Diseño DPOO Entrega #2

Santiago Cabra Chavez 202110929
Manuel Gomez Salazar 202020415
Santiago Linares 201915789

Descripción de las Restricciones:

Respecto a la restricciones técnicas del proyecto, se puede mencionar lo siguiente:

- Los archivos en los cuales se va a almacenar la información, con tal de que esta mantenga el carácter de persistencia, van a funcionar bajo un sistema serializado.

- La escritura y lectura de los mismos solo podrá realizarse si un usuario valido ha ingresado en la aplicación con sus respectivas credenciales. Cabe aclarar que existen tres tipos de usuarios (Recepcionistas, administradores y empleados) con lo cual cada usuario tendrá ciertas restricciones sobre lo que pueden y no realizar dentro del sistema.

- Para evitar la fuga de información, la carpeta de almacenamiento de los archivos serializados se encontrará fuera de donde se aloja el código fuente de la aplicación.

- Con tal de que se pueda llevar un control de auditoría se manejara la opción para el administrador de que se pueda generar un archivo log con el historial de cierto grupo de huéspedes requerido.

- La aplicación debe realizar en Java y manejar, por el momento, una interacción con los usuarios que esté basada en consola.

Requerimientos funcionales:

- Se debe de poder cargar las habitaciones de un hotel a partir de un archivo y de agregar habitaciones manualmente si se desea.
- Se debe de guardar todo lo que se hace mientras el programa está corriendo (si el programa se corre 10 veces, debe de tener en cuenta lo que se hizo la primera vez que se corrió el programa).

- El cliente debe de ser capaz de loguearse para hacer una reserva para una o más personas en unas fechas en especifico, si no hay disponibilidad se debe de informarle al cliente que no hay disponibilidad.
- El administrador se debe de poder loguear para agregar items al catálogo (servicio o restaurante) o modificar los precios ya existentes. También debe de poder agregarle items a la cuenta de alguna habitación en caso de que el cliente lo haya pedido.
- El empleado y el recepcionista deben de poder loguearse para añadir ítems del catálogo (servicio o restaurante) que el cliente haya pedido a la habitación correspondiente.
- El sistema debe de generar la factura del cliente con el precio total de todo lo que se consumió.

Historia de usuario:

Recepcionista:

Como usuario quiero reservar una habitación para mi familia y yo en el hotel, para eso me contacto con el hotel, me atiende un recepcionista al que le digo las fechas en las que quiero ir y el tipo de habitación en la que me quiero hospedar. Después le doy los datos mios y de las personas que van junto a mi (nombre, edad, correo electrónico) y con esto el recepcionista hace la reserva para mi familia y yo y me entrega un ID correspondiente a la reserva que hizo. Cuando vaya a ir al hotel puedo ir con el ID de la reserva y de la habitación y así voy a poder disfrutar de los beneficios del hotel, si mi familia o yo vamos a consumir algo se nos va a cargar a la habitación y se paga el excedente cuando vaya a hacer el checkout. Después de hacer el checkout me entregan la factura con los detalles de la reserva y de lo que consumí durante mi estadía en el hotel.

Administrador:

Yo como administrador me encargo de que todo funcione a la perfección dentro del hotel. Un día un cliente llamado Daniel hizo una reserva pero por motivos logísticos lo voy a tener que cambiar de habitación. Para esto, entro al sistema del hotel y entro a la opción de modificar una reserva e ingreso el código de la reserva de Daniel y lo cambio a una habitación que tenga la capacidad para la reserva de él (Daniel hizo una reserva para él y su familia). Después de resolver la reserva de Daniel, uno de los proveedores me contactó ya que tiene un producto nuevo que nos ofrece para empezar a venderlo en el hotel. Ya teniendo la información del nuevo producto, ingreso al sistema del hotel y entro a la opción de agregar un nuevo producto, ahí digito el nombre, el precio y demás información del producto como puede ser las restricciones y el sistema automáticamente me genera un ID que le corresponde al producto. Ya con el producto nuevo en el sistema, los empleados del hotel van a poder agregarlo a la cuenta de un cliente si el cliente se antoja de este producto.

Métodos e interfaces para la implementación:

En la clase Aplicacion<Controller>:
La clase aplicación va a ser un controller ya que todo lo que controla la aplicación va a ahí. Todos los inputs de usuario van ahí y este es el único programa que se corre.

+ main(String[], args): Void. Este método es el principal de todo el programa, es el que permite que se ejecute todo.
+ mostrarOpciones(): Void. Este método no recibe nada como parámetro y su función principal es mostrar las opciones para que el usuario use el programa.
+ ejecutarOpcion(Integer opcionSeleccionada): Void. Es el método que ejecuta la opción seleccionada del usuario la cual la recibe como un parámetro.

En la clase Hotel<Structurer>:
El hotel es un structurer ya que es donde se encuentran los métodos principales y donde se guarda la información relacionada a los usuarios, las habitaciones, el catálogo, entre otros.

- loadHotelInformation(File Catalog, File users, File roomsInventory, File bookings): Void. Este método recibe como parámetros los archivos relacionados a las habitaciones, los usuarios, el inventario de las habitaciones y los bookings, con los archivos crea las estructuras de datos necesarias para el funcionamiento del programa.
- getUsers(): ArrayList<Usuario>: Este método no recibe nada como parámetro y retorna la lista de los usuarios del hotel.
- getCatalog(): ArrayList <Catalogo>: No recibe nada como parámetro y retorna la lista de los elementos del catálogo.
- getRooms(): ArrayList<Habitacion>: No recibe parámetros y retorna la lista de habitaciones del hotel.
- getBookings(): ArrayList<Reserva>: No recibe parámetros y retorna la lista de las reservas relacionadas al hotel.
- Hotel(): Metodo constructor de la clase Hotel

Usuario <Abstract>: Va a ser usada por recepcionista, empleados y administrador.

+ logIn(): Void. Para que el administrador, recepcionista o empleado se pueda loguear en el sistema del hotel.

+ addItemRoom() Void: Agrega un producto a la cuenta de una habitación.

+ logOut() Void : Para que el administrador, recepcionista o empleado pueda salir del sistema del hotel.

+ deleteItemRoom()Void: Elimina un producto de la cuenta de una habitación.

+ getCatalogList() void : Permite ver el menú del hotel así como los servicios disponibles del mismo.

En la clase de Recepcionista:

+ modifyBooking(String bookingId): Void. El recepcionista va a poder modificar una reserva dado el String correspondiente al ID de la reserva. Modificar también implica cancelar la reserva en caso de ser necesario.

+ checkOut(): void El recepcionista va a poder cerrar la estadía de un huésped en el sistema, guardando la información de la reserva en un documento de reservas terminadas exitosamente.

+ checkIn(): void El recepcionista va a poder generar la reserva y estadía de un cliente en el sistema.

+ getBookingsList() List :El recepcionista tiene la facultad de ver todas las reservas que se encuentren activas en el hotel con el fin de ofrecerle una habitación a un huésped.

En la clase de Administrador:

+ addItemCatalog(Item itemAgregar): Void. El administrador va a poder agregar un producto con su precio respectivo al catálogo para que los clientes del hotel puedan usarlo.

+ deleteItemCatalog(Item itemEliminar): Void. El administrador va a poder eliminar un producto del catálogo del hotel.

+ modifyItemCatalog(Item itemModificar): Void. El administrador va a poder modificar un producto del catálogo (precio, disponibilidad, restricciones)

+ addRoom(Room roomAgregar): Void. El administrador va a poder agregar una habitación con su respectiva capacidad, tipo de habitación, descripción y el identificador se va a generar automáticamente para la habitación. Si se agrega una habitación esta debe de quedar guardada para que las próximas veces pueda ser usada.

+ deleteRoom(Room roomEliminar): Void. El administrador va a poder eliminar una habitación del sistema del hotel.

+ modifyRoom(Room roomModificar): Void. El administrador va a poder modificar las características de cualquier habitación del hotel.

+ getLog(): File. El administrador tiene la facultad de obtener un documento en el cual se encuentre el registro de un grupo de huéspedes para fines de auditoría.

+ newUser(User usuarioCrear): Usuario. El administrador tiene la facultad de crear un nuevo usuario en el sistema, para que en caso de que haya cambio de personal, estos puedan hacer uso del mismo a partir de su cargo.

+ modifyUser(User usuarioModificar): El administrador tiene la capacidad de modificar el tipo de usuario en el sistema. Tal que quede identificado como (recepcionista, empleado o admin)

+ deleteUser(User userEliminar): El administrador tiene la capacidad de eliminar un usuario del sistema del hotel.

+ getUsersList(): List. El administrador puede obtener una lista completa de los usuarios del sistema con sus respectivos correos y claves.

+ getBookingsList(): List. El administrador va a poder obtener una lista completa de las reservas activas del hotel en ese momento.

En la clase Empleado:

+ Se mantienen las implementaciones de la clase Usuario<Abstract>

En la clase Product<Information Holder>:
En esta clase se guarda la información relacionada a los productos que tiene el hotel.

+ Product(String name, Integer value, String locationRestrictions): Product. Toma como parámetros el nombre del servicio, el valor y las restricciones de la ubicación. Retorna un objeto tipo producto. El service ID se genera automáticamente.
+ getId(): Integer. Retorna el integer del id.

+ getName(): String. Retorna el String del nombre del producto.

+ getValue() Integer: Retorna el precio del producto en el catálogo.

+ getRestrictions(): String. Retorna las restricciones para obtener el producto.

En la clase Servicio <Information Holder>:
Es un information holder ya que se guarda información relacionada con los servicios que presta el hotel.

+ Service(String nombre, Integer value, String description): Service. Toma como parámetros el nombre del producto, el valor y la descripción. Retorna un objeto de clase Servicio. El id del servicio se genera automáticamente.

+ getId(): Integer. Retorna el ID del servicio.

+ getName(): String. Retorna el nombre del servicio.

+ getValue(): Integer. Retorna el precio del servicio en el catálogo.

+ getDescription(): Retorna la descripción que tiene el servicio en el catálogo.

En la interfaz Item <Interface>:
Se crea la interface ya que tanto servicio como producto comparten algunos de los métodos como son getID(), getName() y getvalue().

+ getId(): Integer. Retorna el id del item.
+ getName: String. Retorna el nombre del item.
+ getValue(): Integer. Retorna el valor dle item.

En la clase Habitación<Information Holder>:
Habitación es un information holder ya que se guarda la información relacionada a la habitación.

+ Room(List availableServices, Boolean occupancyStatus, Integer consumptionRecord List, Integer vlaueByNight, Integer guestCapacity, String roomType): Room. Toma los parámetros necesarios para crear la habitación y retorna una habitación con el occupancyStatys en valor true y también se genera automáticamente el id de la habitación.

+ getGuestList(): List. Retorna la lista de huéspedes alojados en la habitación

+ getAvailability(): Boolean. Retorna si la habitación está disponible en el momento o si está ocupada. Retorna valor true si está disponible o false en caso de no estarlo.

+ getTotalAccount(): Integer. Muestra el total actual de la cuenta de la habitación.

+ getValueByNight(): Integer. Muestra el valor por noche de la habitación.

+ getServices(): List. Muestra los servicios disponibles para el tipo de habitación.

+ getID(): Integer. Retorna el ID de la habitación.

+ getType(): String. Muestra el tipo de habitación.

En la clase Huésped<Information Holder>:
Es un information holder ya que ahí se guarda la información relacionada al huésped.

+ Guest(String name, Integer age, String email): Guest. Toma como parámetros el nombre, la edad y el email del huésped y crea un objeto guest. El id del huésped se genera automáticamente.
+ getAge(): Integer. Retorna la edad del huésped.

+ getName(): String. Retorna el nombre del huésped.

+ getEmail(): String. Retorna el Email del huésped.

+ getGuestID(): Integer. Retorna el documento del huésped.

+ getRoomID(): Integer. Retorna el ID de la habitación en la que se está alojando el huésped

+ getBookingID(): Integer. Retorna el ID de la reserva asociada al huésped.

En la clase Reserva<Information Holder>:
Es un information holder ya que se guarda toda la información relacionada a la reserva.

+ Booking() hashMap : Método constructor de la clase Reserva

+ getTotalAccountBooking(): Integer. Retorna el total de la reserva, incluyendo el precio total
de las habitaciones asociadas a la reserva y todos los costos extras (productos y servicios).

+ getDate (): String. Retorna las fechas de la reserva (inicio y fin de la reserva).

+ getRoomsBooking(): List. Muestra todas las habitaciones asociadas a la reserva.

+ getGuestListBooking(): List. Retorna la lista de huéspedes asociados a la reserva.

+ getBookingID(): Integer. Retorna el ID de la reserva.

Diagrama de Diseño

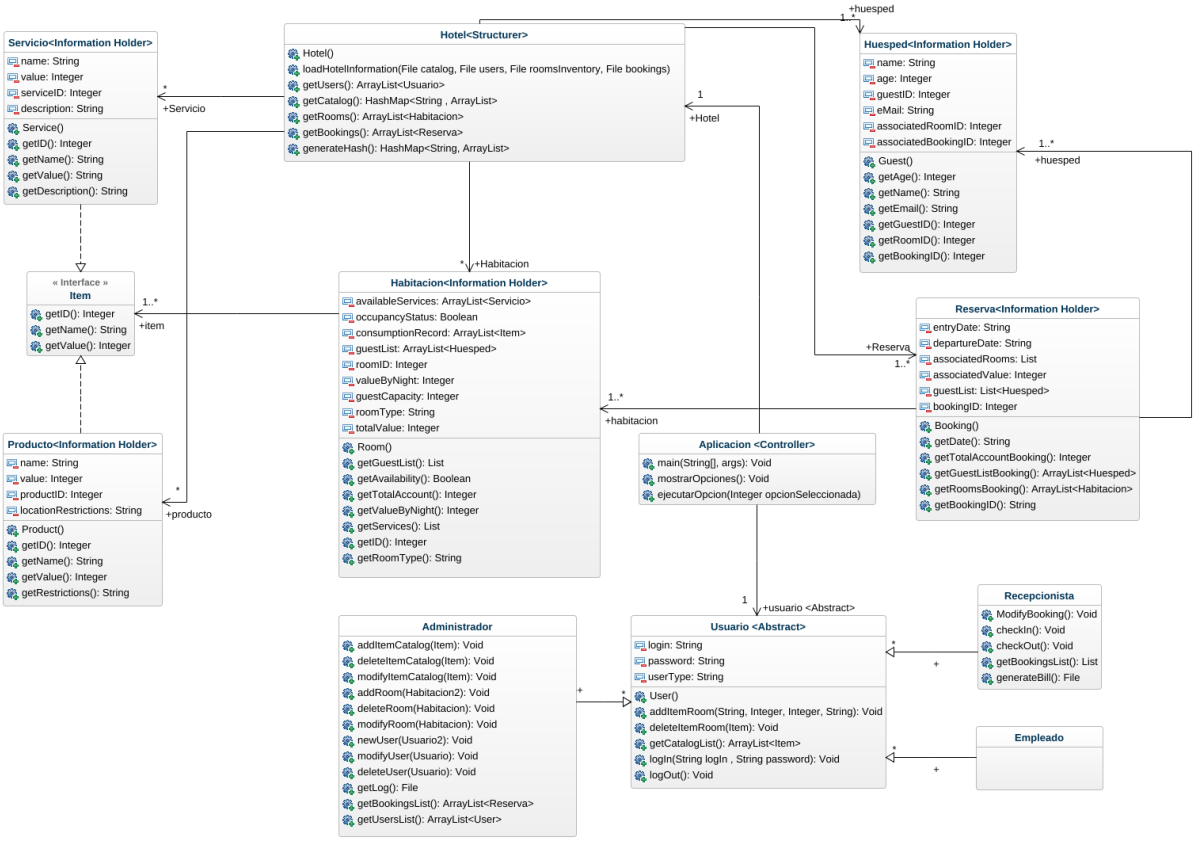

Diagrama de alto nivel

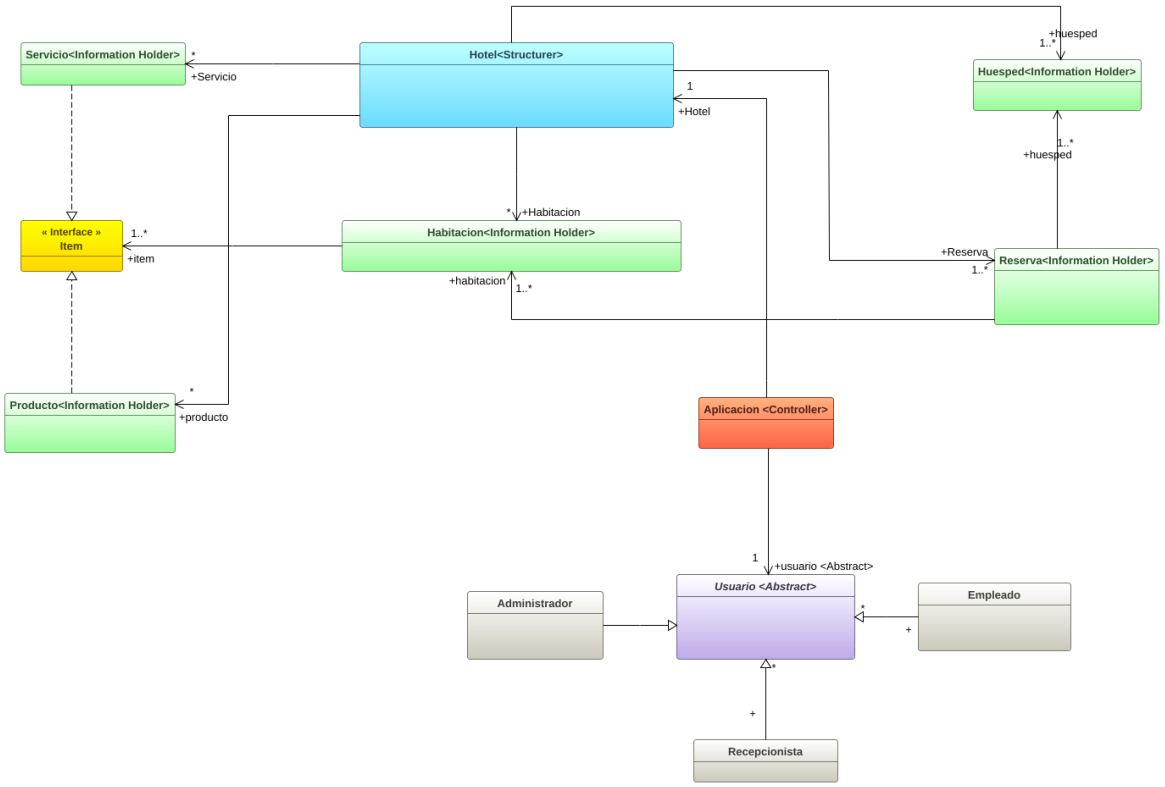

Diagrama de secuencia para una reserva

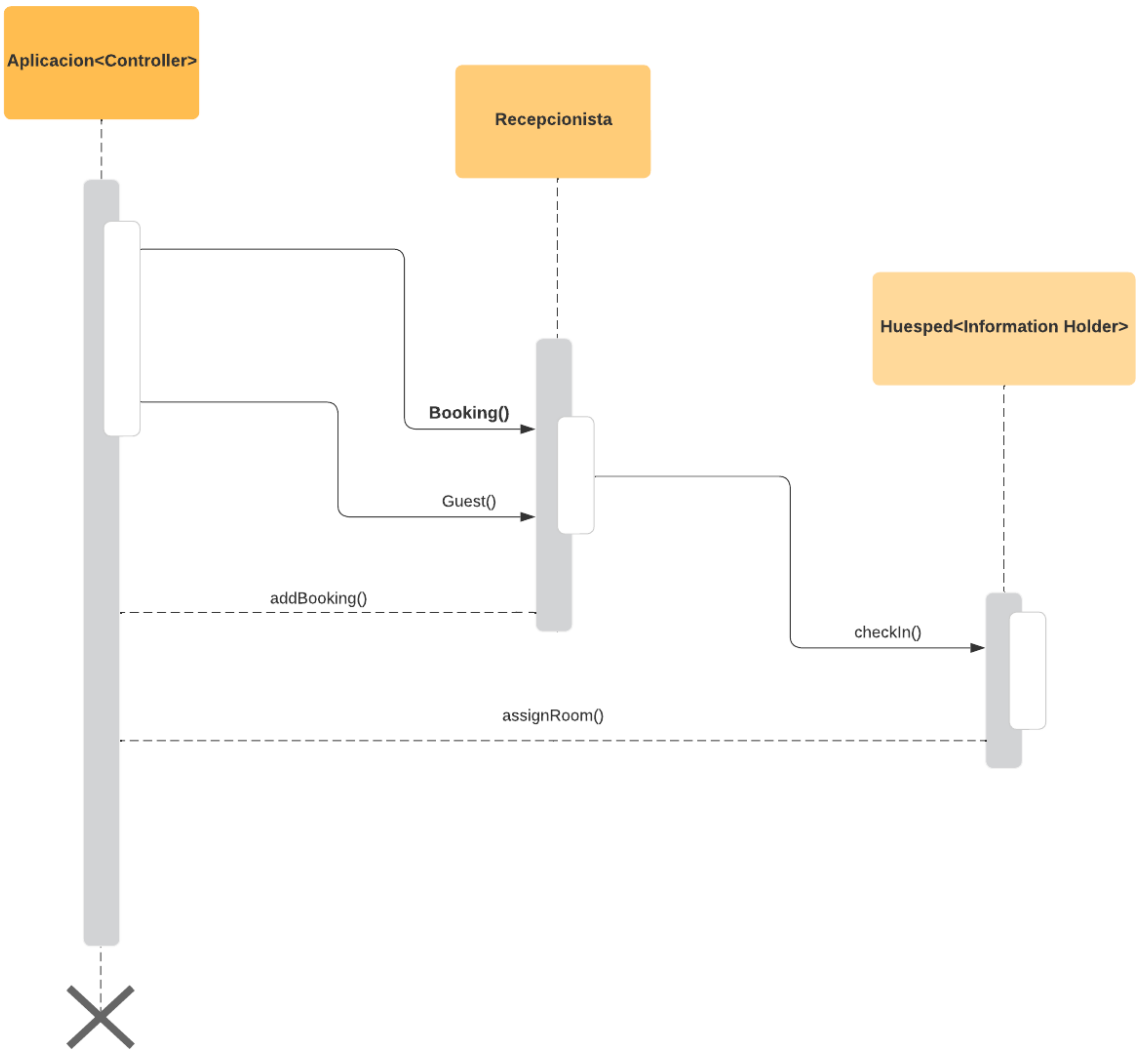
Aplicacion<Controller>
Recepcionista
Huesped<Information Holder>
Booking()
Guest()
addBooking()
checkIn()
assignRoom()

Diagrama de agregar un item a al catálogo.

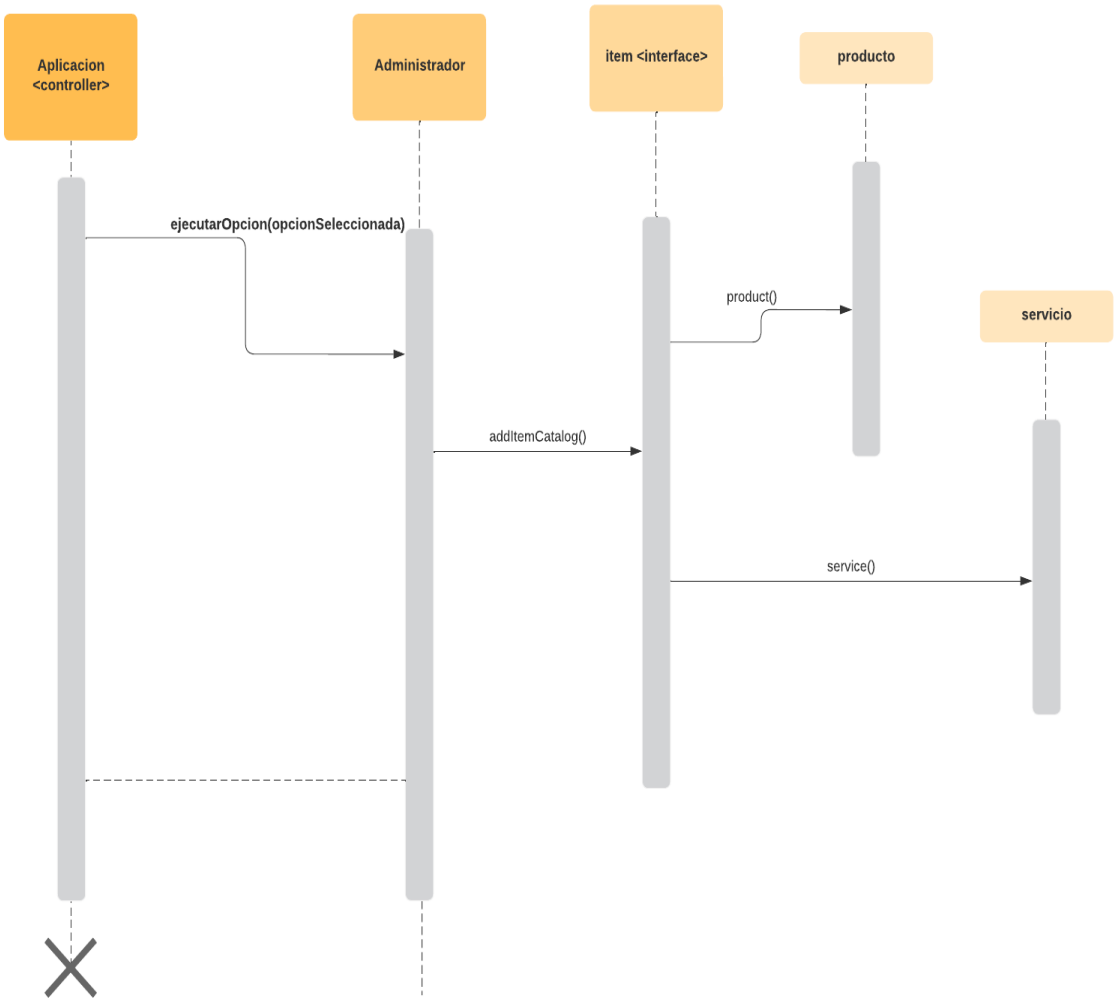

Diagrama de agregar un ítem a una habitación