“Iteración 3: – Desarrollo del Caso de Estudio de Alohandes”

Lina María Gómez Mesa, Eduardo José Herrera Alba
Iteración 3 del Proyecto de Sistemas Transaccionales
Universidad de los Andes, Bogotá, Colombia
{l.gomez1, ej.herreraa}@uniandes.edu.co
Fecha de presentación: Abril 11 de 2023

Tabla de contenido

1 Modelo Conceptual - Análisis 1
1.1 Diagrama de Clases UML 1
1.2 Configuración de la Base de Datos 2
1.3 Normalización del Modelo Relacional 7
2 Lógica de los nuevos requerimientos a Desarrollar 7
2.1 Registrar los operadores de alojamiento para AlohAndes 7
2.2 Registrar propuestas de alojamiento para AlohAndes 8
2.3 Registrar las personas habilitadas para utilizar los servicios 8
2.4 Registrar una reserva 9
2.5 Cancelar una Reserva 10
2.6 Retirar una oferta de alojamiento 10
2.7 Registrar una Oferta Colectiva 11
2.8 Cancelar Reserva Colectiva 12
2.9 Deshabilitar Oferta de Alojamiento 12
2.10 Rehabilitar Oferta de Alojamiento 13
2.11 Mostrar el dinero recibido por cada proveedor (RFC1) 14
2.12 Mostrar las 20 ofertas más populares (RFC2) 14
2.13 Mostrar el Índice de Ocupación (RFC3) 15
2.14 Mostrar los alojamientos disponibles (RFC4) 15
2.15 Mostrar el uso de AlohAndes para cada tipo de usuario (RFC5) 16
2.16 Mostrar el uso de Alohandes para un usuario dado (RFC6) 17
2.17 Analizar la Operación de Alohandes (RFC7) 17
2.18 Mostrar el uso de Alohandes para un usuario dado (RFC8) 18
2.18 Encontrar los clientes frecuentes (RFC8) 19
2.19 Encontrar las Ofertas de Alojamiento que no tengan mucha demanda (RFC9) 19
3 Documentación de Cambios de Cambios en Diseño 20
4 Resultados Logrados 21
5 Resultados No Logrados 22
6 Supuestos Adicionales 22
7 Balance de Pruebas 22
A continuación se muestra el plan de pruebas para los requerimientos funcionales: RF7, RF8, RF9, RF10. Estos incluyen caso de éxito y casos de fallo. 22
7.1 Registrar una Oferta Colectiva (RF7) 22
7.2 Cancelar Reserva Colectiva (RF8) 24

7.4 Rehabilitar Oferta de Alojamiento (RF10) 27
8 Bibliografía 29

1 Modelo Conceptual - Análisis

1. 1.1 Diagrama de Clases UML

Se realizó la revisión del caso de estudio propuesto: Alohandes. A continuación, se muestra un diagrama de clases de acuerdo a lo identificado en el enunciado. El modelo conceptual tiene integradas las reglas de negocio mediante anotaciones, las clases propuestas que son elementos del mundo del negocio y las cardinalidades de las asociaciones.

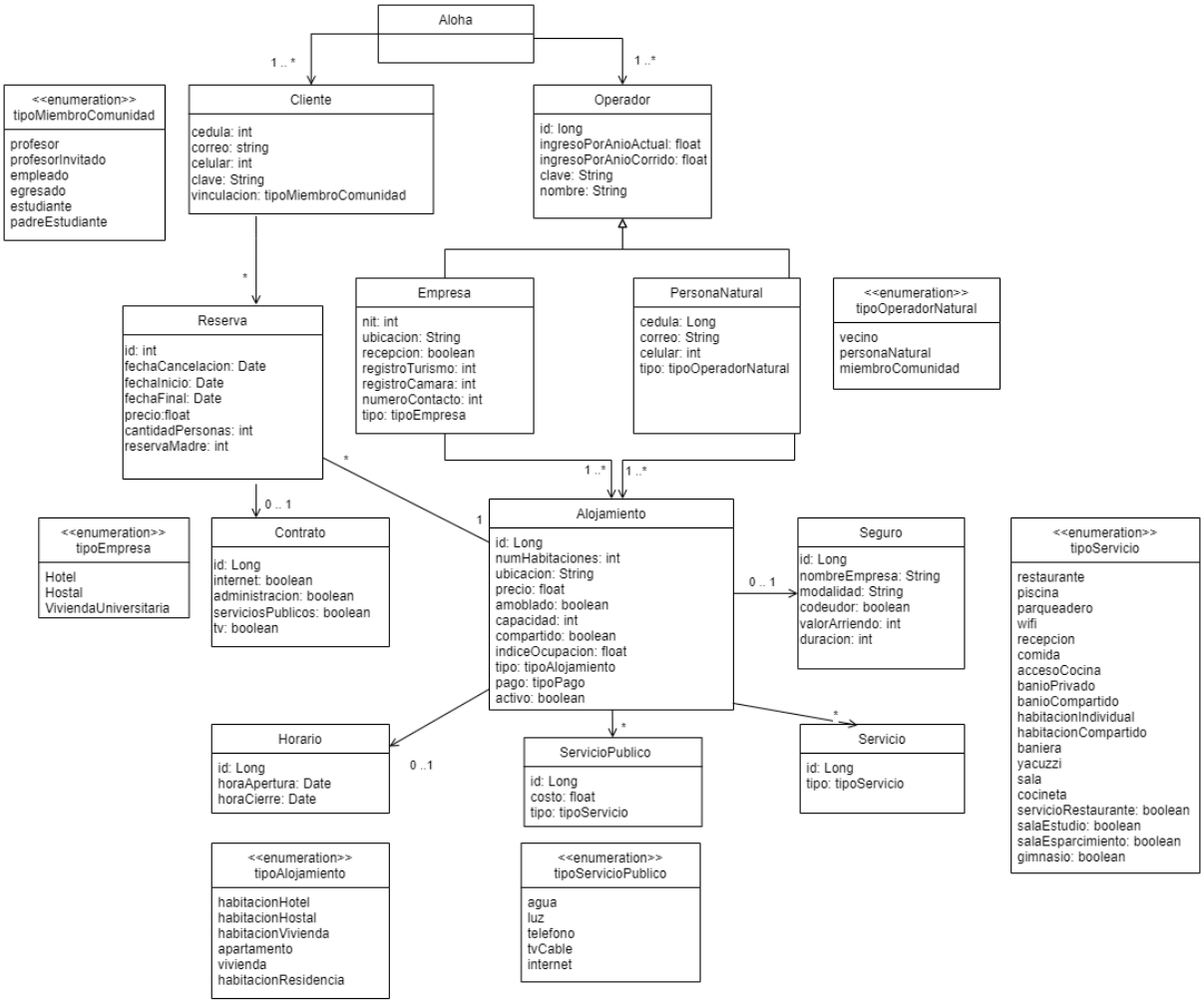

Figura 1. Modelo conceptual completo de 12 clases para el sistema transaccional de AlohaAndes [Elaborado en Diagrams.Net]

Las enumeraciones que aparecen en el diagrama tienen valores que tiene el sistema inicialmente. Sin embargo, es posible agregar nuevos valores en el modelo relacional. Teniendo en cuenta el RNF2, todas las clases deberían ser persistentes, con excepción a Aloha, la cual es una clase de carácter conceptual. Las clases del modelo del mundo que fueron actualizadas para la iteración 3 fueron: *Operador, Reserva y Alojamiento*.

Algunas restricciones de acuerdo al enunciado son:

1. Las empresas deben cumplir con el registro en la cámara de comercio y en la superintendencia de turismo.
2. Los clientes de Alohandes deben tener algún vínculo con la institución: estudiantes, egresados, empleados, profesores, padres de estudiantes, profesores invitados, personas registradas en eventos de Uniandes.
3. Una persona no puede reservar más de un alojamiento en un mismo día.
4. Un alojamiento no acepta reservas que superen su capacidad.
5. El alojamiento en vivienda universitaria sólo está habilitado a estudiantes, profesores, empleados y profesores visitantes.

2. 1.2 Configuración de la Base de Datos

Se muestra a continuación el modelo de datos relacional que corresponde al modelo conceptual que se propuso. Las tablas de algunas relaciones son demasiado extensas para una sola línea, por lo que se partieron para conservar la legibilidad. El modelo se compone de las siguientes relaciones:

Tabla 1. Relación de TipoMiembroComunidad

TipoMiembroComunidad

Id	Nombre
PK,SA	NN, CK[profesor,empleado,egresado,estudiante,padreEstudiante], ND

Tabla 2. Relación de Cliente

Cliente

Cedula	Correo	Nombre	Celular	Vinculacion	Clave
PK,UA	NN,UA	NN,UA	CK[=10], NN,UA, ND	FK[tipoMiembroComunidad.Id],UA	NN,UA

Tabla 3. Relación de Operador

Operador

Id	IngresoPorAnioActual	IngresoPorAnioCorrido	Nombre	Clave
PK, UA	CK[>=0], UA	CK[>=0],UA	NN,UA	NN,UA

Tabla 4. Relación de TipoOperadorEmpresa

TipoOperadorEmpresa

Id	Nombre

SA	NN,CK[Hotel, Hostal,ViviendaUniversitaria], UA, ND

Tabla 5. Relación de Empresa

Empresa

Nit	Ubicacion	Recepcion	RegistroTurismo	Registro Camara	NumeroContacto	Tipo
PK, FK[Operador.Id],UA	NN, UA	NN, CK[0,1],UA	NN, CK[=9],UA	NN, CK[=9],UA	NN,CK[=10],UA	FK[TipoOperadorEmpresa.Id], NN, UA

Tabla 6. Relación de TipoOperadorNatural

TipoOperadorNatural

Id	Nombre
PK,SA	CK[Vecino,personaNatural,miembroComunidad], ND, UA

Tabla 7. Relación de PersonaNatural

PersonaNatural

Cedula	Correo	Celular	Tipo
PK, FK[Operador.Id],UA	NN,UA	CK[=10], NN, UA	FK[TipoOperadorNatural.Id], UA, NN

Tabla 8. Relación de Contrato

Contrato

Id	internet	administracion	serviciosPublicos	tv
PK, SA	CK[0,1], UA, NN	CK[0,1], UA, NN	CK[0,1], NN, UA	CK[0,1], UA,NN

Tabla 9. Relación de Seguro

Seguro

Id	NombreEmpresa	Modalidad	Codeudor	ValorArriendo	Duracion
PK, SA	NN, UA	NN, UA,	NN, CK[0,1]	NN	NN, CK[3,6,12]

		CK[presencial, digital]			

Tabla 10. Relación de Horario

Horario

Id	HoraApertura	HoraCierre
PK,SA	NN, UA	NN, UA

Tabla 11. Relación de TipoAlojamiento

TipoAlojamiento

Id	Nombre	TipoPago
PK, SA	CK[habitacionHotel,habitacionHostal,habitacionVivienda,apartamento,vivienda,habitacionResidencia], UA, ND	CK[mes, dia]

Tabla 12. Relación de Alojamiento

Alojamiento

Id	NumHabitaciones	Ubicacion	Precio	Amoblado	Capacidad
PK	NN, CK[>0]	NN	NN, CK[>0]	NN, CK[0,1]	NN, CK[>0]

Compartido	IndiceOcupacion	Tipo	Horario	Seguro	Operador
NN, CK[0,1]	NN	FK[TipoAlojamiento.Id]	FK[Horario.Id]	FK[Contrato.Id]	FK[Seguro.Id]

Activo
NN, CK[0,1]

Tabla 13. Relación de Reserva

Reserva

Id	fechaCancelacion	fechaInicio	fechaFinal	Alojamiento	Cliente	Contrato	Precio
PK, SA	UA	NN, UA, CK[fechaInicio <fechaFinal]	NN,UA	FK[Alojamiento.Id], UA	FK[Cliente.cedula], UA	FK[Contrato.Id], UA	NN, UA, CK[>=0]

cantidadPersonas	reservaMadre
NN	

Tabla 14. Relación de TipoServicioPublico

TipoServicio

Id	Nombre
PK,SA	CK[restaurante, piscina, ... (ver UML), gimnasio], UA

Tabla 15. Relación de ServicioPublico

ServicioPublico

Id	Costo	Tipo	Alojamiento
PK, SA	NN	FK[TipoServicio.Id]	FK[Alojamiento.Id]

Tabla 16. Relación de TipoServicio

TipoServicio

Id	Nombre
PK,SA	CK[restaurante, piscina, ... (ver UML), gimnasio], ND

Tabla 17. Relación de Servicio

Servicio

Id	Tipo	Alojamiento
PK, SA	FK[TipoServicio.Id]	FK[Alojamiento.Id]

Los cambios realizados al modelo fueron: en la relación modelo Reserva se añadieron los atributos *cantidadPersonas* y *reservaMadre* (para satisfacer los requerimientos RF7 y RF8). Además se cambió la restricción sobre el atributo precio para el requerimiento RF9. Por otro lado, para la relación Alojamiento se agregó un nuevo atributo *Activo* indicando si este alojamiento está activo (1) o inactivo (0) (esto con el fin de cumplir los requerimientos funcionales de habilitar un alojamiento RF9 y deshabilitar un alojamiento RF10). Finalmente, se alteraron las restricciones sobre las columnas *ingresoPorAnioActual* e *ingresoPorAnioCorrido* para el correcto funcionamiento general (se pudo haber hecho desde la iteración pasada).

3. 1.3 Normalización del Modelo Relacional

El modelo relacional planteado se encuentra en el tercer nivel de normalización. A continuación, se justifica nivel por nivel:

Primera forma normal: Un modelo relacional de datos se encuentra en la primera forma normal si todos los atributos de cada una de las relaciones tienen dominios atómicos. El modelo planteado cumple con la primera forma normal dado que no hay atributos multivalor. Esto se puede observar en la sección 3.1 de este documento.

Segunda Forma Normal: Un modelo relacional de datos se encuentra en la segunda forma normal si para cada una de sus relaciones se cumple que no existen dependencias funcionales parciales desde algún atributo primo a un atributo no primo. De acuerdo a lo anterior, el modelo planteado sí cumple con la segunda forma normal porque se encuentra en primera forma normal y no existen dependencias parciales desde los atributos primos; puesto que, las llaves primarias no son compuestas y por lo tanto no hay subconjuntos propios de atributos a los cuales aplicar esta restricción.

Tercera Forma Normal: En tercer lugar, el modelo propuesto cumple con la tercera forma normal debido a que se encuentra en segunda forma normal y no existen dependencias transitivas entre atributos no primos. En el caso de este modelo, los conjuntos de llaves candidatas son de cardinalidad 1, solo tienen la llave primaria que, como se dijo antes, consta de solo un atributo, por lo general un número. Por ende, en este caso particular, todo atributo que no es la llave primaria es un atributo no primo.

Forma normal de Boyce-Codd: Un modelo relacional de datos se encuentra en la cuando no existen dependencias parciales entre atributos primos y además no hay varias llaves candidatas compuestas que no son disjuntas. En el caso de este modelo, el conjunto de llaves candidatas de cada relación consta de solo un atributo: el identificador respectivo (ya sea un id, número de identificación o NIT). Por ende, no hay dependencias parciales entre atributos primos (pues en cada relación hay un único atributo primo, el ya mencionado identificador) y además no hay llaves candidatas compuestas del todo.

4.

2 Lógica de los nuevos requerimientos a Desarrollar

5. 2.1 Registrar los operadores de alojamiento para AlohAndes

En la Tabla 1 se muestra el primer requerimiento funcional de AlohAndes. Este se centra en registrar a los operadores de alojamiento para AlohAndes.

Tabla 1. Registrar los operadores de alojamiento para AlohAndes

Nombre	RF1. Registrar a los operadores de alojamiento para AlohAndes
Resumen	El sistema debe permitir el registro de los roles de usuario, tipo operador, definidos por el negocio: empresa (Hotel, Hostal, ViviendaUniversitaria) o personaNatural (vecina, personaNatural, miembroComunidad).
Entradas	
Long: Operador.id	
String: Operador.clave	
Float: Operador.IngresoPorAnioActual	
Float: Operador.IngresoPorAnioCorrido	
String: nombre	
Atributos dependiendo del tipo de Operador creado: Empresa o PersonaNatural	
Resultados	
Enviar Mensaje: String: Registro exitoso	
Creación de: nueva tupla en relación Operador	
Creación de: nueva tupla en relación Empresa o PersonaNatural	
RNF asociados	
Persistencia: AlohAndes debe garantizar que se realice de manera correcta el registro del nuevo operador de alojamientos en la base de datos. Esto es importante ya que esto debería ser poder consultado a futuro y operador no debería tener que volver a registrarse en futuras ocasiones.	
Aislamiento - Concurrencia: AlohAndes debe garantizar que el registro de un nuevo operador de alojamiento sean operaciones que deben ser realizadas aisladamente para que, en caso de que haya varios operadores inscribiéndose al tiempo, no haya problemas de integridad de datos en la base de datos.	

6. 2.2 Registrar propuestas de alojamiento para AlohAndes

En la Tabla 2 se muestra el segundo requerimiento funcional de Alohandes. Este se centra en registrar propuestas de alojamiento para Alohandes.

Tabla 2. Registrar propuestas de alojamiento para Alohandes

Nombre	RF2. Registrar propuestas de alojamiento para Alohandes
Resumen	La aplicación debe permitir el registro de propuestas de alojamiento por parte de los distintos *operadores* (personaNatural o Empresa). Cada propuesta debe incluir información detallada sobre el alojamiento de acuerdo a los atributos solicitados.
Entradas	
Long: Operador.Id	
String: Operador.Clave	

Atributos del Alojamiento que se pueden ver en la Tabla 8 de la sección 3 en la entidad Alojamiento.
Resultados
Enviar Mensaje: String: Registro exitoso
Creación de: nueva tupla en relación Alojamiento
Creación de: nueva tupla en relaciones que lo necesiten (Horario, Seguro, ServicioPúblico, Servicio).
RNF asociados
Atomicidad(transaccionalidad): Se necesita que los registros de las propuestas de alojamiento se realicen por completo o de plano no se realicen. Esto se debe a que se necesita que un alojamiento no puede prescindir de información como su ubicación, capacidad, tipo de alojamiento, entre otros.
Consistencia(transaccionalidad): Se necesita que el registro de un alojamiento se vea reflejado en la base de datos de una sola forma. No se puede tener un mismo alojamiento con, por ejemplo, 2 ubicaciones diferentes.
Distribución(enunciado): Se necesita que el administrador de datos tenga contacto con una única base de datos centralizada con la información de los alojamientos. De esta forma, la información que se muestra es igual para los roles de usuario, en especial el administrador.

7. 2.3 Registrar las personas habilitadas para utilizar los servicios

En la Tabla 3 se muestra el tercer requerimiento funcional de Alohandes. Este se centra en registrar las personas habilitadas para utilizar los servicios.

Tabla 3. Registrar propuestas de alojamiento para Alohandes

Nombre	RF3. Registrar a las personas habilitadas para utilizar los servicios
Resumen	El sistema debe permitir el registro de nuevos clientes para que utilicen los servicios proveídos por la aplicación. Cada persona debe incluir información personal necesaria para la inscripción de acuerdo a los atributos solicitados.
Entradas	
Atributos de la relación de cliente que se pueden ver en la Tabla 8 de la sección 3 en la entidad Cliente	
String: Cliente.Clave	
Resultados	
Enviar Mensaje: String: Registro exitoso	
Creación de: nueva tupla en relación Cliente	
RNF asociados	
Persistencia: AlohAndes debe garantizar que se realice de manera correcta el registro del nuevo operador de alojamientos en la base de datos. Esto es importante ya que esto debería ser poder consultado a futuro y el operador no debería tener que volver a registrarse en futuras ocasiones.	
Aislamiento - Concurrencia: AlohAndes debe garantizar que el registro de un nuevo operador de alojamiento sean operaciones que deben ser realizadas aisladamente para que, en caso de que haya varios operadores inscribiéndose al tiempo, no haya problemas de integridad de datos en la base de datos.	

8.

9. 2.4 Registrar una reserva

En la Tabla 4 se muestra el cuarto requerimiento funcional de Alohandes. Este se centra en registrar una reserva de acuerdo a las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta … etc.

Tabla 4. Registrar una Reserva

Nombre	RF4. Registro de una reserva

Resumen	La aplicación debe permitir que los *clientes* de Alohandes puedan buscar alojamiento y reservar una estancia del de su elección. Esta debe considerar las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta, etc.
Entradas	
Long: Cliente.Id	
String: Cliente.Clave	
Atributos de Reserva que se pueden ver en la Tabla 8 de la sección 3 en la entidad Reserva	
Atributos de entidades que se relacionan con una Reserva que lo necesiten (Alojamiento, Servicio, ServioPúblico)	
Resultados	
Enviar Mensaje: String: Registro exitoso	
Creación de: nueva tupla en relación Reserva	
RNF asociados	
Durabilidad (transaccionalidad): Para el cumplimiento de este requerimiento funcional se necesita persistencia. Esto se debe a que en el momento en que se genera una reserva se necesita tener la reserva vigente durante el tiempo que dure la reserva para dicho cliente y quitar el alojamiento y su disponibilidad en ese periodo.	
Isolation- Concurrencia (Enunciado): Debido a que varios usuarios pueden intentar realizar una reserva al mismo tiempo, es necesario garantizar que la aplicación maneje correctamente la concurrencia para evitar problemas de consistencia en la base de datos.	
Privacidad (enunciado): La privacidad es importante para garantizar que los usuarios solo puedan manipular y consultar la información que les es propia, evitando así posibles vulneraciones de seguridad y privacidad.	

10.

11. 2.5 Cancelar una Reserva

En la Tabla 5 se muestra el quinto requerimiento funcional de Alohandes. Este se centra en cancelar una reserva.

Tabla 5. Cancelar una Reserva

Nombre	RF5. Cancelar una reserva
Resumen	La aplicación debe permitir que un cliente que ha hecho una reserva en el sistema pueda cancelar dicha reserva antes del inicio de la misma.
Entradas	
Long: Cliente.Id	
Long: Reserva.Id	
String: Cliente.Clave	
Resultados	
Enviar Mensaje: String: Cancelación exitosa	
Deleción de: tupla en relación Reserva	
RNF asociados	
Durabilidad (transaccionalidad): Para el cumplimiento de este requerimiento funcional se necesita persistencia. Esto se debe a que en el momento en que se cancela cierta reserva se necesita que quede guardada en la base de datos para que se pueda consultar la fecha de cancelación en futuras oportunidades.	
Privacidad: Se debe garantizar que el usuario que quiere cancelar una reserva solo pueda acceder a información propia, mas no de otros usuarios, pues se debe garantizar la privacidad de estos.	

12.

13. 2.6 Retirar una oferta de alojamiento

En la Tabla 6 se muestra el sexto requerimiento funcional de Alohandes. Este se centra en cancelar una reserva de acuerdo a las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta … etc.

Tabla 6. Retirar una oferta de Alojamiento

Nombre	RF6. Retirar una oferta de Alojamiento
Resumen	La aplicación debe permitir que los *operadores* de ALOHA puedan retirar una oferta de alojamiento que ya no esté disponible. Este requerimiento implica que la información de las ofertas retiradas se actualice correctamente en la base de datos de la aplicación, garantizando que no se muestran alojamientos que ya no están disponibles para reservar.
Entradas	
Long: Alojamiento.Id	
Atributos de entidades que se relacionan con la entidad Alojamiento que lo necesiten (Contrato,Seguro, Horario)	
String: Operador.Clave	
Resultados	
Enviar Mensaje: String: Cancelación exitosa	
Deleción de: tupla en relación Alojamiento	
RNF asociados	
Persistencia (enunciado): La acción de retirar la oferta de Alojamiento implica una actualización en la base de datos de la aplicación con tal de quitar la oferta y que no se puedan hacer reservas a futuro.	
Seguridad (transaccionalidad): Se requiere que los únicos permitidos para retirar ofertas de alojamiento sean los operadores de acuerdo a sus condiciones actuales y disponibilidad. Esto se debe a que estos son los responsables de publicar ofertas.	
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que los clientes realicen reservas cuando ya no está disponible el alojamiento.	

14.

15. 2.7 Registrar una Oferta Colectiva

En la Tabla 7 se muestra el séptimo requerimiento funcional de Alohandes. Este se centra en crear una reserva colectiva de acuerdo a las preferencias del cliente. Este es únciamente válido para clientes de Alohandes. Una vez este haya ingresado con sus credenciales, se le pide la cantidad de reservas que desea realizar, las fechas de las reservas y se descompone en reservas individuales. Anteriormente, se envían las posibles opciones de alojamiento y se va ubicando cada reserva en uno de ellos. En caso de que se encuentre ocupado, se busca en el siguiente alojamiento. Si recorre todos los alojamientos y no encuentra un posible candidato retorna que no se pudo hacer la reserva.

Tabla 7. Registrar una oferta Colectiva

Nombre	RF7. Registrar una oferta Colectiva
Resumen	En casos de eventos masivos el usuario indica el tipo de alojamiento deseado y la cantidad deseada, y revisa si está en capacidad de satisfacer esa solicitud, eventualmente con varios proveedores, y en caso afirmativo realizar las reservas individuales correspondientes.
Entradas	
Long: Cliente.Id	

String: Cliente.Clave
Long: cantidadReservas
LocalDate: Reserva.FechaInicio
LocalDate: Reserva.FechaFinal
String: Alojamiento.Tipo
List<Servicio> serviciosDeseados
Resultados
Enviar Mensaje: String: Reserva Exitosa / String: Reserva No Exitosa
Inserción de: tupla en relación Alojamiento (tanto de la reserva colectiva como cada una de las reservas individuales)
RNF asociados
Persistencia (enunciado): La acción de retirar la oferta de Alojamiento implica una actualización en la base de datos de la aplicación con tal de quitar la oferta y que no se puedan hacer reservas a futuro.
Seguridad/ Privacidad (transaccionalidad): Se requiere que los únicos permitidos para generar ofertas colectivas sean personas que son clientes de Alohandes.
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que los clientes realicen reservas se conserva tanto las reservas colectivas e individuales en caso de éxito.
Concurrencia: La aplicación se ejecuta en el nivel estándar de Read Committed y utiliza tx.begin, tx.rollback y tx.commit en caso de ser necesario.

16.

17. **2.8 Cancelar Reserva Colectiva**

En la Tabla 8 se muestra el octavo requerimiento funcional de Alohandes. Este se centra en cancelar una reserva colectiva de acuerdo a las preferencias del cliente. Inicialmente, se le pide el id de la reserva colectiva y luego, se cancela tanto la individual como la colectiva al ponerles la fecha de cancelación. Se retorna si se pudo "cancelar con éxito".

Tabla 8. Registrar una oferta Colectiva

Nombre	RF8. Cancelar una oferta Colectiva
Resumen	En casos de eventos masivos el usuario indica el id de la reserva colectiva y cancela las subreservas (les pone fecha de cancelación).
Entradas	
Long: Cliente.Id	
String: Cliente.Clave	
Long: Reserva.id (id de la reserva colectiva)	
Resultados	
Enviar Mensaje: String: Cancelación de la reserva exitosa / No exitosa.	
Cancelación de: tupla en relación Reserva (tanto de la reserva colectiva como cada una de las reservas individuales). Se les pone de fechaCancelacion.	
RNF asociados	
Persistencia (enunciado): La acción de retirar la reserva colectiva de Reserva implica una actualización en la base de datos de la aplicación con tal de ponerle fecha de cancelación.	
Seguridad/ Privacidad (transaccionalidad): Se requiere que los únicos permitidos para cancelar reservas colectivas sean personas que son clientes de Alohandes y que fueron quienes crearon dicha reserva.	
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que la reserva ya no aparezca sin fecha de cancelación en el sistema.	

Concurrencia: La aplicación se ejecuta en el nivel estándar de Read Committed y utiliza tx.begin, tx.rollback y tx.commit en caso de ser necesario.

18.

19. **2.9 Deshabilitar Oferta de Alojamiento**

En la Tabla 9 se muestra el noveno requerimiento funcional de Alohandes. Este se centra en deshabilitar una oferta de alojamiento por razones externas como, por ejemplo, remodelación. Inicialmente, se le pide el id del operador y su contraseña, luego, se le pide id del alojamiento que quiere deshabilitar. Como subproceso, se verifica que dicho alojamiento si sea de ese operador, y luego se comienza la transacción dándole prioridad a las reservas vigentes y próximas. Dichas reservas se pasan a alojamientos con el mismo tipoAlojamiento. Se utiliza el requerimiento descrito en la tabla 4 (RF4) para reubicar cada una de las reservas vigentes y próximas y, posteriormente, se cancelan del alojamiento deshabilitado (RF5). En caso de que no se pueda reubicar, se le indica al operador qué reserva no se logró reubicar. Además, en caso de que dicho alojamiento tuviera reservas colectivas se desagregan para poderse reubicar en otro alojamiento.

Tabla 9. Deshabilitar una oferta de alojamiento

Nombre	RF9. Deshabilitar una oferta de alojamiento
Resumen	Por motivos externos una oferta de alojamiento debe cesar su funcionamiento, entonces se reubican las reservas que en dicho momento se encuentran vigentes y próximas a realizarse. En caso de que no se pueda reubicar se le indica al usuario.
Entradas	
Long: Operador.Id	
String: Operador.Clave	
Long: Alojamiento.id (id del alojamiento a deshabilitar)	
Resultados	
Enviar Mensaje: String: Cancelación de la reserva exitosa / No exitosa.	
Reubicación de: tuplas (*) en relación Reserva que pertenecían a ese Alojamiento.id.	
RNF asociados	
Persistencia (enunciado): La acción de deshabilitar un alojamiento implica una actualización en la base de datos de la aplicación con tal de que en la Relación Alojamiento se cambie el parámetro de Alojamiento.Activo por 0 de inactivo.	
Seguridad/ Privacidad (transaccionalidad): Se requiere que los únicos permitidos para deshabilitar alojamientos sean personas que son operadores de Alohandes y que son dueños de dicho alojamiento en cuestión.	
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que el alojamiento ya esté inactivo y las reservas hayan sido cambiadas a otro alojamiento.	
Concurrencia: La aplicación se ejecuta en el nivel estándar de Read Committed y utiliza tx.begin, tx.rollback y tx.commit en caso de ser necesario.	

20.

21. **2.10 Rehabilitar Oferta de Alojamiento**

En la Tabla 10 se muestra el décimo requerimiento funcional de Alohandes. Este se centra en rehabilitar una oferta de alojamiento que se encuentra en el momento inactiva. Inicialmente, se le pide el id del operador y su contraseña, luego, se le pide id del alojamiento que quiere volver a habilitar. Como subproceso, se verifica que dicho alojamiento si sea de ese operador, y luego se comienza la transacción para poder update el parámetro de Alojamiento.Activo a 1.

Tabla 10. Rehabilitar una oferta de alojamiento

Nombre	RF10. Deshabilitar una oferta de alojamiento
Resumen	Ocurre cuando la oferta de alojamiento vuelve a estar disponible y puede por lo tanto aceptar nuevas reservas.
Entradas	
Long: Operador.Id	
String: Operador.Clave	
Long: Alojamiento.id (id del alojamiento a rehabilitar)	
Resultados	
Enviar Mensaje: String: Rehabilitación del alojamiento (se debe entregar un 1 de activo)	
RNF asociados	
Persistencia (enunciado): La acción de rehabilitar reservas de un alojamiento implica una actualización en la base de datos de la aplicación con tal de que en la Relación Alojamiento se cambie el parámetro de Alojamiento.activo por el nuevo alojamiento al que pertenece cada reserva individual.	
Seguridad/ Privacidad (transaccionalidad): Se requiere que los únicos permitidos para rehabilitar alojamientos sean personas que son operadores de Alohandes y que son dueños de dicho alojamiento en cuestión.	
Durabilidad (transaccionalidad): Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos deben persistir a través de fallos del sistema y otros problemas. En especial, que la reserva ya no aparezca sin fecha de cancelación en el sistema.	
Concurrencia: La aplicación se ejecuta en el nivel estándar de Read Committed y utiliza tx.begin, tx.rollback y tx.commit en caso de ser necesario.	

2.

1. 2.11 Mostrar el dinero recibido por cada proveedor (RFC1)

En la Tabla 11 se muestra el primer requerimiento funcional de consulta de Alohandes. Este se centra en mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido.

Tabla 11. Mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido

Nombre	RFC1. Mostrar el dinero recibido por cada proveedor de alojamiento durante el año actual y el año corrido
Resumen	La aplicación debe poder mostrar el dinero que cada proveedor registrado en la base de datos ha recibido en el año actual y en el año corrido
Entradas	
Long: Operador.Id	
String: Operador.Clave	
Resultados	
Enviar Mensaje: String: valores de dinero del año actual y del año corrido	
RNF asociados	
Concurrencia: se debe garantizar que varios usuarios de la base de datos puedan acceder simultáneamente a la información y no haya problemas de concurrencia	
Persistencia (transaccionalidad): La información del dinero recibido en el año corrido y en el año actual de los operadores debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de los operadores debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos.	

2.

3. 2.12 Mostrar las 20 ofertas más populares (RFC2)

En la Tabla 12 se muestra el segundorequerimiento funcional de consulta de Alohandes. Este se centra en mostrar las 20 ofertas más populares.

Tabla 8. Mostrar las 20 ofertas más populares

Nombre	RFC2. Mostrar las 20 ofertas más populares
Resumen	El sistema debe permitir la operación de consulta por parte de los *cliente*s y *operadores* de Alohandes. Esta se centra en consultar las 20 ofertas más populares de Alojamiento de acuerdo a la cardinalidad de la tabla Reserva.
Entradas	
Long: Operador.Id o Long: Cliente.Id	
String: Cliente.Clave o String: Operador.Clave	
Resultados	
Enviar Mensaje: List<Alojamiento>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento y el operador dueño	
RNF asociados	
Concurrencia (enunciado): Pueden hacerse más de una consulta de este tipo al tiempo.	
Persistencia (transaccionalidad): La información de las ofertas y su popularidad debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de las ofertas y su popularidad debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos. Por lo tanto, esta debe concordar con el número de reservas que ha tenido cada alojamiento.	

4. 2.13 Mostrar el Índice de Ocupación (RFC3)

En la Tabla 13 se muestra el tercer requerimiento funcional de consulta de AlohAndes. Este se centra en mostrar el índice de Ocupación de cada una de las ofertas de alojamiento registradas.

Tabla 13. Mostrar el Índice de Ocupación de cada una de las ofertas de alojamiento registradas

Nombre	RFC3. Mostrar el índice de ocupación de cada una de las ofertas de alojamiento registradas
Resumen	El sistema debe permitir la operación de consulta por parte de los *cliente*s y *operadores* de AlohAndes. Esta se centra en consultar el índice de ocupación de las ofertas de alojamiento registradas en el sistema.
Entradas	
Long: Operador.Id o Long: Cliente.Id	
String: Cliente.Clave o String: Operador.Clave	
Resultados	
Enviar Mensaje: List<string>: valores de indiceOcupacion de todas las tuplas de la relación Alojamiento	
RNF asociados	
Concurrencia: se debe garantizar que varios usuarios de la base de datos puedan acceder simultáneamente a la información del índice de ocupación de los operadores y no haya problemas de concurrencia.	
Persistencia (transaccionalidad): La información del índice de ocupación de los operadores debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).	
Integridad: La información de los operadores debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos.	

3.

1. 2.14 Mostrar los alojamientos disponibles (RFC4)

En la Tabla 14 se muestra el cuarto requerimiento funcional de consulta de AlohAndes. Este se centra en mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios.

Tabla 14. Mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios

Nombre	RFC4. Mostrar los alojamientos disponibles en un rango de fechas, que cumplen con un conjunto de requerimientos de dotación o servicios.
Resumen	La aplicación debe permitir a los *clientes* buscar alojamiento según un conjunto de criterios, como la ubicación, las fechas de estancia, la capacidad, la dotación o servicios incluidos, entre otros. La aplicación debe mostrar una lista de alojamientos disponibles que cumplan con esos criterios.
Entradas	
Date: fechaInicial	
Date: fechaFinal	
Float: precioMinimo	
Float: precioMaximo	
List<Servicio> o List<ServicioPublico>	
Long: Cliente.Id	
String: Cliente.Clave	
Resultados	
Enviar Mensaje: List<Alojamiento>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento y el operador dueño (ej. Alojamiento.ubicacion, Operador.Nombre).	
RNF asociados	
Concurrencia: Pueden hacerse más de una consulta de este tipo al tiempo.	
Seguridad: Los usuarios deben tener permisos adecuados para acceder a la información de alojamientos y disponibilidad, y los datos deben ser protegidos contra acceso no autorizado. Es decir, personas que no sean clientes o empresas relacionadas con la Universidad de los Andes.	
Usabilidad: La interfaz de usuario debe ser clara, sencilla e intuitiva para facilitar la búsqueda de alojamientos por parte de los *clientes*.	

2. 2.15 Mostrar el uso de AlohAndes para cada tipo de usuario (RFC5)

En la Tabla 15 se muestra el quinto requerimiento funcional de consulta de AlohAndes. Este se centra en mostrar el uso de alohandes para cada tipo de usuario de la comunidad. Se le pregunta al usuario por el tipoMiembroComunidad del cual le gustaría ver la actividad y se traen todas las reservas hechas por ese tipoMiembroComunidad. Entiéndase uso como el uso de los alojamientos, es decir las reservas. En este requerimiento, se realiza un inner join entre las tablas reservas y alojamiento para poder seleccionar todas las reservas que se asocian con ese tipoMiembroComunidad.

Tabla 15. Mostrar el uso de alohandes para cada tipo de usuario de la comunidad

Nombre	RFC5. Mostrar el uso de alohandes para cada tipo de usuario de la comunidad
Resumen	La aplicación debe permitir a los clientes y operadores mostrar el uso que se le da a la aplicación dependiendo del tipo de usuario de la comunidad.
Entradas	

String: TipoMiembroComunidad.Nombre
Resultados
Enviar Mensaje: List<Reserva>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento join Reserva para ese tipoMiembroComunidadSeleccionado.
RNF asociados
Concurrencia: se debe garantizar que varios usuarios de la base de datos puedan acceder simultáneamente a la información del uso de la aplicación de los usuarios según su tipo de usuario de la comunidad y no haya problemas de concurrencia.
Persistencia (transaccionalidad): La información del uso de la aplicación de los usuarios según su tipo debe ser persistente para poder ser consultada en cualquier momento y por los usuarios permitidos (clientes o operadores).
Integridad: La información de los operadores debe estar protegida contra posibles modificaciones no autorizadas para mantener la integridad de los datos.

3. 2.16 Mostrar el uso de Alohandes para un usuario dado (RFC6)

4.

En la Tabla 16 se muestra el sexto requerimiento funcional de consulta Alohandes. Este se muestra el uso de Alohandes para un usuario dado. En primer lugar, se pide una verificación de que dicho usuario que quiere hacer esa consulta es un usuario de alohandes. Posteriormente, una vez se ha hecho la verificación, se traen todas las reservas de dicho cliente. Se le muestra el número de noches [días] por reserva que fueron contratados, dinero pagado por reserva y algunas características del alojamiento contratado como si está amoblado o no y la dirección del alojamiento. Es importante recalcar que no se consideran las reservas que fueron canceladas ya que no ocurrieron.

Tabla 16. Mostrar el uso de Alohandes para un usuario dado (número de noches o meses contratados, características del alojamiento utilizado, dinero pagado.

Nombre	RFC6. Mostrar el uso de Alohandes para un usuario dado (número de noches o meses contratados, características del alojamiento utilizado, dinero pagado.
Resumen	Este requerimiento consiste en mostrar el uso de la plataforma por parte de un usuario específico, lo que incluye el número de noches o meses que ha contratado, las características del alojamiento que ha utilizado y el dinero que ha pagado por ello. Este requerimiento tiene como objetivo brindar al usuario información detallada sobre su historial de uso de AlohAndes.
Entradas	
Long: Cliente.Id	
String: Cliente.Clave	
Resultados	
Enviar Mensaje: List<Object>: ciertos atributos de las tuplas seleccionadas de la relación Reserva,Alojamiento [Alojamiento.Id, Alojamiento.Ubicacion, Alojamiento.Amobaldo,Alojamiento.Precio, Reserva.FECHAINICIO, Reserva.FECHAFIN, TotalNoches]	
RNF asociados	
Privacidad (enunciado): La información mostrada debe ser sólo accesible para el cliente que la solicita y no debe ser visible para otros usuarios. De tal manera, que el historial de reservas de un usuario solo puede ser accedido por ese usuario particular.	
Persistencia (transaccionalidad): La información mostrada debe ser obtenida de manera persistente desde la base de datos y no debe perderse en caso de fallos del sistema. Esta debe ser poder consultada en cualquier momento y únicamente por los usuarios cleinte o operador.	
Concurrencia: Se debe permitir que varios usuarios puedan consultar su historial de reservas de alojamientos al mismo tiempo en Alohandes. Por tanto, no debe existir problemas de concurrencia.	

5. 2.17 Analizar la Operación de Alohandes (RFC7)

6.

En la Tabla 17 se muestra el séptimo requerimiento funcional de consulta Alohandes.El RFC8 involucra tres búsquedas a la base de datos que contribuyen al análisis de la operación de Alohandes. La primera involucra encontrar la fecha de mayores ingresos, mientras que las otras dos son búsquedas análogas: encontrar la fecha de mayor ocupación y la fecha de menor ocupación. Para este requerimiento se definió que la búsqueda se haría mensualmente, por lo que la respuesta del requerimiento es una fecha que incluye solamente mes y año (mm-yyyy). Por otro lado, el tipo de alojamiento se pide por consola al usuario.

La lógica de la primera búsqueda empieza buscando las reservas individuales y colectivas del tipo de alojamiento dado, para después añadir el precio del mes de inicio de cada reserva a un *HashMap<String, Long>*, donde se almacenan las fechas encontradas en el formato mencionado anteriormente y los ingresos acumulados en dicha fecha. Posteriormente, se encuentra la fecha de mayor ingreso recorriendo dicho mapa.

La lógica de las otras dos búsquedas consiste en encontrar las reservas individuales y subreservas del tipo de alojamiento solicitado, para luego recorrer estas reservas encontradas y añadirlas a un *HashMap<String, HashSet<Integer>>*, donde se almacenan las fechas en el formato y los id de los alojamientos con reservas en dicha fecha. Finalmente se busca en el mapa la fecha con mayor y menor cantidad de alojamientos.

Tabla 17. Mostrar la operación de Alohandes para fecha de mayores ingresos, mayor ocupación y menor ocupación.

Nombre	RFC7. Analizar la operación de Alohandes
Resumen	Para una unidad de tiempo definido (por ejemplo, semana o mes) y un tipo de alojamiento, considerando todo el tiempo de operación de AloHandes, indicar cuáles fueron las fechas de mayor demanda (mayor cantidad de alojamientos ocupados), las de mayores ingresos (mayor cantidad de dinero recibido) y las de menor ocupación.
Entradas	
String: TipoDeAlojamiento.Nombre	
Resultados	
Enviar Mensaje: String[]: Se retorna la fecha de mayores ingresos, mayor ocupación y menor ocupación.	
RNF asociados	
Persistencia (transaccionalidad): La información mostrada debe ser obtenida de manera persistente desde la base de datos y no debe perderse en caso de fallos del sistema. Esta debe ser poder consultada en cualquier momento y únicamente por los usuarios cleinte o operador.	
Concurrencia: Se debe permitir que varios usuarios puedan consultar su historial de reservas de alojamientos al mismo tiempo en Alohandes. Por tanto, no debe existir problemas de concurrencia.	

7. 2.18 Mostrar el uso de Alohandes para un usuario dado (RFC8)

8.

En la Tabla 18 se muestra el octavo requerimiento funcional de consulta Alohandes. En este se muestra los clientes frecuentes para un alojamiento (más de 3 reservas o más de 15 noches). En primer lugar, se pide el id del alojamiento de interés y posteriormente se hacen dos subconsultas anidadas en una mayor. Primero, se buscan los clientes que tienen más de tres reservas activas; es decir, no están canceladas y, en segundo lugar, los clientes que tengan una reserva cuya fechafinal menos la fecha de inicio sea mayor a 15. Se realiza una unión entre

ambas consultas y se retorna la información de los clientes que cumplen con ello (a excepción de su clave).

Tabla 18. Mostrar los clientes frecuentes para un alojamiento

Nombre	RFC8. Encontrar los clientes frecuentes
Resumen	Para un alojamiento dado, encontrar la información de sus clientes frecuentes. se considera frecuente a un cliente si ha utilizado (o tiene reservado) ese alojamiento por lo menos en tres ocasiones o por lo menos 15 noches, durante todo el periodo de operación de AlohAndes
Entradas	
Long: Alojamiento.Id	
Resultados	
Enviar Mensaje: List<Cliente>: ciertos atributos de las tuplas seleccionadas de la relación Cliente.	
RNF asociados	
Persistencia (transaccionalidad): La información mostrada debe ser obtenida de manera persistente desde la base de datos y no debe perderse en caso de fallos del sistema. Esta debe ser poder consultada en cualquier momento y únicamente por los usuarios cleinte o operador.	
Concurrencia: Se debe permitir que varios usuarios puedan consultar su historial de reservas de alojamientos al mismo tiempo en Alohandes. Por tanto, no debe existir problemas de concurrencia.	

9. 2.18 Encontrar los clientes frecuentes (RFC8)

10.

En la Tabla 18 se muestra el octavo requerimiento funcional de consulta Alohandes. En este se muestra los clientes frecuentes para un alojamiento (más de 3 reservas o más de 15 noches). En primer lugar, se pide el id del alojamiento de interés y posteriormente se hacen dos subconsultas anidadas en una mayor. Primero, se buscan los clientes que tienen más de tres reservas activas; es decir, no están canceladas y, en segundo lugar, los clientes que tengan una reserva cuya fechafinal menos la fecha de inicio sea mayor a 15. Se realiza una unión entre ambas consultas y se retorna la información de los clientes que cumplen con ello (a excepción de su clave).

Tabla 18. Mostrar los clientes frecuentes para un alojamiento

Nombre	RFC8. Encontrar los clientes frecuentes
Resumen	Para un alojamiento dado, encontrar la información de sus clientes frecuentes. se considera frecuente a un cliente si ha utilizado (o tiene reservado) ese alojamiento por lo menos en tres ocasiones o por lo menos 15 noches, durante todo el periodo de operación de AlohAndes
Entradas	
Long: Alojamiento.Id	
Resultados	
Enviar Mensaje: List<Cliente>: ciertos atributos de las tuplas seleccionadas de la relación Cliente.	
RNF asociados	
Persistencia (transaccionalidad): La información mostrada debe ser obtenida de manera persistente desde la base de datos y no debe perderse en caso de fallos del sistema. Esta debe ser poder consultada en cualquier momento y únicamente por los usuarios cleinte o operador.	
Concurrencia: Se debe permitir que varios usuarios puedan consultar su historial de reservas de alojamientos al mismo tiempo en Alohandes. Por tanto, no debe existir problemas de concurrencia.	

11. 2.19 Encontrar las Ofertas de Alojamiento que no tengan mucha demanda (RFC9)

12.

En la Tabla 19 se muestra el noveno requerimiento funcional de consulta Alohandes. En este se muestra las ofertas con poca demanda. Se entiende como poca demanda a aquellos alojamientos que hayan pasado al menos un periodo de 30 días sin reservas. Para ello, se consideró tres subqueries. La primera buscaba entre las reservas de un alojamiento la última reserva y compara con la fecha actual si han pasado más de 30 días se añade ese alojamiento a la lista de poca demanda. En segundo lugar, se considera entre todas las fechas de reserva de un alojamiento si entre reservas hay más de treinta días. Si esto es afirmativo, entonces también se añade a la lista de alojamientos con poca demanda. Finalmente, si un alojamiento no tiene reservas entonces también se añade a la lista de alojamientos con poca demanda.

Tabla 19. Mostrar ofertas con poca demanda

Nombre	RFC9. Mostrar ofertas con poca demanda
Resumen	Encontrar las ofertas de alojamiento que no han recibido clientes en periodos superiores a 1 mes, durante todo el periodo de operación de AlohAndes.
Entradas	
Resultados	
Enviar Mensaje: List<Alojamiento>: ciertos atributos de las tuplas seleccionadas de la relación Alojamiento.	
RNF asociados	
Persistencia (transaccionalidad): La información mostrada debe ser obtenida de manera persistente desde la base de datos y no debe perderse en caso de fallos del sistema. Esta debe ser poder consultada en cualquier momento y únicamente por los usuarios cleinte o operador.	
Concurrencia: Se debe permitir que varios usuarios puedan consultar su historial de reservas de alojamientos al mismo tiempo en Alohandes. Por tanto, no debe existir problemas de concurrencia.	

En general, para esta iteración 3, los requerimientos cumplen con ACID dado que:

Atomicidad: Se cumple ya que todas las operaciones transaccionales se realizan dentro de una misma transacción, de modo que no se pueden ver resultados parciales y estos no quedan en la base de datos.

Consistencia: Se cumple ya que todas las operaciones quedan en la base de datos y los estados antes y después de la operación tienen sentido con respecto a la operación realizada

Isolation: La incorporación de un modelo de aseguramiento de la información SERIALIZABLE asegura el cumplimiento del aislamiento de la transacción

Durabilidad: La utilización de instrucciones de confirmación (TX.COMMIT) después de las operaciones de la transacción asegura la durabilidad de los datos modificados/agregados

3 Documentación de Cambios de Cambios en Diseño

Entre los cambios considerados para las tres clases se incluye la modificación y adición de los atributos de las siguientes relaciones:

1. Operador: Se modificó las restricciones sobre las columnas *ingresoPorAnioActual*, *ingresoPorAnioCorrido* sea mayor o igual a cero. Anteriormente, únicamente era mayor a cero. No obstante, a la hora de calcular los ingresos iniciales de un operador estos

deben iniciar siendo cero; por lo que, se realizó esta modificación para que esto fuera posible.

2. Reserva: Se incluyeron nuevas columnas con los atributos de *cantidadPersonas* y *reservaMadre*. La cantidad de personas facilita la verificación de la restricción de cupo de cada alojamiento. Por otro lado, el atributo de reservaMadre permite cumplir con el nuevo requerimiento de reservas colectivas. Las reservas se van a señalar a sí mismas si son colectivas. Si es unica entonces este atributo es nulo. De lo contrario, si es una subreserva de la reserva colectiva, señala al nodo padre (id). En esta relación también se modificó la restricción el atributo *precio* sea mayor o igual a cero. Anteriormente, únicamente era estrictamente mayor a cero. No obstante, a la hora de definir el precio de una reserva que es cancelada en el proceso de deshabilitación de un alojamiento, se tiene que no hay penalización alguna, por lo que el precio de la reserva tras la cancelación queda en cero, por lo que se realizó esta modificación para que esto fuera posible.
3. Alojamiento: Se agregó una nueva columna a alojamiento indicando si este está activo (1) o inactivo (0). Esto con el fin de cumplir los requerimientos funcionales de habilitar y deshabilitar un alojamiento. Ahora cuando se quiera añadir una reserva, primero se debe hacer una verificación en dicho atributo para confirmar que el alojamiento donde se quiere reservar se encuentra activo actualmente.

En el módulo de persistencia *persistence.xml*, definimos el nivel de aislamiento como serializable con la sentencia *<property name="hibernate.connection.isolation" value="8"/>*, donde el número 8 es el identificador para el nivel de aislamiento deseado. Con este nivel logramos satisfacer el requerimiento no funcional de transaccionalidad (RNF5), el cual establece que debe tenerse en cuenta que en el funcionamiento diario de Alohandes puede haber solicitudes simultáneas, que pueden comprometer los planes de los clientes, por lo que nuestra aplicación debe asegurar la transaccionalidad en el proceso de registro de reservas de alojamientos y sus correspondientes cancelaciones. El nivel serializable elegido garantiza que las transacciones se ejecuten de manera aislada, como si se ejecutaran secuencialmente. Este es el nivel de aislamiento más estricto y previene lecturas sucias, lecturas no repetibles y fantasmas, por lo que se evitan los inconvenientes en caso de concurrencia en la operación de la aplicación. Si bien es cierto que este nivel puede tener un impacto en el rendimiento de la aplicación, tomamos la decisión de darle prioridad a la satisfacción de las transacciones, puesto que debido a la naturaleza de nuestra aplicación, su alcance es bajo. Por lo anterior, no esperamos que este nivel de aislamiento tenga un impacto negativo significativo en términos de desempeño.

4 Resultados Logrados

De forma general, se logró poblar las tablas, los requerimientos de RF1 a RF10 y los requerimientos de consulta RFC1-RFC8. Asimismo se cumplió con:

1. **Privacidad**: El sistema tiene incorporado la autenticación de usuario (ya sea cliente o empresa). Por lo que, se necesita que se haga una autenticación y especificación de la información para poder hacer los requerimientos funcionales. Esto se adicionó en Registrar una Reserva, cancelar una Reserva, adicionar un Alojamiento y Retirar una oferta de Alojamiento.

2. **Persistencia**: Dado que es una base de datos, es persistente el manejo de la información para todas las clases a excepción de Alohandes.

3. **Concurrencia**: Para este requerimiento no funcional se mencionaba que cada una de las solicitudes podían ser hechas de manera múltiple y simultánea. Esto es particularmente importante para el caso de las consultas de información y registros. La carencia de información duplicada garantiza la unicidad de la información. De esta manera, cada consulta o registro se maneja como si fuera secuencial, siguiendo el concepto de transaccionalidad *Isolation.* Además, es posible que dos usuarios distintos hagan uso de la aplicación en máquinas diferentes accediendo a la misma base de datos, y aun así podrán realizar sus operaciones de manera simultánea.

4. **Distribución**: Para la garantía de la información en la base de datos de manera centralizada, se maneja el concepto de Alohandes, que tiene conexión directa o indirecta con todas las entidades del modelo, y es a partir de donde se inicia el análisis para cada uno de los requerimientos funcionales una vez se ejecuta la autenticación de Usuario (empresa/cliente). Además, está centralizada haciendo uso del SMBD de Oracle SQL Developer.

5. **Transaccionalidad:** La aplicación cumple con transaccionalidad ya que considera que puede haber solicitudes simultáneas. Esto se consideró la capa de persistencia, en especial, al considerar tx.begin, tx.commit y el nivel de aislamiento como se discutió en la sección anterior.

5 Resultados No Logrados

En relación con esta iteración y la anterior, se pudo haber mejorado un poco más la separación de los distintos actores y sus requerimientos funcionales. Esto se debe a que aunque se pide el usuario y contraseña antes de ejecutar la mayoría de requerimientos, aún aparece en la interfaz todos los requerimientos a la vista de otros usuarios. Por lo que los pueden visualizar aunque no ejecutar. Por ello, hubiera sido ideal mostrar únicamente los requerimientos funcionales de acuerdo al usuario que ingresa y los permisos de acceso que tiene. Esto es algo que no se ha modificado desde la iteración pasada.

6 Supuestos Adicionales

Con respecto a anteriores iteraciones algunos de los supuestos adicionales que se tuvieron en cuenta fueron:

1. Si una reserva ha sido reubicada luego de que un alojamiento fue deshabilitado esta no vuelve a ser reubicada una vez el alojamiento se vuelva a rehabilitar. Estas se quedan en el alojamiento en el que se ubican hasta la fechafinal.
2. Una reserva solo puede ser reubicada en el mismo tipoAlojamiento. Es decir, si una reserva fue hecha, por ejemplo, para un hotel solo podrá ser reubicada en otro hotel.
3. El alojamiento que queda deshabilitado no recibe penalización ni descuentos en ingresos por reubicar reservas.
4. Nunca se eliman reservas de la base de datos. Las reservas se dan por canceladas cuando se les da una fecha de cancelación.

7 Balance de Pruebas

4. A continuación se muestra el plan de pruebas para los requerimientos funcionales: RF7, RF8, RF9, RF10. Estos incluyen caso de éxito y casos de fallo.

1.

2. 7.1 Registrar una Oferta Colectiva (RF7)

Los casos de éxito para el requerimiento funcional son:

Estado Inicial de la BD

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)

Figura 2. Datos involucrados RF7 [Elaborado en OracleSQL]

Datos involucrados en la operación transaccional solicitada

```
-- reserva colectiva
INSERT INTO RESERVA (ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1000, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), null, 1, null, 0, 0,1000);

-- subreservas de la colectiva
INSERT INTO RESERVA
(ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1001, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), 7, 1, null, 100000, 1,1000);
INSERT INTO RESERVA
(ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1002, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), 10, 1, null, 300000, 2,1000);
```

Figura 3. Datos involucrados RF7 [Elaborado en OracleSQL]

Estado Final de la BD

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	1000	(null)	01/06/23	01/08/23	(null)	1	(null)	0	0	1000
12	1001	(null)	01/06/23	01/08/23	7	1	(null)	100000	1	1000
13	1002	(null)	01/06/23	01/08/23	10	1	(null)	300000	2	1000

Figura 4. Estado final de la BD [Elaborado en OracleSQLt]

Los casos de fallo para el requerimiento funcional se consideran a continuación, note que el estado inicial de la BD de datos es el mismo que en la figura 2.

Datos involucrados en la operación transaccional solicitada

```
-- reserva colectiva
INSERT INTO RESERVA (ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1000, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), null, 1, null, 0, 0,1000);

-- subreservas de la colectiva
INSERT INTO RESERVA
(ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1001, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), 7, 1, null, 100000, 6,1000);
INSERT INTO RESERVA
(ID, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO,PRECIO,CANTIDADPERSONAS,RESERVAMADRE)
values (1002, to_date('2023-06-01','yyyy-mm-dd'), to_date('2023-08-01','yyyy-mm-dd'), 10, 1, null, 300000, 7,1000);
```

Figura 5. Datos involucrados en el caso de falla para RF7 [Elaborado en OracleSQL]

Estado Final de la BD

En el caso de falla se considera la opción cuando el usuario solicita una mayorCantidad de personas a la disponible. En este caso, falla ya que la máxima capacidad por alojamiento es 4.

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)

Figura 6. Estado Final para la BD caso de falla [Elaborado en OracleSQL]

3. 7.2 Cancelar Reserva Colectiva (RF8)

Note que para correr el caso de éxito debe haber corrido el RF7 del literal anterior. Los casos de éxito para el requerimiento funcional son:

Estado Inicial de la BD

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	1000	(null)	01/06/23	01/08/23	(null)	1	(null)	0	0	1000
12	1001	(null)	01/06/23	01/08/23	7	1	(null)	100000	1	1000
13	1002	(null)	01/06/23	01/08/23	10	1	(null)	300000	2	1000

Figura 7. Estado de la BD inicial para RF8 [Elaborado en OracleSQL]

Datos involucrados en la operación transaccional solicitada

```
UPDATE RESERVA SET FECHACANCELACION = TO_DATE('2023-05-09', 'YYYY-MM-DD') WHERE RESERVAMADRE = 1000;
```

Figura 8. Datos para RF8 [Elaborado en OracleSQL]

Estado Final de la BD

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	1000	09/05/23	01/06/23	01/08/23	(null)	1	(null)	0	0	1000
12	1001	09/05/23	01/06/23	01/08/23	7	1	(null)	100000	1	1000
13	1002	09/05/23	01/06/23	01/08/23	10	1	(null)	300000	2	1000

Figura 9. Estado Final de la BD inicial para RF8 [Elaborado en OracleSQL]

Se considera que los únicos casos de falla para este requerimiento sería si el usuario ingresa un idReservaMadre que no se encuentre en la baseDeDatos.

Estado Inicial de la BD

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	1000	(null)	01/06/23	01/08/23	(null)	1	(null)	0	0	1000
12	1001	(null)	01/06/23	01/08/23	7	1	(null)	100000	1	1000
13	1002	(null)	01/06/23	01/08/23	10	1	(null)	300000	2	1000

Figura 10. Estado de la BD inicial para RF8 caso falla [Elaborado en OracleSQL]

Datos involucrados en la operación transaccional solicitada

```
UPDATE RESERVA SET FECHACANCELACION = TO_DATE('2023-05-09', 'YYYY-MM-DD') WHERE RESERVAMADRE = 2000;
```

Figura 11. Datos para RF8 falla [Elaborado en OracleSQL]

Estado Final de la BD

Como no se actualizó ninguna fila ya que 2000 no existe entonces queda igual que al inicio.

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDADPERSONAS	RESERVAMADRE
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	1000	(null)	01/06/23	01/08/23	(null)	1	(null)	0	0	1000
12	1001	(null)	01/06/23	01/08/23	7	1	(null)	100000	1	1000
13	1002	(null)	01/06/23	01/08/23	10	1	(null)	300000	2	1000

Figura 12. Estado Final para RF8 falla [Elaborado en OracleSQL]

7.3 Deshabilitar Oferta de Alojamiento (RF9)

Los casos de éxito para el requerimiento funcional son:

Estado Inicial de la BD

Alojamiento

	ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR	ACTIVO
1	1	18	71074 Lyons Parkway	1329228,6	1	4	1	50	4	10	9	1	1
2	2	5	18 Dunning Drive	1646894	0	4	0	50	4	7	6	4	1
3	3	17	62 Carioca Junction	854126,8	0	4	0	50	5	10	9	3	1
4	4	17	8 Buell Trail	5386667,9	1	4	1	50	1	2	4	6	1
5	5	16	0056 Hanover Crossing	3994171,9	0	4	1	50	5	7	5	4	1
6	6	8	224 Harbort Trail	4027126,9	0	4	0	50	4	10	10	1	1
7	7	3	70867 Chinook Parkway	9679071	0	4	0	50	3	9	1	7	1
8	8	12	5 Johnson Way	8223543	0	4	0	50	1	3	3	9	1
9	9	11	2 Quincy Street	8160659,5	0	4	0	50	1	10	7	5	1
10	10	3	0138 Linden Drive	3826016,6	0	4	0	50	3	3	1	2	1
11	11	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	2	1
12	12	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	3	1

Reserva

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDA...	RESERVA...
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	12	(null)	01/04/23	01/06/23	3	2	(null)	3052932,6	1	(null)
12	11	(null)	01/06/24	01/08/24	3	2	(null)	3365758,4	1	(null)

Datos involucrados en la operación transaccional solicitada

```
UPDATE ALOJAMIENTO SET ACTIVO = 0 WHERE ID = 3;

-- en caso de haber colectivas y subreservas
UPDATE RESERVA SET FECHACANCELACION = '2023-05-09'
WHERE RESERVAMADRE = 3 AND ID IN (SELECT RESERVA.ID FROM Reserva where reserva.alojamiento = 3 and fechafinal > TO_DATE('2023-05-09', 'YYYY-MM-DD'));

UPDATE RESERVA SET RESERVAMADRE = NULL WHERE RESERVAMADRE != ID
               AND ID IN (SELECT RESERVA.ID FROM Reserva where reserva.alojamiento = 3
                   and fechafinal > TO_DATE('2023-05-09', 'YYYY-MM-DD'));

-- luego se vuelve a usar el RF4 y RF5

insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO, PRECIO, CANTIDADPERSONAS, RESERVAMADRE)
values (13, null, to_date('2024-06-01', 'yyyy-mm-dd'), to_date('2024-08-01', 'yyyy-mm-dd'), 5, 2, null, 1365758.4, 1, null);
insert into RESERVA (ID, FECHACANCELACION, FECHAINICIO, FECHAFINAL, ALOJAMIENTO, CLIENTE, CONTRATO, PRECIO, CANTIDADPERSONAS, RESERVAMADRE)
values (14, null, to_date('2023-04-01', 'yyyy-mm-dd'), to_date('2023-06-01', 'yyyy-mm-dd'), 5, 2, null, 1052932.6, 1, null);

UPDATE RESERVA SET FECHACANCELACION = TO_DATE('2023-05-09', 'YYYY-MM-DD') WHERE ID = 11;
UPDATE RESERVA SET FECHACANCELACION = TO_DATE('2023-05-09', 'YYYY-MM-DD') WHERE ID = 12;
```

Estado Final de la BD

Alojamiento

	ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR	ACTIVO
1	1	18	71074 Lyons Parkway	1329228,6	1	4	1	50	4	10	9	1	1
2	2	5	18 Dunning Drive	1646894	0	4	0	50	4	7	6	4	1
3	3	17	62 Carioca Junction	854126,8	0	4	0	50	5	10	9	3	0
4	4	17	8 Buell Trail	5386667,9	1	4	1	50	1	2	4	6	1
5	5	16	0056 Hanover Crossing	3994171,9	0	4	1	50	5	7	5	4	1
6	6	8	224 Harbort Trail	4027126,9	0	4	0	50	4	10	10	1	1
7	7	3	70867 Chinook Parkway	9679071	0	4	0	50	3	9	1	7	1
8	8	12	5 Johnson Way	8223543	0	4	0	50	1	3	3	9	1
9	9	11	2 Quincy Street	8160659,5	0	4	0	50	1	10	7	5	1
10	10	3	0138 Linden Drive	3826016,6	0	4	0	50	3	3	1	2	1
11	11	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	2	1
12	12	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	3	1

Reserva

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDA...	RESERVA...
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	11	09/05/23	01/06/24	01/08/24	3	2	(null)	3365758,4	1	(null)
12	12	09/05/23	01/04/23	01/06/23	3	2	(null)	3052932,6	1	(null)
13	13	(null)	01/06/24	01/08/24	5	2	(null)	1365758,4	1	(null)
14	14	(null)	01/04/23	01/06/23	5	2	(null)	1052932,6	1	(null)

Se considera que los únicos casos de falla para este requerimiento sería si el usuario ingresa un idReserva que no se encuentre en la baseDeDatos.

4. 7.4 Rehabilitar Oferta de Alojamiento (RF10)

Los casos de éxito para el requerimiento funcional son:

Estado Inicial de la BD

Alojamiento

	ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR	ACTIVO
1	1	18	71074 Lyons Parkway	1329228,6	1	4	1	50	4	10	9	1	1
2	2	5	18 Dunning Drive	1646894	0	4	0	50	4	7	6	4	1
3	3	17	62 Carioca Junction	854126,8	0	4	0	50	5	10	9	3	0
4	4	17	8 Buell Trail	5386667,9	1	4	1	50	1	2	4	6	1
5	5	16	0056 Hanover Crossing	3994171,9	0	4	1	50	5	7	5	4	1
6	6	8	224 Harbort Trail	4027126,9	0	4	0	50	4	10	10	1	1
7	7	3	70867 Chinook Parkway	9679071	0	4	0	50	3	9	1	7	1
8	8	12	5 Johnson Way	8223543	0	4	0	50	1	3	3	9	1
9	9	11	2 Quincy Street	8160659,5	0	4	0	50	1	10	7	5	1
10	10	3	0138 Linden Drive	3826016,6	0	4	0	50	3	3	1	2	1
11	11	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	2	1
12	12	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	3	1

Reserva

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDA...	RESERVA...
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	11	09/05/23	01/06/24	01/08/24	3	2	(null)	3365758,4	1	(null)
12	12	09/05/23	01/04/23	01/06/23	3	2	(null)	3052932,6	1	(null)
13	13	(null)	01/06/24	01/08/24	5	2	(null)	1365758,4	1	(null)
14	14	(null)	01/04/23	01/06/23	5	2	(null)	1052932,6	1	(null)

Datos involucrados en la operación transaccional solicitada

```
UPDATE ALOJAMIENTO SET ACTIVO = 1 WHERE ID = 3;
```

Estado Final de la BD

Alojamiento

	ID	NUMHABITACIONES	UBICACION	PRECIO	AMOBLADO	CAPACIDAD	COMPARTIDO	INDICEOCUPACION	TIPO	HORARIO	SEGURO	OPERADOR	ACTIVO
1	1	18	71074 Lyons Parkway	1329228,6	1	4	1	50	4	10	9	1	1
2	2	5	18 Dunning Drive	1646894	0	4	0	50	4	7	6	4	1
3	3	17	62 Carioca Junction	854126,8	0	4	0	50	5	10	9	3	1
4	4	17	8 Buell Trail	5386667,9	1	4	1	50	1	2	4	6	1
5	5	16	0056 Hanover Crossing	3994171,9	0	4	1	50	5	7	5	4	1
6	6	8	224 Harbort Trail	4027126,9	0	4	0	50	4	10	10	1	1
7	7	3	70867 Chinook Parkway	9679071	0	4	0	50	3	9	1	7	1
8	8	12	5 Johnson Way	8223543	0	4	0	50	1	3	3	9	1
9	9	11	2 Quincy Street	8160659,5	0	4	0	50	1	10	7	5	1
10	10	3	0138 Linden Drive	3826016,6	0	4	0	50	3	3	1	2	1
11	11	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	2	1
12	12	3	0138 Linden Drive	3826016,6	0	4	0	50	4	10	1	3	1

Reserva

	ID	FECHACANCELACION	FECHAINICIO	FECHAFINAL	ALOJAMIENTO	CLIENTE	CONTRATO	PRECIO	CANTIDA...	RESERVA...
1	1	03/06/22	17/03/22	07/06/22	2	4	1	3365758,4	1	(null)
2	2	(null)	22/04/21	19/02/23	3	5	3	3052932,6	1	(null)
3	3	23/09/22	26/07/21	23/12/22	1	8	10	5524267	1	(null)
4	4	12/06/22	19/03/21	12/08/22	9	8	(null)	5627346,4	1	(null)
5	5	(null)	20/03/21	26/09/22	10	6	3	7067208	1	(null)
6	6	(null)	27/02/22	07/09/22	8	7	8	6100698,6	1	(null)
7	7	20/11/22	31/08/21	15/12/22	5	3	9	4000931,8	1	(null)
8	8	(null)	31/07/21	08/07/22	8	4	(null)	4975193	1	(null)
9	9	(null)	16/02/22	12/08/22	1	6	7	152258,3	1	(null)
10	10	(null)	29/11/21	31/08/22	6	3	1	4458495	1	(null)
11	11	09/05/23	01/06/24	01/08/24	3	2	(null)	3365758,4	1	(null)
12	12	09/05/23	01/04/23	01/06/23	3	2	(null)	3052932,6	1	(null)
13	13	(null)	01/06/24	01/08/24	5	2	(null)	1365758,4	1	(null)
14	14	(null)	01/04/23	01/06/23	5	2	(null)	1052932,6	1	(null)

Se considera que los únicos casos de falla para este requerimiento sería si el usuario ingresa un idReserva que no se encuentre en la baseDeDatos.

8 Bibliografía

1. *www.xml.org.* [En línea] [Citado el: 28 de Abril de 2010.] http://www.xml.org.

2. The Institution of Engineering and Technology. *A Guide to Technical Report Writing.* [En línea] www.theiet.org/students/resources/technicalreport.cfm.

3. *Universidad de los Andes.* [En línea] [Citado el: 28 de Abril de 2010.] http://uniandes.edu.co.

4. IEEE. *Manual de estilo de documentos técnicos.* [En línea] [Citado el: 28 de Abril de 2010.] http://standards.ieee.org/guides/style/2009_Style_Manual.pdf.

5. LNCS Springer Verlag. *Lecture Notes in Computer Science.* [En línea] [Citado el: 28 de Abril de 2010.] http://www.springer.com/computer/lncs?SGWID=0-164-12-73062-0.

6. Universidad de los Andes. *CARTILLA DE CITAS: Pautas para citar textos y hacer listas de referencias.* [En línea] Universidad de los Andes. [Citado el: 28 de Abril de 2010.] http://decanaturadeestudiantes.uniandes.edu.co/Documentos/Cartilla_de_citas.pdf.