**Documentación del Proyecto E1: HOTEL DE LOS ANDES, Sistemas Transaccionales**

**María Fernanda De la Hoz – 202214512**

**Juan David Obando Novoa - 202123148**

**Julián Contreras Velandia – 202211884**

## REQUERIMIENTOS FUNCIONALES

RF1 - Tipos de Usuarios

Registrar/Actualizar/Borrar/Consultar Tipos de Usuarios: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Usuario, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Usuario, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde tambien definimos estas funciones.

### RF2 - Usuarios

Registrar/Actualizar/Borrar/Consultar Usuario: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Usuario, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Usuario, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde tambien definimos estas funciones.

### RF3 - Tipos de Habitación

Registrar/Actualizar/Borrar/Consultar Tipo de Habitación: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Habitacion, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, que también podría definirse como número de habitación, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Habitación, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde tambien definimos estas funciones. Al poseer el atributo de “tipo_habitación” es posible acceder a este a través de la PK.

### RF4 - Habitaciones

Registrar/Actualizar/Borrar/Consultar Habitación: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Habitacion, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, que también podría definirse como número de habitación, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias

las colocamos en el repository de Habitación, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde tambien definimos estas funciones.

**RF5 - Servicios del Hotel**

Registrar/Actualizar/Borrar/Consultar Servicio del Hotel: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Servicio, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Servicio, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde también definimos estas funciones.

**RF6 - Planes de Consumo**

Registrar/Actualizar/Borrar/Consultar Plan de Consumo: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Plan_De_Consumo, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Plan_De_Consumo, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde también definimos estas funciones.

**RF7 - Reserva de Alojamiento**

Registrar/Actualizar/Borrar/Consultar Reserva de Alojamiento: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Reserva, junto con todos sus atributos, y manejamos el PK como un ID generado automáticamente, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Reserva, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde también definimos estas funciones.

**RF8 - Reserva de Servicios del Hotel**

Registrar/Actualizar/Borrar/Consultar Reserva de Servicios del Hotel: Para el cumplimiento de este requerimiento, primeramente, creamos la entidad de Reserva_Servicio, junto con todos sus atributos, y manejamos una PK generada de las dos llaves primarias de reserva_id y servicio_id para poder juntar estos dos atributos como diferenciador único, por lo que para las funciones de Create, Update,Delete y Search, usamos sentencias de SQL, utilizando el PK como identificador. Estas sentencias las colocamos en el repository de Reserva_Servicio, que al ser utilizado en un programa con formato MVC, va a ser llamado desde el controller, donde también definimos estas funciones.

**RF9 - Llegada de un Cliente**

Registrar/Actualizar/Borrar/Consultar Llegada de un Cliente: Para este requerimiento, manejamos el atributo check_in y check_out como booleanos en los atributos de reserva, por lo que para saber si un cliente llegó a su reserva, se debe cumplir que check_in sea True.

Estos atributos fueron especificados en la clase reserva, y luego tratados como sentencias SQL en el repositoty y controller.

**RF10 - Consumo de Servicios del Hotel**

Registrar/Actualizar/Borrar/Consultar Consumo de Servicios del Hotel: Los empleados del hotel pueden registrar el consumo de servicios por parte de un cliente o sus acompañantes, asociando cada consumo a la reserva en el atributo costo total, que este modelado como un Integer, que va a ir variando dependiendo de los servicios que tome el cliente. Este atributo esta implementado tanto en la clase, como en sentencias SQL en repository.

**RF11 - Salida de un Cliente**

Registrar/Actualizar/Borrar/Consultar Salida de un Cliente: Los recepcionistas del hotel pueden registrar la salida de un cliente, revisar los consumos realizados y generar la cuenta correspondiente. Para este requerimiento, manejamos el atributo check_in y check_out como booleanos en los atributos de reserva, por lo que para registrar la salida de un cliente, el recepcionista puede acceder a la reserva, y marcar que check_out como True. Estos atributos fueron especificados en la clase reserva, y luego tratados como sentencias SQL en el repositoty y controller.

**REQUERIMIENTOS NO FUNCIONALES**

**RNF1 - Privacidad**

Privacidad: Definimos que para privacidad, plantearíamos dentro de nuestra lógica un sistema de acceso dependiendo del tipo de usuario que tienes definido. Sin embargo, no fue posible completar esta funcionalidad.

**RNF2 - Persistencia**

Persistencia: La información manejada por la aplicación debe ser persistente, almacenada en una base de datos para garantizar la integridad y disponibilidad de los datos. Por esta razón, desde el modelo relacional se manejaron las clases con relaciones de herencia, solo con las clases que necesitan tener acceso a ciertos atributos. Además, para cerciorarnos que los datos tienen un formato correcto se especificó un tamaño de entrada y formato adecuados a los requerimientos.

**RNF3 - Distribución**

Distribución: La base de datos de la aplicación está centralizada, asegurando que todos los hoteles que utilizan el sistema tengan acceso a una única fuente de datos.

**Primera Forma Normal (1NF):**

Todos los atributos son atómicos, es decir, no están compuestos por varios valores.

Parece que todos los atributos en las tablas cumplen con esta condición. No hay atributos multivaluados ni atributos compuestos visibles.

**Segunda Forma Normal (2NF):**

Para cumplir con 2NF, la base de datos debe estar en 1NF y, además, todos los atributos no clave deben depender completamente de la clave principal.

En la tabla habitaciones, el atributo tipo_habitacion parece depender solo de la clave principal numbero_habitacion. Esto podría ser problemático si un tipo de habitación se asocia a múltiples números de habitación. Sería necesario dividir la tabla en dos para resolver esto.

En la tabla reservas, el atributo costo_total parece depender de otros atributos además de la clave principal. Esto podría ser un problema si la información sobre los precios cambia con el tiempo. Podría necesitarse una tabla separada para los detalles de la reserva, incluidos los costos.

Ya con esto podemos definir que se encuentra en 1FN.