Caso de estudio: ALOHANDES

José D. Flórez Ruiz, Carlos M. Muñoz Almeida
Proyecto 1 – Iteración 1
Universidad de los Andes, Bogotá, Colombia
{jd.florezr1, c.munoza}@uniandes.edu.co
Fecha de presentación: Febrero 26 de 2023

Tabla de contenido

1 Análisis y modelo conceptual

1.1 Requerimientos funcionales

Nombre	RF1. **REGISTRAR LOS OPERADORES DE ALOJAMIENTO PARA ALOHANDES**
Resumen	Los operadores necesitan ser registrados en la base de datos de AlohAndes.
Entradas	
Los operadores se registrarán en la base de datos proveyendo su identificación, nombre y tipo de vinculación con la universidad que están definidas por los siguientes tipos: • Estudiante • Profesor • Padre de Estudiante • Trabajador • Externo (Personas naturales)	
Resultados	
El resultado de esta transacción será un retorno por parte del sistema confirmando en el caso de que la transacción sea satisfactoria una respuesta de éxito, y en el caso contrario en el que la transacción no se realice de manera satisfactoria, se le debe enviar una respuesta de error/problemas.	
RNF asociados	

Este requerimiento exige un comportamiento transaccional puesto que necesitamos mantener una alta calidad en el manejo de los datos. Esto lo podemos lograr mediante un manejo transaccional que cumpla con ACID.
Este requerimiento necesita hacer uso de persistencia ya que como se puede intuir es de gran importancia mantener de manera permanente los datos después de que se realice la transacción.

Nombre	RF2. **REGISTRAR PROPUESTAS DE ALOJAMIENTOS PARA ALOHANDES.**
Resumen	Los posibles operadores pueden brindar propuestas a AlohAndes para que, en caso de ser seleccionados, se conviertan en proveedores de AlohAndes.
Entradas	
Dependiendo del tipo de operador que esté tratando de registrar una propuesta de alojamiento para AlohAndes tendrán distintas opciones habilitadas como es el caso cuándo se es miembro de la comunidad institucional el individuo podrá registrar una habitación dentro de su hogar, un apartamento en alquiler o una vivienda temporal. Sin embargo, hay casos en los que no tienen todas estas habilitadas, como es el caso de las personas naturales externas a la universidad las que solo tendrán la posibilidad de registrar una habitación dentro de su hogar o una vivienda temporal.	
El caso es distinto cuando una empresa trata de registrar una habitación. El tipo de habitación variara dependiendo del tipo de empresa que sean. Por ejemplo: hoteles solo podrán registrar habitación de hotel, hostales solo podrán registrar habitación de hostal y vivienda universitaria solo podrá registrar una habitación de vivienda universitaria. Las entidades cambian debido a que cada una tiene reglas de funcionamiento diferentes las cuales serían imposible de plasmar todas en conjunto.	
Las entradas necesarias dependiendo cada caso son: • **Habitación huésped / Habitación dentro de un hogar:** 1. Comidas: Integer 2. Acceso a Cocina: Boolean 3. Tipo de baño: String 4. Tipo de habitacion: String 5. Cuota de luz: Integer 6. Cuota de telefono: Integer 7. Cuota de agua: Integer 8. Cuota de tv cable: Integer 9. Descuento de cada mes extra: Integer • **Apartamento Alquiler:** 1. Amoblado: Boolean 2. Servicio público incluido: Boolean 3. Tv incluido: Boolean 4. Internet incluido: Boolean 5. Administración incluida: Boolean • **Vivienda Temporal:**	

1. Número de habitaciones: Integer
2. Mensaje: String
3. Precio seguro de arrendamiento: Integer
4. Características de seguro: String
5. Días que ha sido alquilado en el año: Integer

- Habitación hotel:
  1. Tipo de habitación: String
  2. Tamaño habitación: String
  3. Tiene Bañera: Boolean
  4. Tiene Yacuzzi: Boolean
  5. Tiene sala: Boolean
  6. Tiene Cocineta: Boolean
- Habitación vivienda universitaria:
  1. Tipo de habitación: String
  2. Capacidad de habitación: Integer
  3. Descripción de menaje: String
- Habitación hostal:
  1. Aforo: Integer

Además, algunos otros datos que todos tienen que dar en general:

1. Ubicación: String
2. Costo base: Integer
3. Duración mínima: Integer
4. Costo Total: Integer

Resultados

Tras él envió de los datos la base de datos notificara si la transacción fue realizada exitosamente. Si efectivamente salió bien, se envía mensaje de éxito en el retorno, y en el caso contrario se envía un mensaje de error.

RNF asociados

Este requerimiento necesita transaccionalidad ya que como se evidencia en la descripción es necesario manipular los datos por medio de un conjunto de queries en forma de una transacción con el fin de poder registrar todos los datos necesarios dentro de la tabla necesarias.

Por otro lado, este requerimiento hace uso de persistencia esto con el fin de que tras haberse realizado la transacción los cambios realizados por este permanezcan en la base de datos.

Nombre	RF3. **REGISTRAR LAS PERSONAS HABILITADAS PARA UTILIZAR LOS SERVICIOS**

Resumen	Los clientes necesitan que su información sea registrada en la base de datos de AlohAndes.
Entradas	
Las personas habilitadas para utilizar los servicios proveerán los siguientes datos: • Nombre • Tipo de vínculo con la universidad • Identificación • Teléfono • Correo Electrónico	
Resultados	
Cuando se intente realizar la transacción de registro de un usuario habrá dos respuestas. Una de estas es la de éxito la cual se mostrará en dado caso todas las quieres de la transacción se hayan podido realizar exitosamente, en el caso contrario se retornará un error.	
RNF asociados	
Se necesita transaccionalidad ya que por medio de esta se manipula la base de datos de la manera que deseamos por medio de las queries pertenecientes a la transacción. Haciendo imperativo para poder manipular la base de datos de manera segura, manteniendo la integridad de los datos.	
Se necesita persistencia puesto que los datos del cliente deben conservarse tras realizarse la transaccion.	

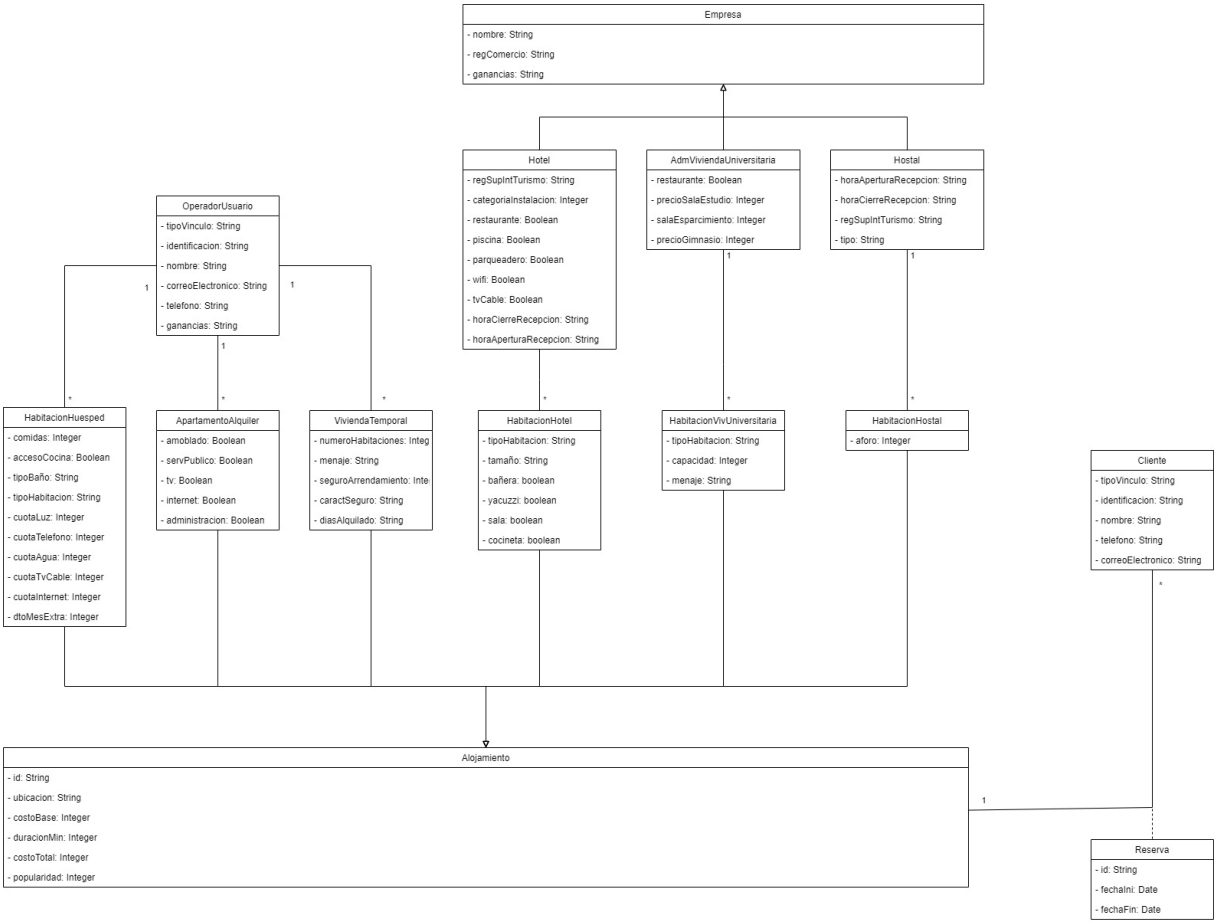
Nombre	RF4. **REGISTRAR UNA RESERVA**
Resumen	Los clientes deben tener la capacidad de realizar una reserva en el sistema. El sistema debe considerar las preferencias del cliente. Por ejemplo, vivienda compartida, internet incluido, cocineta, etc.
Entradas	
Se reciben datos del cliente, tales como: su tipo de vinculo con la universidad, su identificación y su nombre. El tipo de vinculo puede ser: estudiante, egresado, empleado, profesor, padre de estudiante, profesor invitado, persona registrada en evento de Uniandes.	
Resultados	
Registro de una nueva reserva.	
Notificación al cliente y al operador de que se realizó una nueva reserva exitosa.	
RNF asociados	
Este requerimiento necesita transaccionalidad puesto que se debe garantizar que la operación de realizar una reserva se haga de manera completa y segura. También, es necesario contar una alta integridad que los datos que el cliente provea.	
Este requerimiento necesita persistencia ya que la reserva debe ser almacenada.	

Nombre	RF5. **CANCELAR UNA RESERVA**
Resumen	El cliente debe ser capaz de cancelar una reserva dentro del sistema.

Entradas
Se requieren datos de la reserva, tales como su id, fecha de inicio y fecha de fin.
Se requieren datos del cliente, tales como su identificación y nombre.
Resultados
Actualización en la base datos.
Notificación al operador y al cliente de que la reserva fue cancelada exitosamente.
RNF asociados
Se requiere transaccionalidad puesto que es necesario garantizar la completitud de la cancelación de reserva. Es decir, tanto el cliente como el operador deben ser conscientes de que la reserva fue cancelada. Esto por medio de la actualización de la base de datos.
Este requerimiento debe tener persistencia ya que se debe guardar en los registros que la reserva fue cancelada.

Nombre	RF6. **RETIRAR UNA OFERTA DE ALOJAMIENTO**
Resumen	El operador puede ser capaz retirar una oferta de alojamiento dentro del sistema. Es importante resaltar que la oferta de alojamiento se puede retirar solo después de la finalización de la última reserva vigente.
Entradas	
Se requiere conocer que operador va a retirar que oferta. De modo que, se recibe el id del alojamiento que el operador va retirar.	
En caso de que el operador sea una persona natural o un miembro de unidades, se recibe su identificación. En caso de que el operador sea una empresa se recibe su numero de registro en la cámara de comercio.	
Resultados	
Actualización en la base de datos.	
Notificación al operador de que su oferta fue retirada con éxito.	
RNF asociados	
Este requerimiento tiene transaccionalidad ya que se necesita modificar la base datos de una manera íntegra y completa de manera que se retire por completo la oferta del operador. Y así, ya no sea visible para los clientes.	
Este requerimiento hay persistencia ya que se debe guardar el registro de que la oferta fue retirada.	

1.2 Modelo conceptual



Análisis de persistencia:

Como podemos observar, todas las clases del modelo conceptual cumplen un papel fundamental en el en el negocio. Por tal razón, consideramos que todas las clases corresponden a entidades que deben ser presidentes a excepción de la clase empresa.

En primer lugar, para cada una de las clases que mencionamos como persistentes se debe almacenar la información de sus atributos de manera íntegra, completa y segura. Así, la información almacenada de cada entidad persistirá para que pueda ser usada de manera coherente, consistente y segura por un operador o cliente.

Por otra parte, decimos que la clase empresa no necesita de persistencia dada la lógica del negocio. Es decir, las clases que heredan de empresa recogen sus atributos y tenemos un modelo relacional para cada clase que hereda. Por tanto, no tiene sentido crear un modelo persistente para empresa puesto que se replicaría información y se podría dar lugar a inconsistencias.

2 Diseño de la base de datos

2.1 Modelo relacional

OperadorUsuario

identificacion	nombre	tipoVinculo	telefono	correoElectronico
PK, UA	NN	NN, CK5	NN, CK7	NN, CK8

HabitacionHuesped

idAlojamiento	comidas	accesoCocina	tipoBaño	tipoHabitacion	cuotaLuz	cuotaTelefono	cuotaAgua	cuotaTvCable	cuotaInternet	dtoMesExtra	propietario
PK, FKAlojamiento.id	NN, CK >= 0	NN, CK1	NN, CK2	NN, CK3	NN, CK > 0	NN, CK > 0	NN, CK > 0	NN, CK > 0	NN, CK > 0	NN, 100 >= CK > 0	NN, FKOperadorUsuario.identificacion

ApartamentoAlquiler

idAlojamiento	amoblado	servPublico	tv	internet	administracion	propietario
PK, FKAlojamiento.id	NN, CK1	NN, CK1	NN, CK1	NN, CK1	NN, CK1	NN, FKOperadorUsuario.identificacion

Restricciones : El apartamento solo podra ser alquilado por un propietario que tenga algun tipo de vinculo con la comunidad uniandina.

ViviendaTemporal

idAlojamiento	numeroHabitaciones	menaje	seguroArrendamiento	caracSeguro	diasAlquilado	propietario
PK, FKAlojamiento.id	NN, CK > 0	NN	NN, CK > 0	NN	NN, CK <= 30	NN, FKOperadorUsuario.identificacion

Cliente

identificacion	nombre	tipoVinculo	telefono	correoElectronico
PK, UA	NN	NN, CK5	NN, CK7	NN, CK8

Reserva

id	fechaIni	fechaFin	identificacionCliente	idAlojamiento
PK, SA	NN	NN	NN, FKCliente.Identificacion	NN, FKAlojamiento.id

Alojamiento

id	ubicacion	costoBase	duracionMin	costoTotal	popularidad
PK, SA	NN, CK4	NN, CK > 0	NN, CK >= 30	NN, CK > 0	NN

Hotel

regComercio	nombre	regSupInrTurismo	categoriaInstalación	restaurante	piscina	parqueadero	wifi	tvCable	horaCierreRecepción	horaAperturaRecepción	ganancias
PK, UA	NN	NN	NN, CK(>0)	NN, CK1	NN, CK1	NN, CK1	NN, CK1	NN, CK1	NN	NN	NN

HabitaciónHotel

id	tipoHabitación	tamaño	bañera	yacuzzi	sala	cocineta	hotelEncargado
PK, FKAlojamiento.id	NN, CK6	NN	NN, CK1	NN,CK1	NN,CK1	NN	NN, FKHotel.regComercio

AdmViviendaUniversitaria

regComercio	nombre	restaurante	precioSalaEstudio	precioSalaEsparcimiento	precioGimnasio	ganancias
PK, UA	NN	NN, CK1	NN, CK(>=0)	NN, CK(>=0)	NN, CK(>=0)	NN

HabitaciónVivUniversitaria

id	tipoHabitación	capacidad	menaje	admViviendaUniversitaria
PK, FKAlojamiento.id	NN	CK(>0)	NN	**NN, FKAdmViviendaUnive**

Hostal

regComercio	nombre	regSupIntTurismo	horaAperturaRecepción	horaCierreRecepción	tipo	ganancias
PK, UA	NN	NN	NN	NN	NN	NN

HabitaciónHostal

id	aforo	hostal
PK, FKAlojamiento.id	CK(>=0)	**NN, FKHostal.regComerci**

CKs

CK1	('true', 'false')
CK2	('privado', 'compartido')
CK3	('individual', 'compatida')
CK4	calle {string} #{string} - {string}
CK5	('profesor', 'empleado', 'egresado', 'estudiante', 'padre de estudiante', 'externo')
CK6	('estándar', 'semisuite', 'suites')
CK7	+{prefijo} {7 - 10 enteros}
CK8	{username} @ {dominio}

2.2 Nivel de normalización

Para lograr determinar la forma normal en la que se encuentra nuestro modelo, realizaremos un recorrido por las diferentes normas y evaluaremos si nuestro modelo la cumple.

1FN: Según la bibliografía del curso, más específicamente, según Jan L. Harrington en "Relational Database Design and Implementation" especifica que un modelo que se encuentra en 1FN cumple con que no hay grupos repetidos, o dicho de otra forma sus dominios son

atómicos. Así pues, en nuestro modelo, cada uno de los atributos de cada entidad cumple con tal requisito.

2FN: Contando con la bibliografía ya mencionada, para cumplir con 2FN, se debe estar en 1FN, que ya lo verificamos anteriormente, y todos los atributos que no son llaves (o sea atributos no primos) deben ser funcionalmente dependientes de todos los atributos que conformen la llave primaria y no solo de una parte de ella.

Así pues, validando en nuestro modelo, encontramos que cada relación cuenta con una llave primaria compuesta por solamente un atributo. Por tal motivo, podemos afirmar que el resto de los atributos, que son no llaves, dependen funcionalmente la llave primaria completa.

3FN: En lo que respecta a esta forma, se debe cumplir con la 2FN y con que ningún atributo es determinado por otro atributo no primo. Así pues, revisando cada relación de nuestro modelo, solamente las llaves primarias son capaces de determinar a los demás atributos, ningún atributo que no es llave primaria es capaz de determinar a otro.

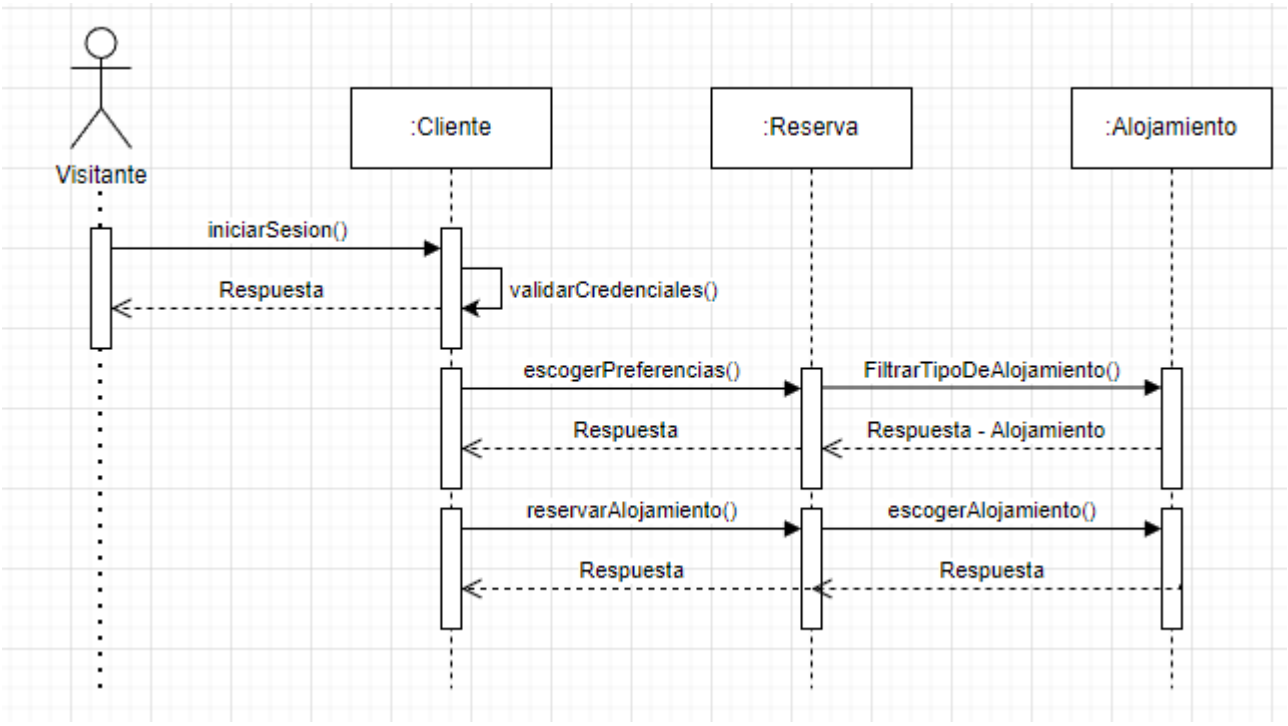
Boyce-Codd FN: Teniendo en cuenta la bibliografía sugerida llegamos a la definición de la forma normal de Boyce-Codd la cual dice: “para cada función de dependencia (x → y), x debe ser una super llave”, además tiene que ya ser de la forma normal 3. Teniendo en cuenta la definición anteriormente dada, podemos llegar a la conclusión de que nuestro modelo relacional cumple con ella en todas sus tablas y por ende este llega a la forma normal de Boyce-Codd.

De esta forma, y teniendo en cuenta la bibliografía ya mencionada, cuando un modelo se encuentra en FNBC es porque ha superado la gran mayoría de anomalías, y cuenta con buenas bases de diseño. Por tanto, ya no continuaremos con la revisión de las siguientes FN, y concluiremos en que nuestro modelo se encuentra en FNBC.

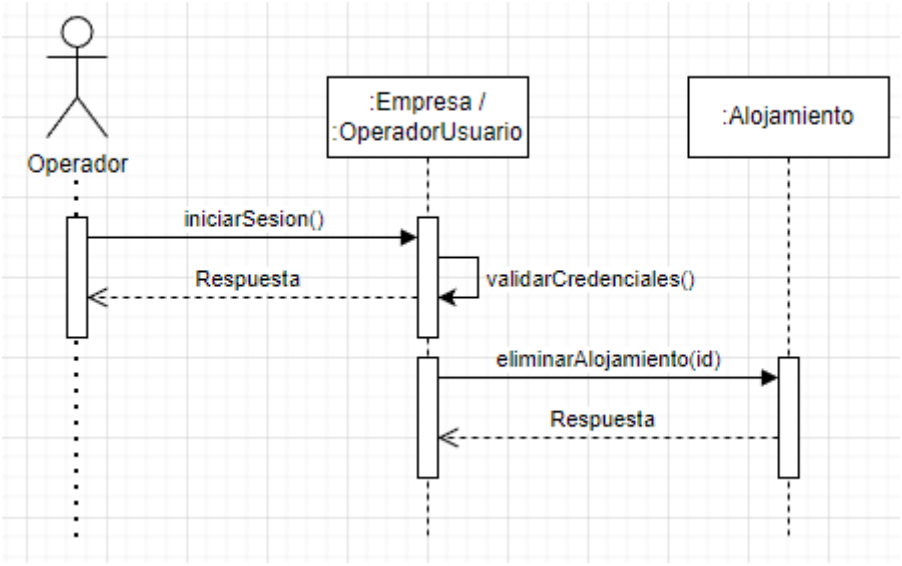
3 Control de calidad del modelo

3.1 Diagramas de secuencia

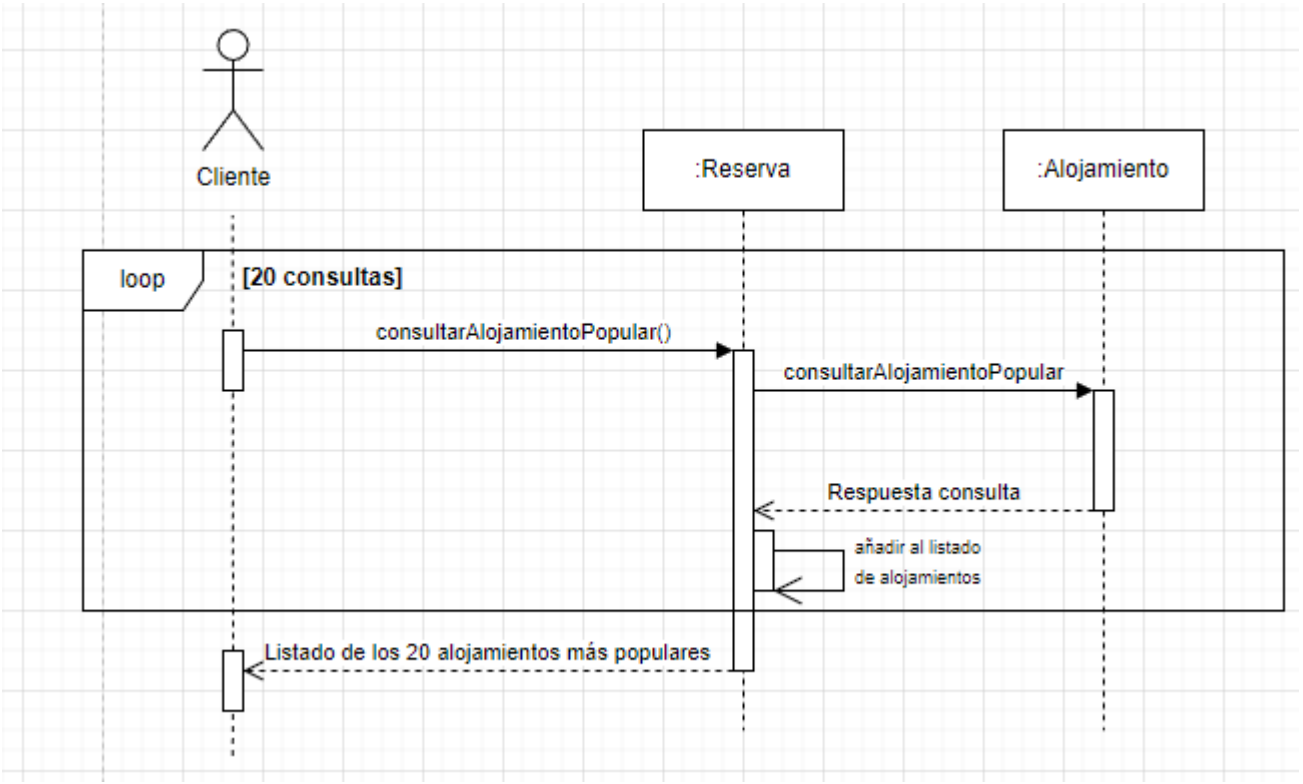
RF4


Visitante
:Cliente
:Reserva
:Alojamiento
iniciarSesion()
Respuesta
validarCredenciales()
escogerPreferencias()
FiltrarTipoDeAlojamiento()
Respuesta
Respuesta - Alojamiento
reservarAlojamiento()
escogerAlojamiento()
Respuesta
Respuesta

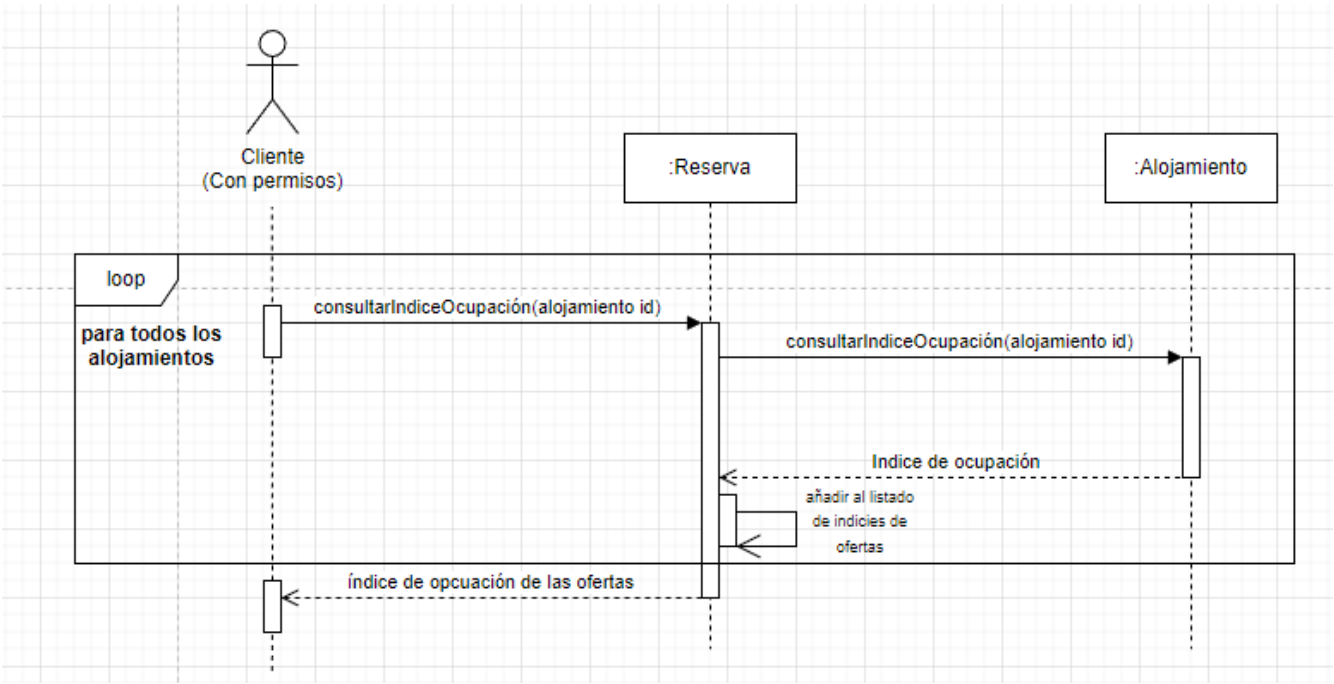
RF6


Operador
:Empresa /
:OperadorUsuario
:Alojamiento
iniciarSesion()
Respuesta
validarCredenciales()
eliminarAlojamiento(id)
Respuesta

RFC2



RFC3



3.2 Implementación de requerimientos no funcionales

El requerimiento no funcional N°1 o también llamado requerimiento de “privacidad” se cumple para nuestro modelo de datos debido a que debido a la manera que este modelado el UML hace que los usuarios de AlohAndes solo pueden manipular y consultar información

propia como lo son sus datos personales y así mismo los datos de las entidades que les pertenecen (Sus alojamientos / rentas).

El requerimiento no funcional N°2 o “persistencia”, esta presenta desde todo aspecto de todo el diseño de nuestro modelo ya que, en la mayoría de las entidades, la persistencia de los datos es indispensable a tal punto de que si este no estuviera presente el sistema simplemente no podría funcionar. Por ejemplo, ¿qué pasaría si los datos de los clientes no fueran persistentes? Pues lo más lógico es que los datos se perderían, muchas reservas realizadas bajo ese usuario se perderían y el negocio sufriría graves desgarros. El modelo facilita la implementación debido al seguimiento del uso de las propiedades ACID lo que hace a nuestro modelo ideal para que sus entidades y sus atributos sean persistentes sin

El requerimiento no funcional N°4, conocido como “concurrencia” se cumple para nuestro modelo de datos puesto que hacemos uso de la transaccionalidad. Gracias a eso, podemos hacer uso las propiedades ACID (*atomicity, consistency, isolation, durability*), más específicamente del aislamiento (*isolation*), el cual cumple que un cambio no afecte a otros que se estén ejecutando al mismo tiempo. Así, se hace posible que varios requerimientos se ejecuten al tiempo sin afectar la veracidad ni la consistencia de los datos.

En lo que respecta al requerimiento no funcional N°5, llamado “distribución”, nuestro modelo de datos facilita que la base de datos de la aplicación este centralizada. Esto debido a la normalización con la cuenta. Así, nuestro modelo se encuentra organizado de manera coherente lo que permite que varios componentes del sistema accedan a la base de datos sin que existan inconsistencias.

4 Bibliografía

1. **Harrington, Jan L.** *Relational Database Design and Implementation.* 2016. Fourth Edition.

2. **COMIT.** NORMALIZACIÓN. [En línea] Universidad de Los Andes.