Proyecto - Entrega 1

Leidy J. Lozano Flórez, Laura M. Restrepo Palomino, Karen T. Vera Hernández
Proyecto Sistemas Transaccionales - Caso Hotel de los Andes
Universidad de los Andes, Bogotá, Colombia
{l.lozanof, l.restrepop, k.verah}@uniandes.edu.co
Fecha de presentación: Octubre 1 del 2023

Tabla de contenido:

1. Introducción

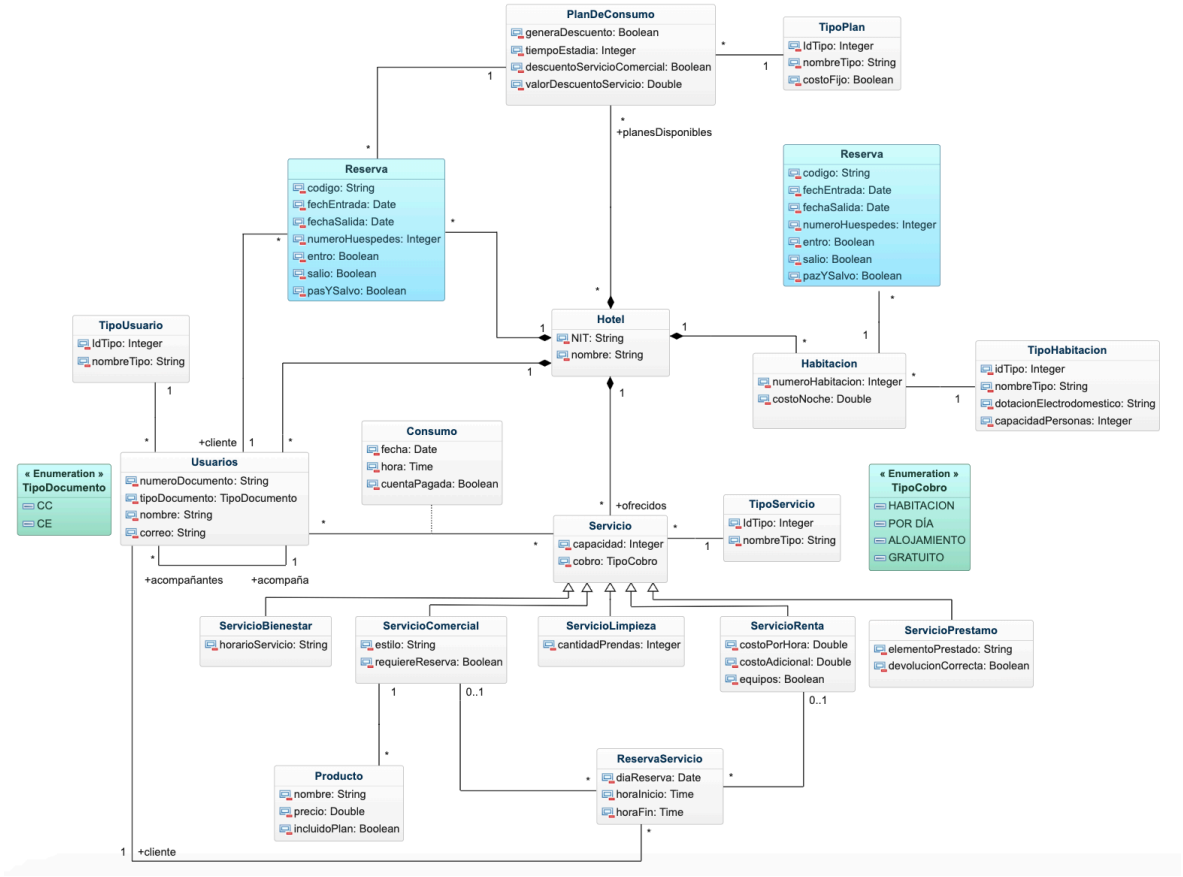
Los sistemas transaccionales son sistemas que "deben poder soportar un alto número de usuarios concurrentes y tipos de transacciones." (IBM, 2010). Para entender mejor este concepto, se analizó el caso de Hotel de los Andes, una aplicación que apoya a hoteles en sus operaciones diarias. Para ello, se realizó un proyecto JPA a partir de la creación de un modelo UML y un modelo relacional.

2. Análisis y modelo conceptual

2.1 Modelo UML

El diagrama UML de Hotel los Andes es una herramienta fundamental en el desarrollo de software y su ingeniería. Básicamente, es un lenguaje empleado por los desarrolladores, diseñadores y clientes para comunicarse de manera efectiva, comprender conceptos y diseños complejos. Por otra parte, facilitara la comprensión del funcionamiento del hotel y como se relacionan sus componentes, lo que es importante para el mantenimiento y la evolución del software en un futuro. En cuando al diseño y análisis, el diagrama permite a los diseñadores modelar la estructura y el comportamiento antes de escribir código. A su vez, los analistas comprenden los requisitos del sistema y cómo estos se traducen en la estructura y

comportamiento del software, lo que es esencial para garantizar que cumpla con los objetivos y necesidades del cliente



Inicialmente se crea una clase principal Hotel que será nuestro punto de partida, en el que le asigna un NIT y un nombre. Posteriormente, cada hotel se compone de diferentes tipos de habitaciones que definen el costo de alojamiento por noche, por lo que se crea una clase Habitación (con un numero respectivo de habitación y un costo por noche) y otra clase Tipo Habitación con un id, un nombre que define el tipo, una dotación electrodoméstica y una capacidad de personas que definirán dicho costo.

Se crea la clase Reserva:

Cada cliente reserva una habitación del hotel de acuerdo a sus necesidades (fechas en la que desea reservar, número de personas y plan de consumo), esta clase Reserva tiene un código, una fecha de entrada y salida, numero de huéspedes, un registro de entrada (Check-in, que registra en el hotel la información del cliente, de sus acompañantes y la habitación donde se va a alojar) y salida (Check-out, que registra que todos los consumos realizados por el cliente genera la cuenta correspondiente) y finalmente un paz y salvo.

Se crea la clase Usuarios:

Hotel los Andes se compone de 4 tipos de usuarios: cliente, recepcionista del hotel, empleados del hotel, administrador de datos y gerente del hotel. Por lo que se crea una clase llamada usuario en donde se reunieron todas las características de las cuales estos usuarios comparten, tales como: número documento, tipo del documento (cedula de ciudadanía o cedula extranjera), nombre y correo. Adicionalmente, se crea otra clase que define el tipo de usuario.

Se crea la clase PlanDeConsumo:

Para efecto de tener mayor ocupación, Hotel de los Andes suele proveer varios tipos de consumo, algunos ejemplos de los planes son: Larga estadía, tiempo compartido, todo incluido y promociones particulares. Creamos la clase plan de consumo, con los atributos: generaDescuento, descuentoServicioComercial, valorDescuentoServicio; Adicionalmente, se crea una clase Tipoplan que define el plan que se tomó.

Se crea la clase TipoServicio:

El hotel cuenta con varios tipos de servicio, en esta clase se reúnen las principales características que los conforman como servicios son: Id y nombre.

Se crea la clase Servicio:

Hotel de los Andes ofrece servicios adicionales como Piscina, Gimnasio, Internet, Bar, Restaurante, Supermercado, Tiendas, SPA, Lavado/planchado/embolada o Préstamo de utensilios, Salones de reuniones, Salones de conferencias; estos servicios se decidió dividirlos en conjuntos reuniéndolos por sus características, entonces tenemos: Servicio limpieza, Servicio Bienestar, Servicio Comercial, Servicio renta, Servicio préstamo y estas entidades fueron herencias de la clase Servicio. Además, al servicio comercial se le añadió una clase llamada producto ya que se trata del alquiler o préstamo de algún producto en específico. Asimismo, para las clases servicio renta y comercial se creó una clase llamada Reserva servicio, ya que esta clase de servicios requieren de una reserva puesto que son espacios exclusivos del hotel.

Es importante resaltar que los servicios fueron divididos de la siguiente manera:

- Servicios bienestar: Gimnasio, Piscina e Internet.

- Servicios comerciales: Bar, Restaurante, Supermercado, Tienda joyería, Tienda ropa y Spa.

- Servicios limpieza: Lavado, Planchado, Embolada.

- Servicios renta: Salón de reunión, Salón de conferencia.

- Servicio préstamo: Préstamo de utensilios.

Se crea la clase Consumo:

Los usuarios pueden consumir servicios y por este motivo se creó la clase consumo que contiene de fecha, hora y cuenta pagada.

Finalmente, tenemos una enumeración TipoCobro que puede ser por habitación, por día, alojamiento y gratuito.

Cardinalidades:

Hotel – Plan de consumo: varios hoteles se componen de varios planes y viceversa.

Plan de consumo – Tipo plan: Un plan de consumo si o si debe tener un único tipo, un tipo de plan puede tener 0 o 1 plan de consumo.

Hotel – Habitación: Un hotel se compone de muchas habitaciones, pero muchas habitaciones solo pueden pertenecer a 1 hotel.

Habitación – Tipo habitación: 0 o 1 habitación estrictamente tiene un tipo de habitación, un tipo de habitación puede tener uno o no habitación.

Habitación – reserva: Una habitación puede tener muchas reservas, pero muchas reservas solo pueden tener 1 habitación.

Hotel – servicio: 1 hotel se compone de muchos servicios y varios servicios componen de un hotel.

Servicio – bienestar, comercial, limpieza, renta y préstamo: son herencia de servicio, porque un servicio puede ser cualquiera de los anteriormente mencionados.

Servicio comercial – producto: un servicio puede tener muchos productos y varios productos solo tienen un servicio.

Servicio comercial – reserva servicio: 0 o 1 servicio comercial puede tener muchas reservas y varias reservas pueden tener 1 o no servicio comercial.

Servicio renta – reserva servicio: 0 o 1 servicio renta pueden tener muchas reservas y varias reservas pueden tener 1 o no servicio renta.

Reserva servicio – usuarios: Una reserva de un servicio siempre tiene asociado un cliente, y un cliente puede tener muchas (o ninguna) reservas de un servicio.

Hotel – usuarios: El hotel se compone de varios usuarios y varios usuarios componen un hotel.

Hotel – reserva: El hotel se compone de varias reservas y varias reservas componen a un único hotel.

Plan de consumo – reserva: un plan puede tener muchas reservas, pero muchas reservas pueden tener únicamente un plan.

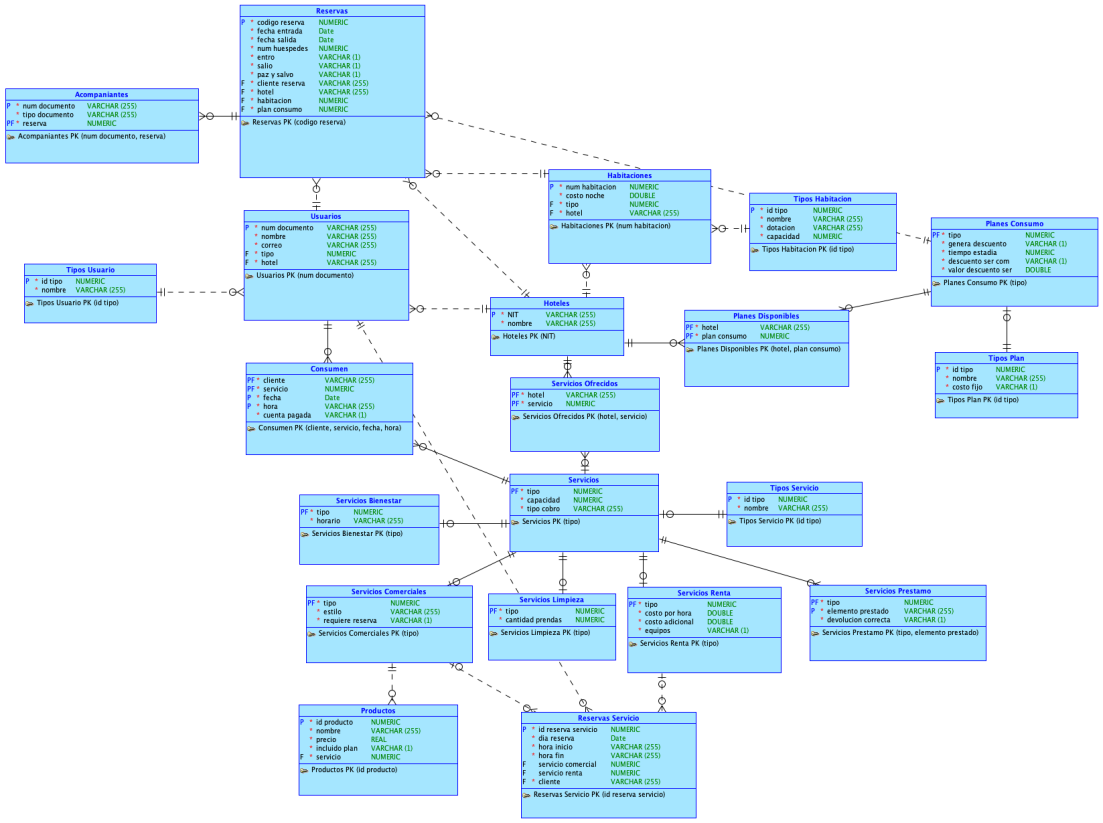
Usuarios – reserva: un usuario puede tener muchas reservas, pero muchas reservas pueden tener únicamente un usuario.

Usuarios – tipo usuarios: 0 o 1 usuario pueden tener únicamente un tipo de usuario.

Usuarios – servicios: Muchos usuarios pueden tener muchos servicios.

2.2 Modelo E/R

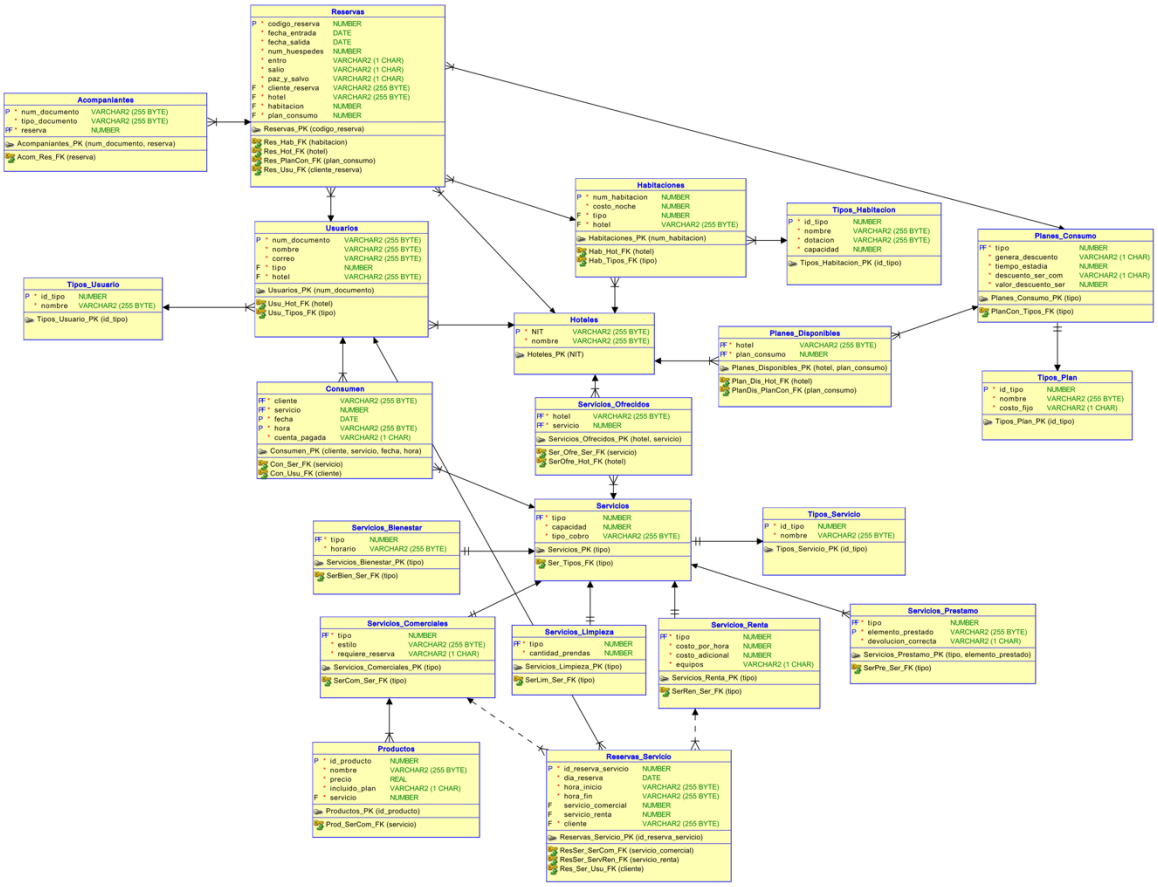
El modelo entidad relación es una herramienta fundamental para Hotel los Andes, ya que este permite representar la realidad del hotel, organizando todas las entidades que son relevantes en el contexto y las relaciones entre ellas. Además, ayuda a definir, y contribuir a la integridad y consistencia de la estructura de la base de datos.



Este modelo fue creado automáticamente haciendo uso de la herramienta OracleDataModeler y función de ingeniería inversa a partir del modelo relacional, que será explicado a continuación.

2.3 Modelo relacional

El modelo relacional es una herramienta fundamental para Hotel los Alpes, ya que este permite representar la realidad del hotel en la base de datos mediante tablas, y la relación entre ellas. Como se mencionó, este modelo fue creado en OracleDataModeler.



Para hacer este modelo, fue necesario ver las asociaciones existentes entre las clases del modelo UML. Para aquellas clases que tuvieran una asociación con nombre propio, se creó una tabla adicional (como es el caso para Acompaniantes). Luego, antes de hacer la conexión entre cada tabla, se definieron las tablas individualmente con sus atributos propios (que son los mismos del UML). También estos atributos fueron catalogados como obligatorios. Después se crearon las relaciones mediante FKs teniendo en cuenta la cardinalidad:

- 0..* – 0..* → Se crea una tabla adicional que conecta a las dos clases. Su PK es FKs de ambas tablas.
- 0..* – 1 → En la tabla de la izquierda se añade una FK obligatoria.
- 0..* – 0..1 → En la tabla de la izquierda se añade una FK no obligatorio.
- Composiciones y agregaciones se toman como asociaciones normales.

3. Diseño de la base de datos

Para diseñar la base de datos, simplemente se usó la herramienta OracleDataModeler para generar el DDL. Con este script generado automáticamente, en la base de datos se pegaron las sentencias SQL y se corrieron todas al tiempo. Vale la pena añadir que, para el caso de los valores generados automáticamente (como IDs y códigos), se crearon secuencias manualmente que fueron ejecutadas antes de poblar la BD.

4. Normalización

Este procedimiento contribuirá a reducir la duplicación innecesaria de datos, lo que a su vez mejorará la integridad de la base de datos al garantizar la precisión, coherencia y confiabilidad de los datos. Además, aumentará la eficiencia y la consistencia de la base de datos.

Reservas: Está en Boyce Codd porque no tiene atributos multivalores, no tiene dependencias parciales ni transitivas y, finalmente, el atributo código puede determinar cualquier otro atributo (es una superllave).

Acompaniantes: Está en 1NF porque hay una dependencia parcial.

Usuarios: Está en 1NF porque existen dependencias parciales.

Tipo usuario: Está en Boyce Codd porque con el atributo id se puede determinar cualquier otro atributo (es una superllave).

Habitaciones: Está en 1NF porque existe una dependencia parcial: con el número de habitación y el tipo se puede obtener el costo; pero teniendo únicamente el tipo también se puede saber qué costo tiene.

Tipos habitación: Está en Boyce Codd.

Hoteles: Está en Boyce Codd.

Servicios ofrecidos: Está en Boyce Codd.

Servicios: Está en Boyce Codd.

Servicios bienestar: Está en Boyce Codd.

Servicios comerciales: Está en 1NF porque existe la siguiente dependencia parcial: si se tiene el tipo y estilo se puede saber si requiere reserva, pero sabiendo únicamente el tipo también se puede saber si requiere reserva.

Servicios limpieza: Está en 1NF.

Servicios renta: Esta en 3NF.

Servicios préstamo: Está en 1NF.

Productos: Está en 1NF porque es un atributo multivalor (es un string en donde cada producto está separado con coma).

Reservas servicio: Está en Boyce Codd.

Tipo servicio: Está en Boyce Codd.

Consumen: Está en Boyce Codd.

Planes disponibles: Está en Boyce Codd.

Planes consumo: Está en Boyce Codd.

Tipos plan: Está en 1NF porque tiene dependencia parcial.

5. Resultados logrados

Se tiene una aplicación funcional para todos los RFs.

6. Resultados no logrados

Si bien el código funciona a partir de los modelos creados, el diseño de la BD puede mejorar. Por otro lado, también hizo falta agregar manejo de errores (*try catch*). Esto es importante en, por ejemplo, casos en que se estén incumpliendo reglas de PKs, FKs, checks, etc. De momento, cuando esto sucede, en el front se muestra un error de mappeo, lo cual no es muy amigable con el usuario.

7. Balance del plan de pruebas

Para realizar las pruebas, y para facilitar el proceso, simplemente se consultaban/creaban/editaban/borraban las tablas en la aplicación del navegador y, una vez se haya hecho cualquier acción, con una sentencia SELECT sencilla se probaba que los datos fueran consistentes con lo mostrado en la aplicación y en la BD. Sin embargo, para el caso de servicios y planes de consumo, como no existe un front para estos tipos, en la BD directamente se crearon tipos para luego asignarlos a un servicio o plan de consumo nuevo.

También para probar la adición de una reserva de alojamiento y una reserva de servicio, al ingresar fechas ocupadas, se esperaba que saliera un error de mappeo.

8. Supuestos adicionales

No se supuso nada aparte de lo mencionado en la interpretación de la realidad de Hotel de los Alpes en el modelo UML.

9. Elementos complementarios

Además de todos los elementos mencionados en este informe, es importante notar que existen documentos complementarios que permiten un mejor entendimiento del proyecto y su desarrollo. Estos pueden ser encontrados en la carpeta *docs* y son los siguientes:

- Modelos en formato fuente y pdf.
- Archivos SQL (esquema, población y pruebas).
- Documentación de los requerimientos funcionales.
- Documentación del proyecto de SW.
- Contribución de trabajo.

10. Conclusiones

A partir del caso Hotel de los Alpes, se pudo entender la importancia de una implementación adecuada de sistemas transaccionales, a partir del desarrollo de modelos y un proyecto JPA.

11. Bibliografía

1. Transaction management systems on z/OS. (2010). Recuperado el 30 de septiembre de 2023, de https://www.ibm.com/docs/en/zos-basic-skills?topic=mainframe-what-is-transaction-system