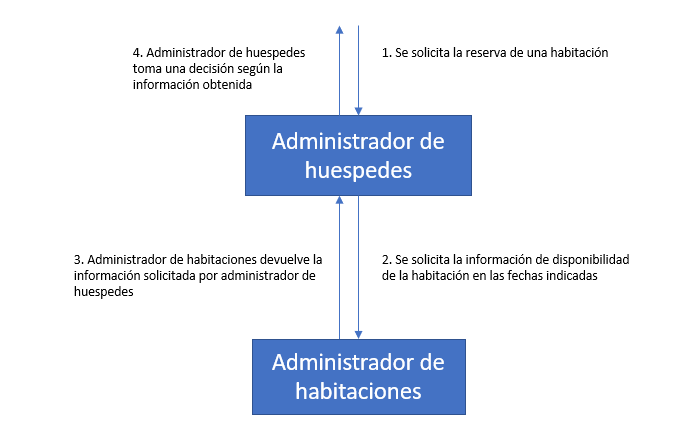
**Tabla 1.** Tabla de responsabilidades con componentes candidatos asociados. Se exponen una serie de responsabilidades preliminares que se consideraron a la hora de desarrollar los componentes candidatos para el sistema (PMS).

Dado que el registro del cobro de los servicios consumidos por los huéspedes se puede ingresar a un cobro de habitación relacionado directamente con el huésped, se tomó la decisión de que para esta responsabilidad, el encargado de llevar la cuenta de cobros de los servicios adicionales consumidos por los huéspedes sea el administrador de huéspedes y no el de servicios. El resto de las responsabilidades se delegaron de acuerdo con las facilidades y accesos con los que cuenta cada componente candidatos descritas previamente. Se tomo la decisión de separar las responsabilidades correspondientes a cada administrador de la forma en la que se muestra puesto que en el contexto del problema nos dimos cuenta de que hay 3 principales módulos de interacción de la plataforma, los cuales son: cosas que hacen los huéspedes, servicios que ofrece el hotel y manejo de habitaciones. De ahí, cada una de estas formas de interacción debería permitir una serie de acciones, por lo que se prosiguió a identificar cuales serían las cosas que deberían poder hacerse para estos y estas acciones son las responsabilidades correspondientes.

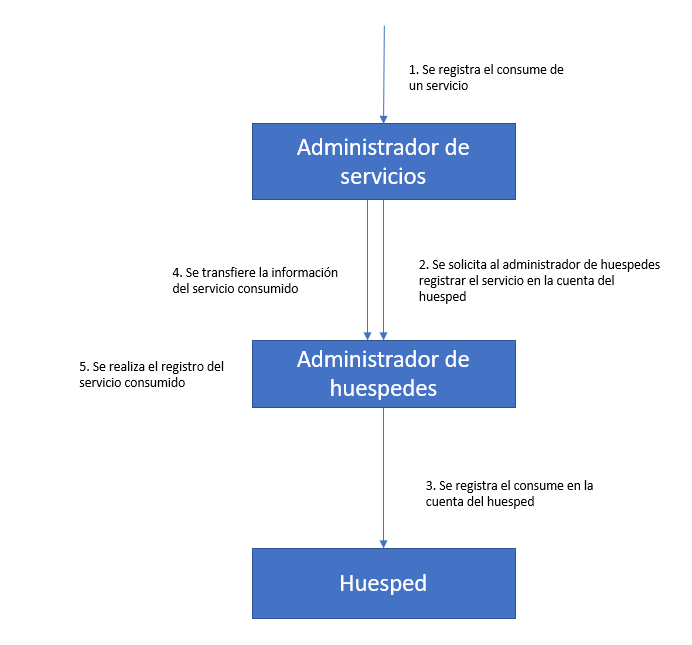
**2.3. Colaboraciones**

Algunas de las responsabilidades se deben realizar en conjunto entre los diferentes componentes candidatos que se tienen para desarrollar el sistema solicitado. Estas se presentan a continuación:

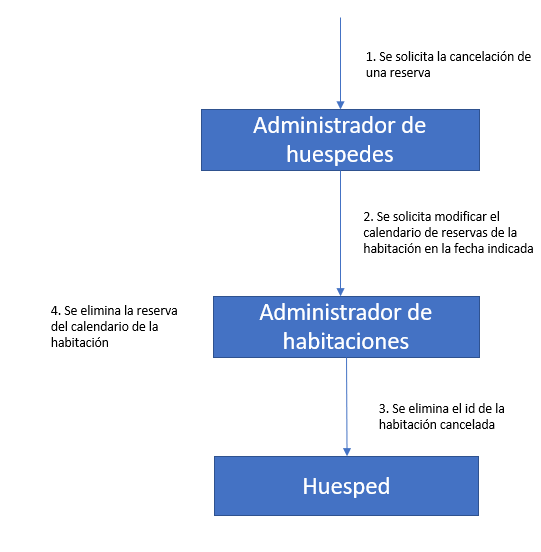
1. **Reservar una habitación:** Para poder realizar esta tarea, el sistema debe poder interactuar entre el administrador de huéspedes y de habitaciones de forma que se pueda recopilar la información necesaria para llevarla a cabo. El flujo de este proceso se ilustra a continuación:
   * 1. Administrador de huéspedes recibe la solicitud de realizar una reserva de una habitación
     2. Administrador de huéspedes le solicita al administrador de habitaciones la disponibilidad de la habitación que se desea reservar para las fechas indicadas
     3. Administrador de habitaciones revisa la disponibilidad y se la devuelve al administrador de huéspedes
     4. Con base en esto el administrador de huéspedes toma una decisión respecto a la reserva solicitada



1. **Agregar un servicio a una cuenta:** Para poder agregar un servicio a una cuenta el sistema debe tener la posibilidad de realizar una interacción entre el administrador de servicios y de huespedes para recopilar la información necesaria para llevarla a cabo. El flujo de este proceso se ilustra a continuación:
   * 1. Administrador de servicios registra el consumo de un servicio y le solicita al administrado de huéspedes que registre en la cuenta del huésped el consumo del servicio indicado
     2. Administrador de servicios le pasa la información del servicio consumido al administrador de huéspedes
     3. Administrador de huéspedes realiza el registro del servicio consumido por el huésped



1. **Cancelar una reserva:** Para cancelar una reserva el sistema debe poder interactuar entre el administrador de huéspedes y de habitaciones de forma que se pueda recopilar la información necesaria para llevarla a cabo. El flujo de este proceso se ilustra a continuación:
   * 1. Administrador de huésped recibe la solicitud de cancelar una reserva y le indica al administrador de habitaciones que necesita modificar el calendario de reservas
     2. El administrador de habitaciones recibe la información de la fecha y habitación a la que se le va a realizar la cancelación de la reserva junto con el huésped que la había realizado para poder ejecutar la tarea completa
     3. El administrador de habitaciones elimina la reserva del calendario de la habitación.

**3. Nivel 2**

Siguiendo con el proceso de diseño por niveles, se procede a descomponer cada uno de los componentes perfilados en el nivel anterior.

**3.1. Administrador Habitaciones**

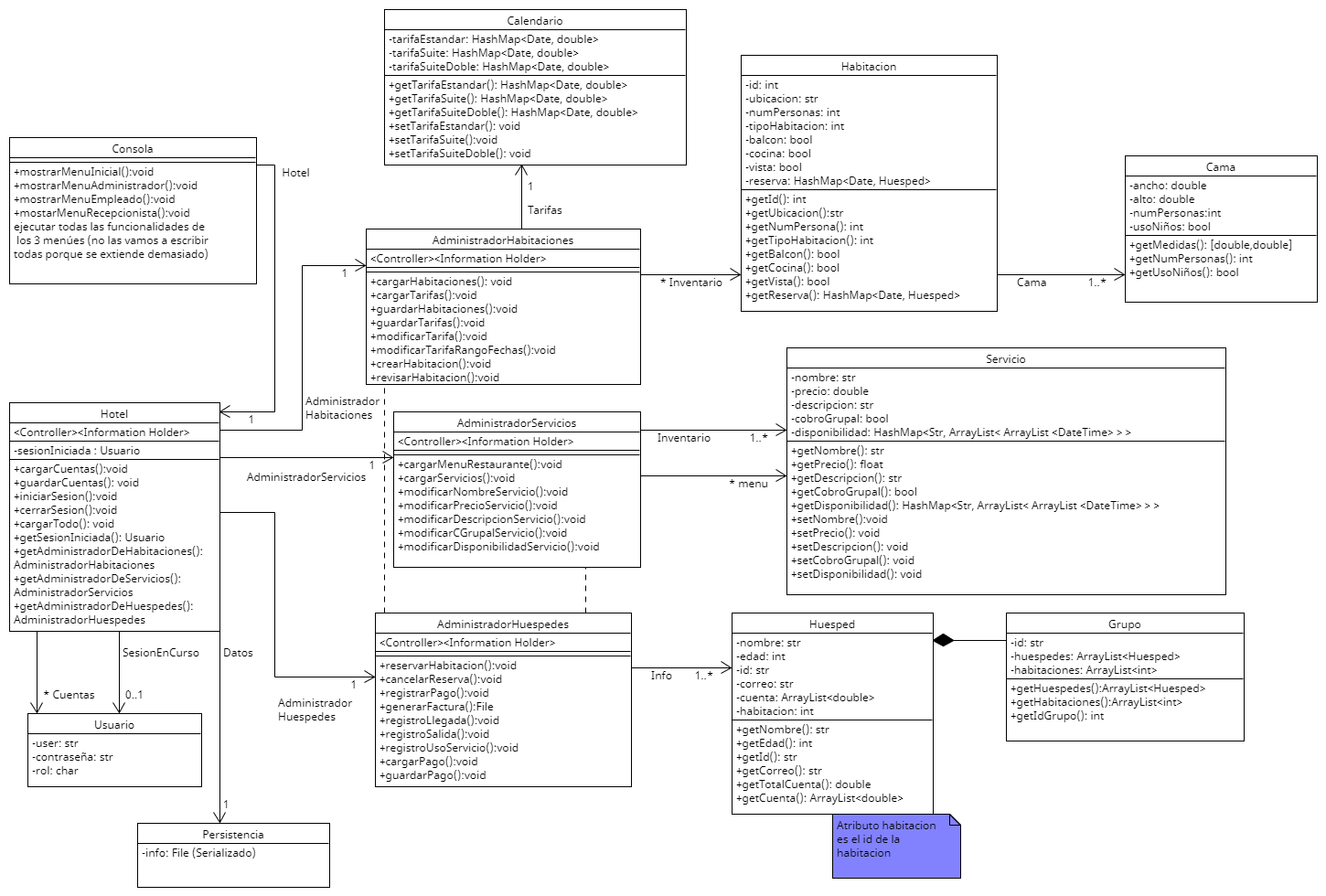
En esta sección se almacenan y controla todo lo relacionado con la gestión de las habitaciones esto incluye:

-Gestionar el inventario (Cargar, ver y crear habitaciones) junto con sus características.

-Guardar las tarifas de las habitaciones.

**3.1.1. Componentes candidatos, estereotipos y responsabilidades.**

Considerando el hecho de que el administrador de habitaciones tiene que gestionar todo lo relacionado con tarifas y las características que contienen todas las habitaciones, administrador habitaciones es un “*controller*” e “*information* *holder*”, el cual para fraccionar y acudir más fácil a su información tiene 2 clases las cuales solo son “*information* *holders*”. El inventario es una tabla de hash única la cual va a contener a cada habitación usando como Key su id, para así posteriormente buscar la información de cada habitación, como el tipo de habitación, su ubicación, y que días está reservada y por quienes. El calendario por otra parte almacena la tarifa de cada tipo de habitación (Estándar, suite, doble suite) teniendo en cuenta las fechas.



Administrador Habitaciones

**Administrador de habitaciones**

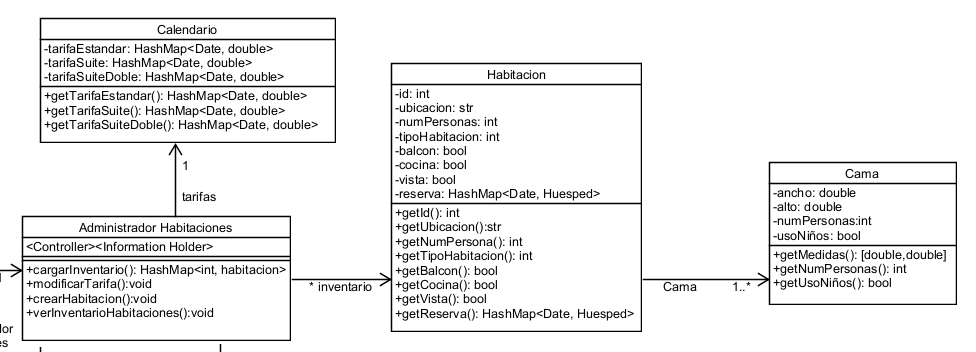


Figura 4: Descomposición del componente “Administrador de habitaciones”

**3.1.2. Colaboraciones**

Con respecto a las colaboraciones entre los 3 componentes, administrador habitaciones es el único que puede modificar sus características y estos se van a mantener intactas, **a excepción de cuando en el administrador de huéspedes se requiere:**

1. Cuando el administrador de habitaciones requiere crear una nueva habitación, indica a la clase habitación la acción en cuestión para configurarla con la información necesaria. Tras hacer lo anterior, se actualiza el inventario de habitaciones en el administrador de habitaciones.
2. Cuando el administrador de habitaciones requiere cambiar la tarifa de una habitación, este solicita al calendario y modifica el precio de las habitaciones en cuestión.

**3.2. Administrador de huéspedes**

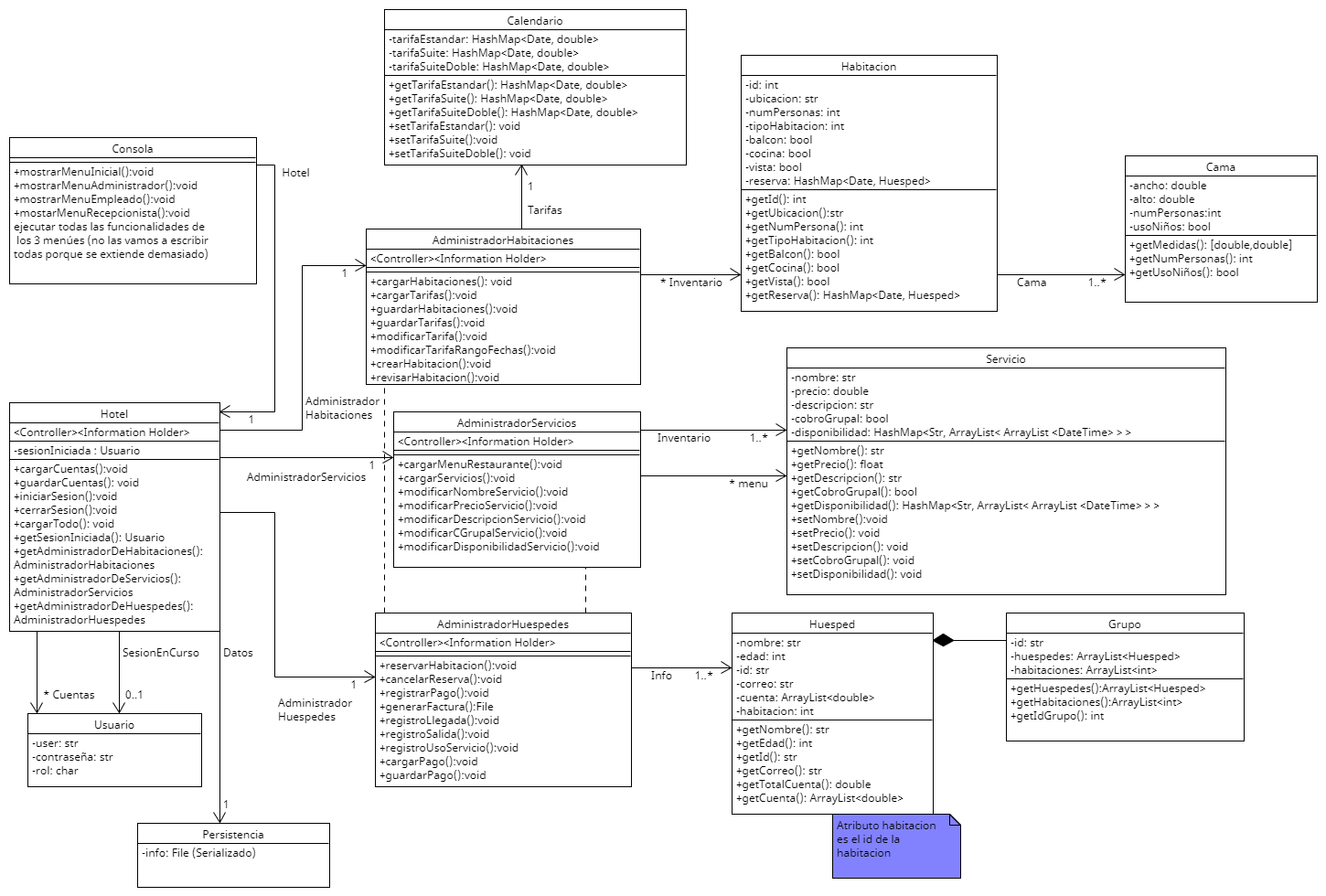
En esta sección se guarda la información y gestiona la interacción con los huéspedes mediante las siguientes responsabilidades:

* Gestionar reservas de habitaciones (Crear y cancelar).
* Gestionar la llegada y salida de los huéspedes.
* Recibir pagos.
* Generar facturas.
* Almacenar la información de los huéspedes.
* Registrar el uso de servicios por parte de los huespedes.

**3.2.1 Componentes candidatos, estereotipos y responsabilidades.**

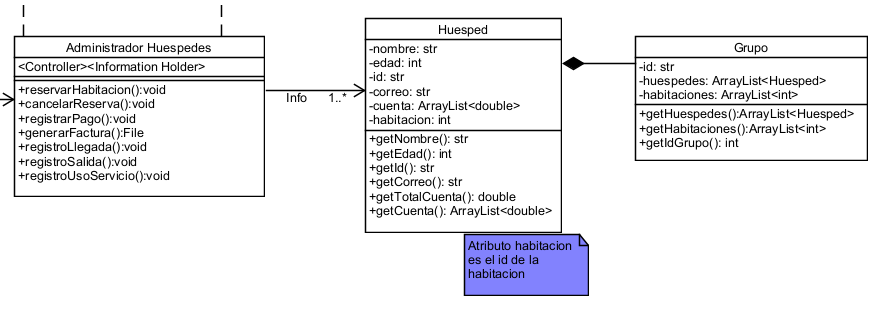
Considerando las responsabilidades que incluyen almacenar información y modificar esta, administrador de huéspedes es un “*controller*” e “*information* *holder*” con una clase y una subclase, las cuales son “*information holders*”, la clase huésped se encarga de guardar la información de cada huésped individualmente y la subclase grupo se encarga de guardar la información de un conjunto de huéspedes más específicamente su información individual y las habitaciones que ocupan. Toda esta información es guardada en un hashmap que contiene el administrador y al cual se accede con un identificador

. Figura 5: Descomposición del componente “Administrador de huéspedes”



Administrador Huéspedes

**Administrador de huespedes**



**3.2.2. Colaboraciones.**

Teniendo en cuenta las relaciones presentadas anteriormente, se definen las siguientes colaboraciones entre los componentes planteados:

El componente administrador de huéspedes no necesita colaboraciones ya que es estrictamente controlado por el componente “Administrador Huéspedes”, **a excepción de:**

1. Cuando se va a hacer una nueva reserva o se va a cancelar esta por parte del administrador de huéspedes, es necesario indicarle al administrador de habitaciones que cargue el inventario.
2. Por otro lado, una vez se realiza la reserva y se carga la información de las habitaciones, se debe configurar el id de la habitación reservada en el perfil del huésped.
3. Cuando el huésped va a hacer uso de un nuevo servicio, es necesario indicarle al administrador de servicios que retorne el inventario de estos, para así, obtener la información pertinente del servicio requerido.

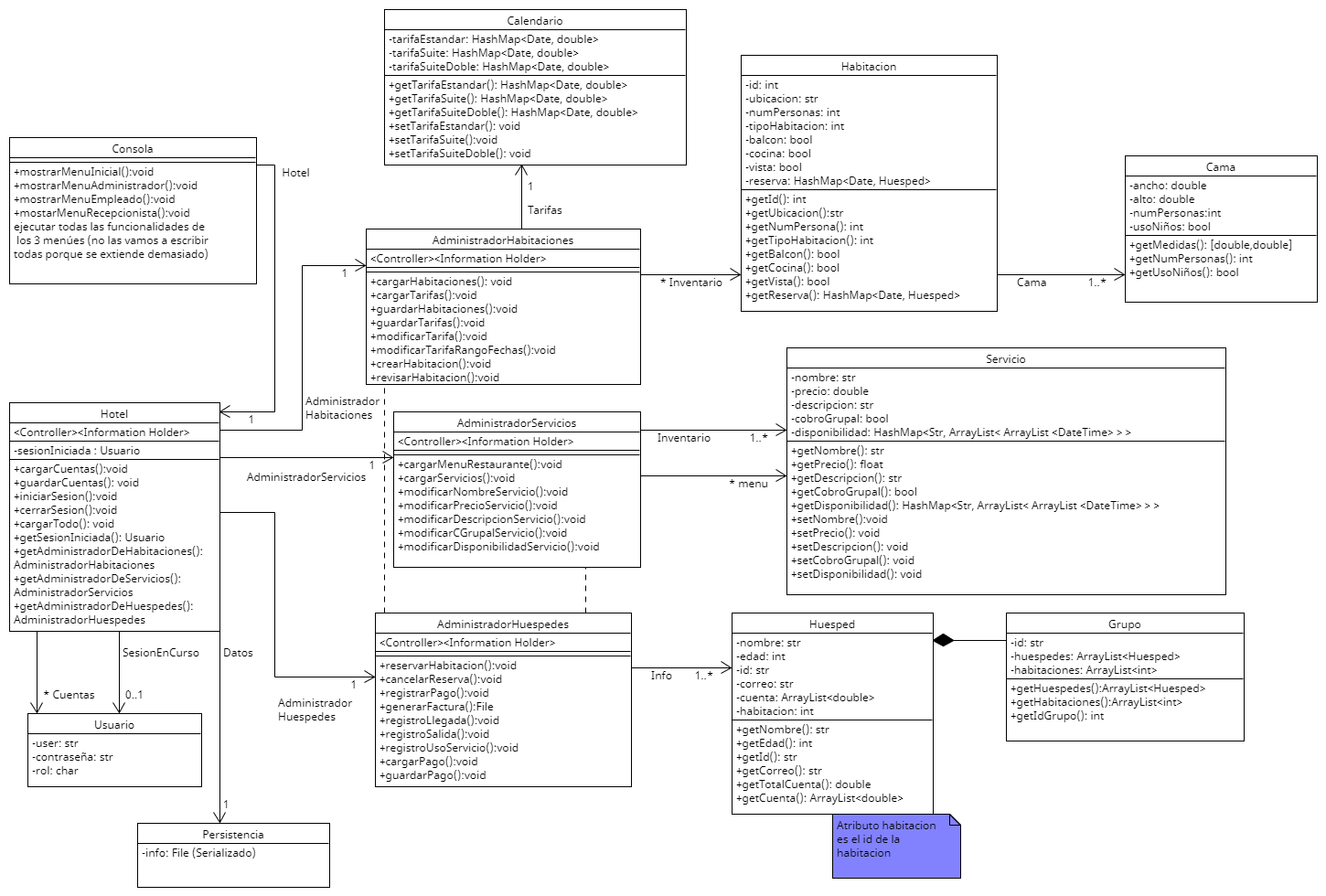
**3.3. Administrador de servicios.**

En esta sección se guarda y controla toda la información relacionada con los servicios y su uso por parte de los huespedes, lo anterior mediante las siguientes responsabilidades:

* Guardar la información de los servicios (Servicios individuales y el restaurante) y modificarla (Precios, nombres, descripciones y disponibilidad)

**3.3.1. Componentes candidatos, estereotipos y responsabilidades.**

Considerando las responsabilidades administrador de servicios tiene un componente *“controller”* e *“information holder”* el cual almacena el inventario de servicios en un hashmap y el menú del restaurante y a la vez los manipula. Esto mediante una clase llamada servicio la cual es un *"information holder”* y otra clase llamada restaurante la cual también es un *“information holder”* que contiene en un ArrayList el menú el cual está compuesto de servicios (Platos y bebidas).



Administrador Servicios

**Administrador de servicios**

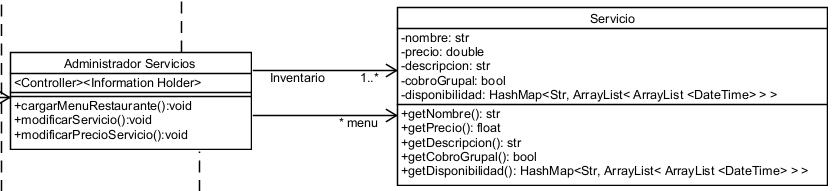


Figura 6: Descomposición del componente “Administrador de servicios”

**3.3.2. Colaboraciones.**

El componente administrador de servicios requiere las siguientes colaboraciones para su funcionamiento:

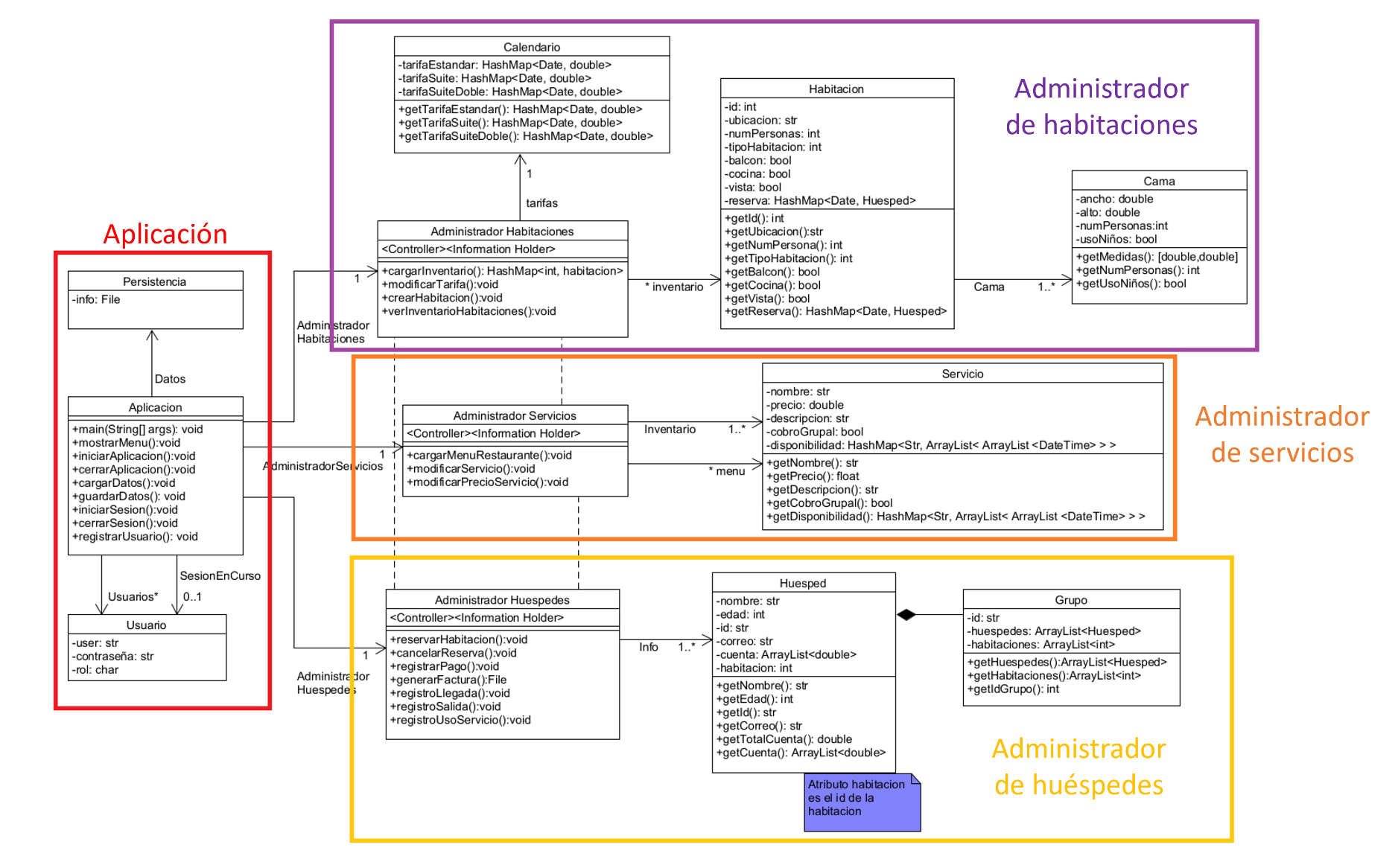
1. Cuando el administrador de servicios va a cargar el menú del restaurante, cada integrante del menú se crea como un servicio por aparte, por lo cual se requiere de la clase servicio, la cual, tras haber creado una instancia, se añade al inventario de servicios.
2. Cuando el administrador de servicios va a crear un nuevo servicio requiere de la clase servicio para configurar toda la información de este. Una vez hecho lo anterior, el nuevo servicio se añade al inventario de servicios.

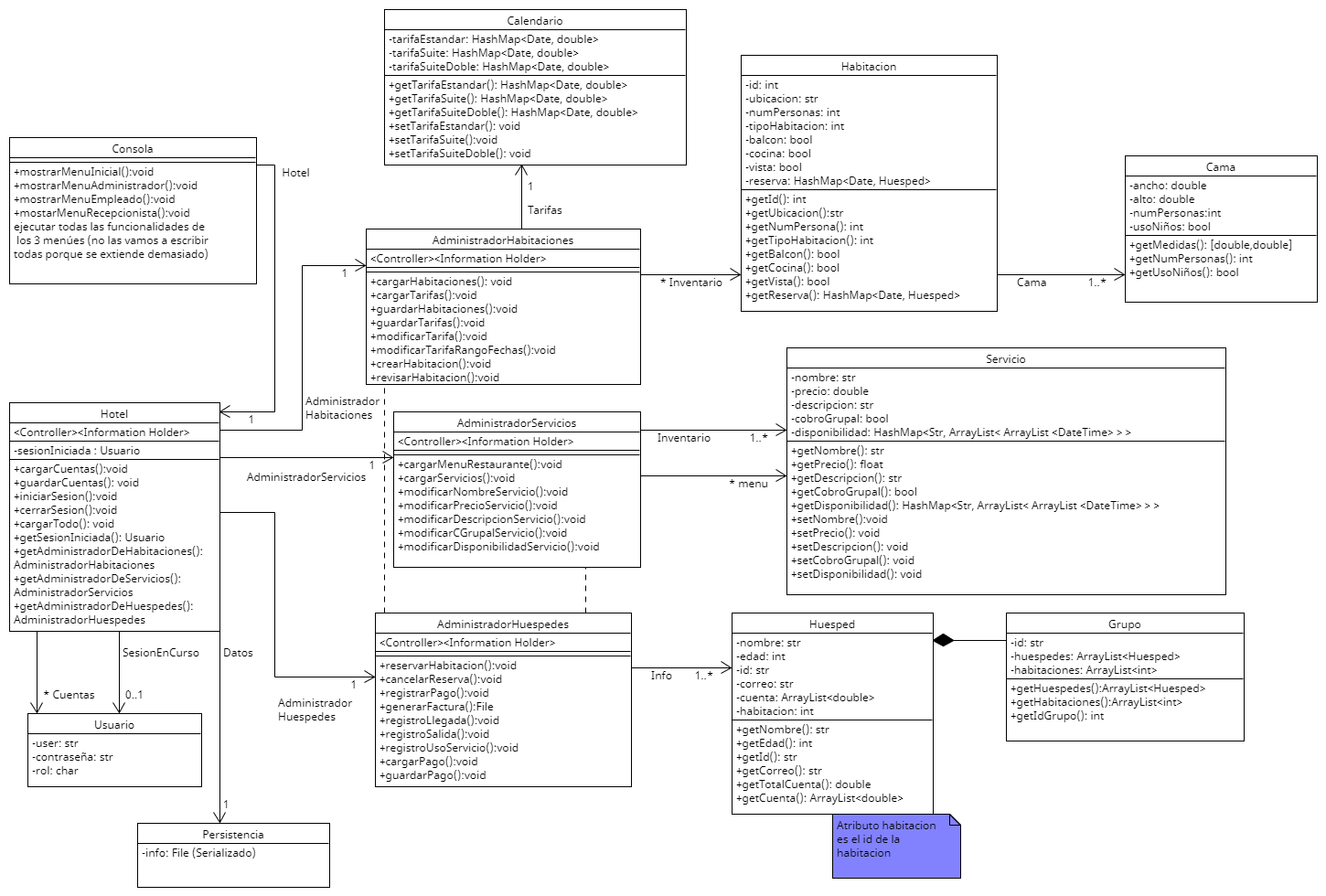
**3.4. Aplicación.**

Antes de realizar el primer borrador del UML no se habían tenido en cuenta que el programa necesita interactuar con los usuarios por consola y que además necesita tener un *login* antes de hacer uso de las opciones, junto con tener una forma de almacenar la información, estas son las responsabilidades correspondientes a este componente.

**3.4.1 Componentes candidatos, estereotipos y responsabilidades.**

Para cumplir con las responsabilidades nombradas anteriormente Aplicación tiene un componente llamado Aplicación el cual es un *“information holder”* y *“controller”,* y unas dos componentes *“information holder”* llamados Persistencia y Usuario. Estos permiten almacenar los datos en general y la información de los usuarios para el login. Mientras Aplicación se encarga de cargar los datos e interactuar con los usuarios por consola.





Aplicación

Figura 8: Descomposición del componente “Aplicación”.

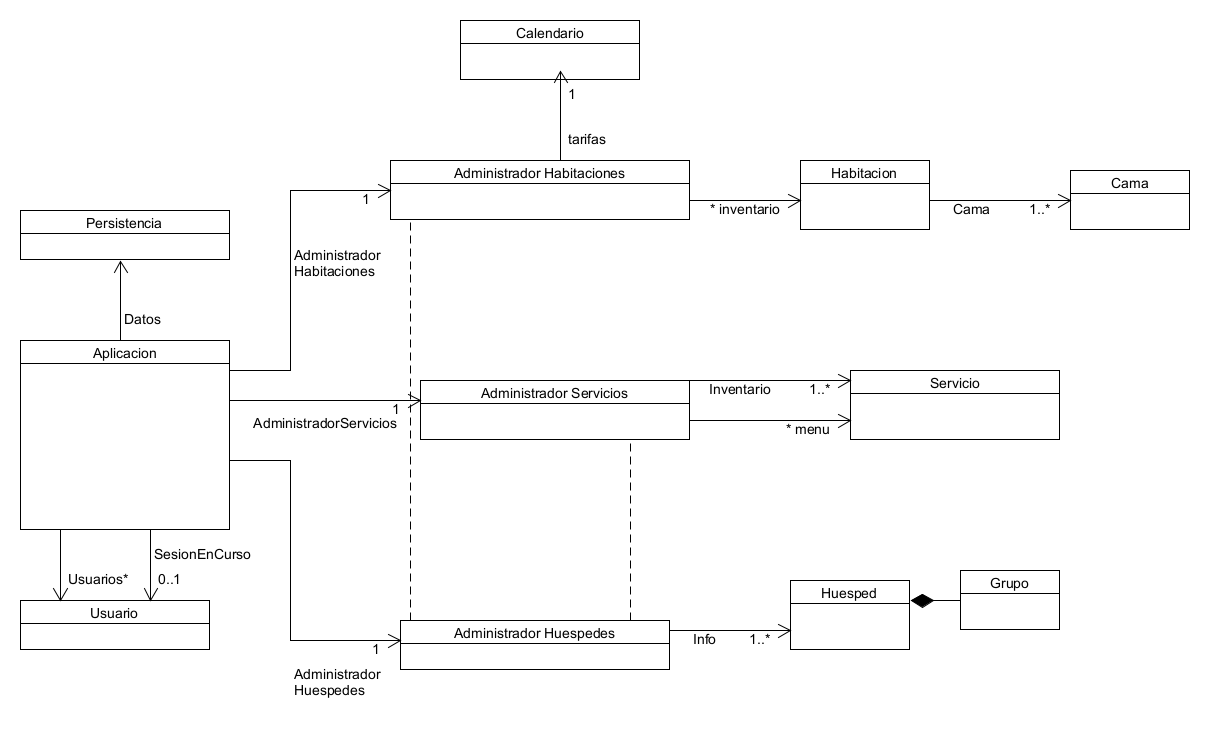
**3.4.2. Colaboraciones.**

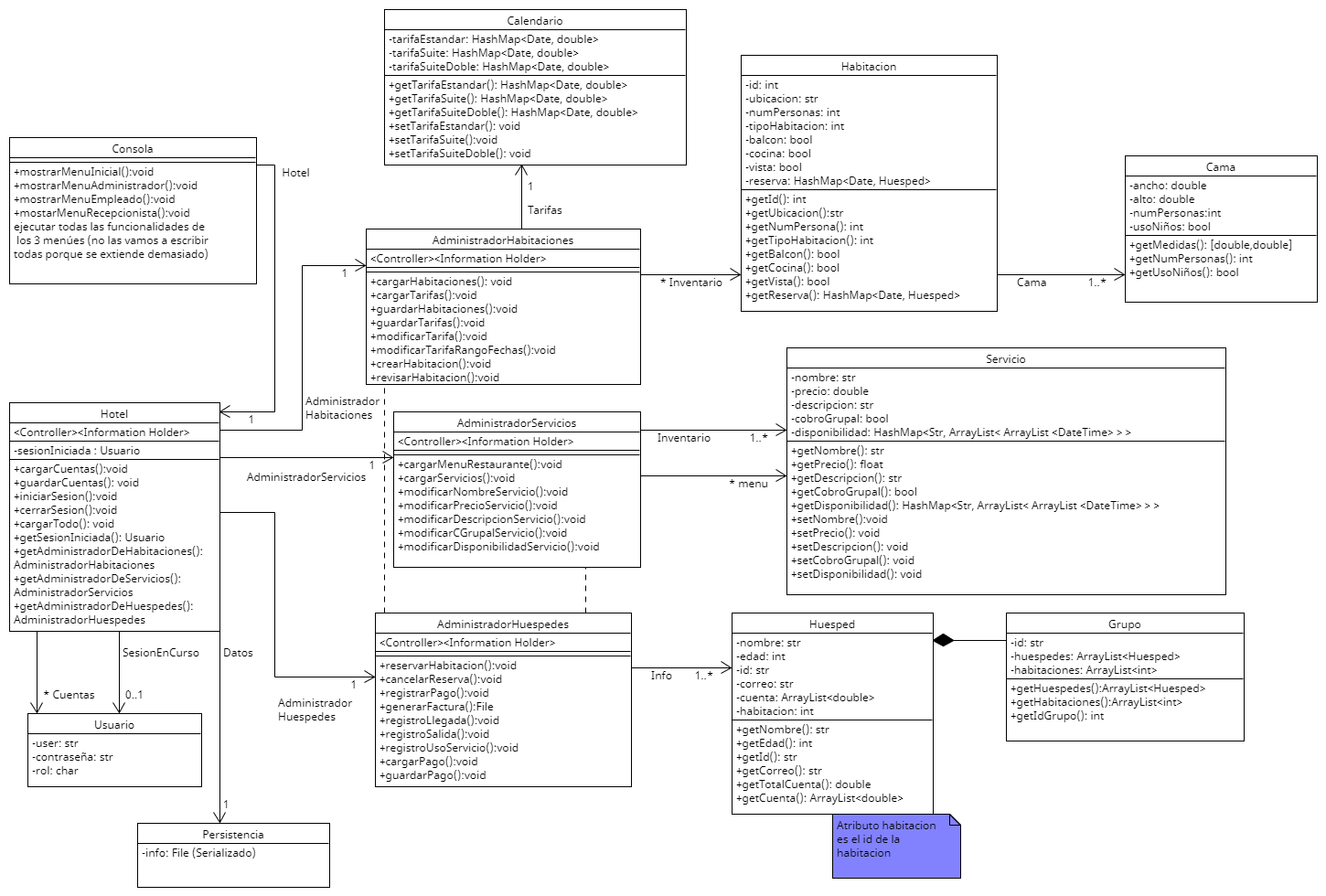
Aplicación está conectado con todos los componentes del programa, debido a que debe interactuar al momento de la carga de datos y también al usar cualquiera de los métodos del menú del programa. Aplicación recibe la instrucción y posteriormente manda la solicitud al componente requerido.

**4. Diseño Final**

Diagrama

Descripción generada automáticamenteFinalmente, en concordancia con las colaboraciones y la descomposición de cada componente, se obtiene el siguiente diseño final.





Aplicación

Administrador Servicios

Administrador Habitaciones

Administrador Huéspedes

Programa

Figura 8 y 9: Diseño final. Diagrama UML final y diagrama UML final con componentes diferenciados.

Figura 10: Diagrama de alto nivel para el diseño final.

Gráfico, Gráfico en cascada

Descripción generada automáticamente

Gráfico, Diagrama, Gráfico de cajas y bigotes

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente