

Iteración 2 Manejo Transaccional de Información

Sebastián García 201630047, Nicolas Sotelo 201623026

Grupo D-05

Sistemas Transaccionales

Ingeniería de Sistemas y Computación

{js.garcia1, n.sotelo}@uniandes.edu.co

Abril 24 de 2018

Tabla de contenido

1	Introducción	1
2	Diseño y Construcción de la Aplicación.....	1
2.1	Modificaciones al modelo conceptual.....	1
2.2	Lógica de los requerimientos funcionales.....	6
3	Balance de pruebas	7
4	Consideraciones.....	11
5	Resultados	11

1 Introducción

El objetivo de la siguiente iteración es desarrollar habilidades en el proceso de diseño de una aplicación transaccional, a partir de la descripción de un caso de negocio. En este proyecto, el caso de negocio gira en torno al alojamiento de la comunidad universitaria: Alohandes. Alohandes es un facilitador de opciones de alojamiento para la comunidad universitaria.

La principal meta de la siguiente iteración es integrar requerimientos funcionales y no funcionales relacionados con los aspectos ACID de una aplicación transaccional desarrollada en una arquitectura de tres niveles con manejo de persistencia en base de datos.

2 Diseño y Construcción de la Aplicación

2.1 Modificaciones al modelo conceptual

Para realizar los requerimientos funcionales de la presente iteración 2, se agregaron atributos a las relaciones *Reservas* y *Propuestas* para facilitar la consulta de datos y poder llevar a cabo requerimientos de modificación más complejos. En la relación *Reserva* se agregó el atributo *ID_COLECTIVO* que representa el identificador de una reserva realizada de forma colectiva. Si una reserva se realiza de forma conjunta con otras, las anteriores tendrán en común el atributo que los identifica como una reserva colectiva, sin perder su identificador como reserva individual. En el caso de que una reserva se realice de forma individual, el valor de su atributo *ID_COLECTIVO* será *NULL*.

<p>El usuario debe estar registrado como operador. El cliente debe enviar la reserva colectiva según el formato especificado, en el cual se listan la cantidad de reservas que se desean realizar. La base datos esta inicializada</p>		
Flujo normal de Eventos		
	Usuario	Sistema
1	Usuario envía la información en la cual se informa las reservas que se desean hacer especificando que alojamiento desea ,la cantidad de reservas deseadas que se quieren realizar y la información de cada uno de los usuarios que participan de esta	Verifica que exista en la base de datos y asimismo valida que este registrado como operador. Si no lo encuentra envía el caso de excepción #1.
2		Realiza cada una de las sub-reservas que participan de la reserva colectiva En caso de que no se puedan realizar las reservas en un solo tipo de alojamiento se le informa al usuario que no se cumplen para este sin embargo la transacción continua
3		Se le informa al usuario que se realizó la reserva colectiva de manera exitosa mostrándole cada una de las sub-reservas registradas en el sistema
Post-condiciones principales del caso de uso		
Se modifican las tablas RESERVA, PROPUESTA. En las cuales se realizó el proceso de reserva colectiva		
Caminos de Excepción		
Caso 1: el sistema envía un mensaje diciendo “no existe un operador registrado con ese identificador” y finaliza el caso de uso.		
Caso 2:El sistema le envía al usuario una excepción si no se pudo realizar la reserva colectiva		

<p>CU09:Deshabilitar oferta de alojamiento</p> <p>Esta operación permite a un usuario de tipo operador deshabilitar una oferta de alojamiento es decir que no se encuentra disponible para realizar futuras reservas</p>		
Entidades Involucradas		
PROPUESTAS RESERVAS ALOJAMIENTO CLIENTE OPERADOR		
Precondiciones		
<p>El usuario debe estar registrado como operador. La oferta de alojamiento debe estar registrada en la base de datos La base datos esta inicializada</p>		
Flujo normal de Eventos		
	Usuario	Sistema

1	Usuario envía el id que es asociada con una oferta de alojamiento	Verifica que exista la oferta de alojamiento
2		Deshabilita la oferta de alojamiento y reubica sus reservas en diferentes alojamientos de alohandes dando prioridad a las activas en el momento de la operación
3		Se le informa al usuario que se deshabilito la oferta de alojamiento y en donde se reubicaron las reservas.
Post-condiciones principales del caso de uso		
Se modifican las tablas RESERVA, PROPUESTA. En las cuales se realizó el proceso de cancelación de reserva colectiva.		
Caminos de Excepción		
Caso 1: el sistema envía un mensaje diciendo “no existe reserva colectiva con ese id” y finaliza el caso de uso.		
Caso 2 :El sistema le envía al usuario una excepción si hubo inconvenientes al momento de deshabilitar la oferta de alojamiento		

CU10: Rehabilitar una oferta de alojamiento.		
Esta operación permite a un operador de tipo operador rehabilitar una oferta de alojamiento		
Entidades Involucradas		
PROPUESTAS CLIENTE OPERADOR		
Precondiciones		
El usuario debe estar registrado como operador. La propuesta debe estar deshabilitada La propuesta debe existir en la base de datos La base datos esta inicializada		
Flujo normal de Eventos		
	Usuario	Sistema
1	Usuario envía la información con respecto al alojamiento que desea rehabilitar	Verifica que exista en la base de datos la oferta de alojamiento que se desea rehabilitar
2		Rehabilita la oferta de alojamiento
3		Se le informa al usuario que se rehabilito la oferta de alojamiento
Post-condiciones principales del caso de uso		
Se modifica la tabla PROPUESTA. En las cuales se realizó el proceso de rehabilitación		
Caminos de Excepción		
Caso 1: el sistema envía un mensaje diciendo “no existe alojamiento” en caso de que no exista y finaliza el caso de uso.		
Caso2: Manda la excepción en caso de que encuentre inconvenientes al momento de rehabilitar la propuesta.		

Requerimientos de consulta:

CU01: ANALIZAR LA OPERACIÓN DE ALOHANDES		
Para unidad de tiempo definido muestra cuales fueron las fechas de mayor demanda ,las de mayor recaudación, y las de menor demanda dentro de todo el sistema		
Entidades Involucradas		
PROPUESTAS RESERVAS		
Precondiciones		
El usuario debe estar registrado como cliente La base datos esta inicializada		
Flujo normal de Eventos		
	Usuario	Sistema
1		Realizo los cálculos y operaciones para cumplir con el requerimiento de consulta
Post-condiciones principales del caso de uso		
Camino de Excepción		

CU02: ENCONTRAR LOS CLIENTES FRECUENTES		
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		
Entidades Involucradas		
PROPUESTAS ALOJAMINETO RESERVAS CLIENTES		
Precondiciones		
El usuario debe estar registrado como cliente La base datos esta inicializada Tiene que existir el alojamiento en la base de datos		
Flujo normal de Eventos		
	Usuario	Sistema
1		Realizo los cálculos y operaciones para cumplir con el requerimiento de consulta
Post-condiciones principales del caso de uso		
Camino de Excepción		

CU03: ENCONTRAR LAS OFERTAS DE ALOJAMIENTO QUE NO TIENEN MUCHA DEMANDA		
Encontrar las ofertas de alojamiento que no han recibido clientes en periodos superiores a 1 mes, durante todo el periodo de operación de AlohanDes.		

Entidades Involucradas		
PROPUESTAS ALOJAMIENTO RESERVAS		
Precondiciones		
El usuario debe estar registrado como cliente La base datos esta inicializada		
Flujo normal de Eventos		
	Usuario	Sistema
1		Realizo los cálculos y operaciones para cumplir con el requerimiento de consulta
Post-condiciones principales del caso de uso		
Caminos de Excepción		

Tabla 1. Casos de uso para la iteración 2.

2.2 Lógica de los requerimientos funcionales

- RF 7 Registrar Reserva Colectiva

Para este requerimiento se codificó un método que consume un archivo `JSON` en donde se especifica: una lista de usuarios que hacen parte de la reserva colectiva (en donde cada usuario especifica el id de su reserva individual y la cantidad de personas que ocuparan el inmueble que se reserva a nombre del usuario respectivo), el identificador de la reserva colectiva, la fecha de inicial de ocupación del inmueble, el tipo de inmueble deseado, la duración de la reserva, la privacidad (un inmueble compartido o sencillo) y una lista de servicios deseados.

En primer lugar, se buscan todas las propuestas (ofertas) que cumplan con las condiciones especificadas (tipo de inmueble y servicios deseados). Si el sistema no cuenta con las suficientes propuestas, se le informa por medio de un mensaje al usuario la cantidad de propuestas disponibles y se realizan las reservas respectivas para esas propuestas. Como ejemplo del caso anterior, si el usuario quiere reservar 10 apartamentos sencillos pero el sistema solo cuenta con 6 apartamentos, luego se le informa al usuario que se realizarán solamente 6 de las 10 reservas que inicialmente solicitó y finalmente el sistema continua con el proceso normal de registrar una reserva.

En caso de que se realicen las reservas de forma exitosa se finaliza el requerimiento con un `COMMIT`. De lo contrario se aborta la anterior operación con una `ROLLBACK` en la base de datos. De la misma forma, los siguientes requerimientos realizan sus respectivas verificaciones para decidir su deben finalizar guardando o abortando las modificaciones a la base de datos.

- RF 8 Cancelar Reserva Colectiva

Este es el proceso inverso al requerimiento anterior. En este caso se hace uso del nuevo atributo creado en la relación `RESERVAS` para obtener el identificador de la reserva colectiva que se piensa cancelar. Luego, haciendo uso del requerimiento funcional de la iteración 1 que cancela reservas, se procede a eliminar las respectivas reservas que se hicieron de manera

conjunta y se procede a penalizar a los respectivos usuarios de cada reserva con una multa en el caso de ser necesario.

- RF 9 Deshabilitar una propuesta

El requerimiento funcional 9 recibe como parámetro el identificador de la propuesta que se piensa deshabilitar. Por medio del atributo `ID_PROPOSTA` presente en la relación `RESERVAS`, se obtienen las reservas que se verán afectadas por la cancelación de la propuesta. Luego se procede a buscar propuestas que estén disponibles y que sean del mismo tipo de inmueble que la propuesta original para comenzar a reubicar las reservas afectas. Primero se reubican las reservas vigentes, aquellas reservas o reserva que hacían uso del inmueble en el momento en el que se decidió deshabilitarlo. Por medio de un comparador se organizan las reservas por prioridad para ser reubicadas antes que las demás. La prioridad es establecida por la fecha en la que se registró en el sistema la reserva. Luego, se procede a realizar el anterior proceso con las reservas colectivas afectadas y finalmente con las reservas generales.

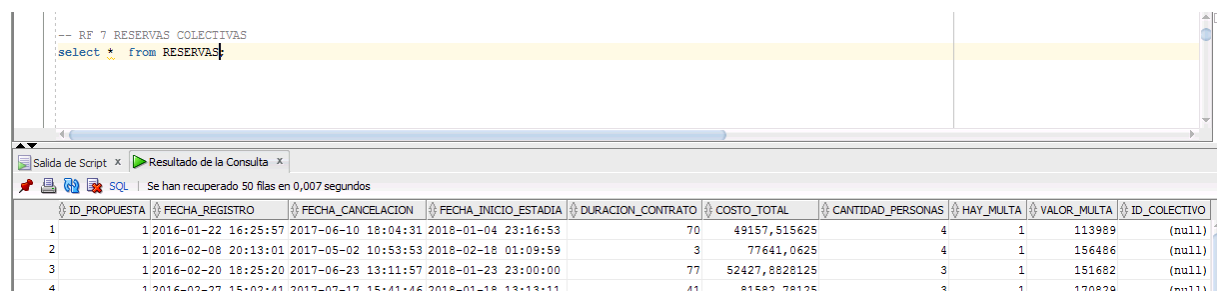
- RF 10 Rehabilitar una propuesta

Por medio del atributo `DISPONIBLE` en la relación `PROPUESTAS` se pueden obtener las propuestas que se encuentran disponibles o no. En el anterior requerimiento, el valor de este atributo es modificado para avisarle al sistema que no se encuentra disponible. De forma análoga, al rehabilitar una propuesta, se modifica el valor de este atributo nuevamente para poder recibir reservas en el futuro.

3 Balance de pruebas

Para realizar pruebas de transacción exitosa se realizaron los siguientes casos.

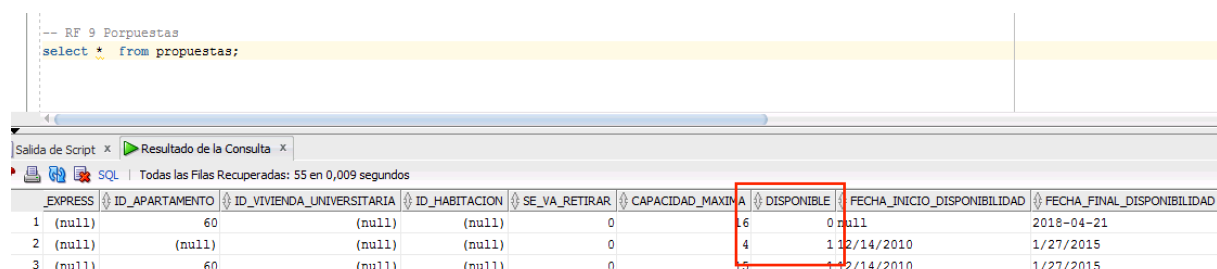
1. Estado inicial de la base de datos.



```
-- RF 7 RESERVAS COLECTIVAS
select * from RESERVAS;
```

ID_PROPOSTA	FECHA_REGISTRO	FECHA_CANCELACION	FECHA_INICIO_ESTADIA	DURACION_CONTRATO	COSTO_TOTAL	CANTIDAD_PERSONAS	HAY_MULTA	VALOR_MULTA	ID_COLECTIVO
1	2016-01-22 16:25:57	2017-06-10 18:04:31	2018-01-04 23:16:53	70	49157,515625	4	1	113989	(null)
2	2016-02-08 20:13:01	2017-05-02 10:53:53	2018-02-18 01:09:59	3	77641,0625	4	1	156486	(null)
3	2016-02-20 18:25:20	2017-06-23 13:11:57	2018-01-23 23:00:00	77	52427,8828125	3	1	151682	(null)
4	2016-02-27 15:02:41	2017-07-17 15:41:46	2018-01-18 13:13:11	41	81582,78125	3	1	170829	(null)

Figura 2. Estado inicial de la base de datos para los requerimientos 7 y 8. Es importante observar el valor de la columna `ID_COLECTIVO`.



```
-- RF 9 Propuestas
select * from propuestas;
```

EXPRESS	ID_APARTAMENTO	ID_VIVIENDA_UNIVERSITARIA	ID_HABITACION	SE_VA_RETIRAR	CAPACIDAD_MAXIMA	DISPONIBLE	FECHA_INICIO_DISPONIBILIDAD	FECHA_FINAL_DISPONIBILIDAD
1	(null)	60	(null)	(null)	0	1	0	2018-04-21
2	(null)	(null)	(null)	(null)	0	4	1	2015-01-27
3	(null)	60	(null)	(null)	0	1	2015-01-27	2015-01-27

Figura 3. Estado Inicial de la base de datos para los requerimientos 9 y 10.

2. Los datos involucrados en la operación transaccional solicitada.

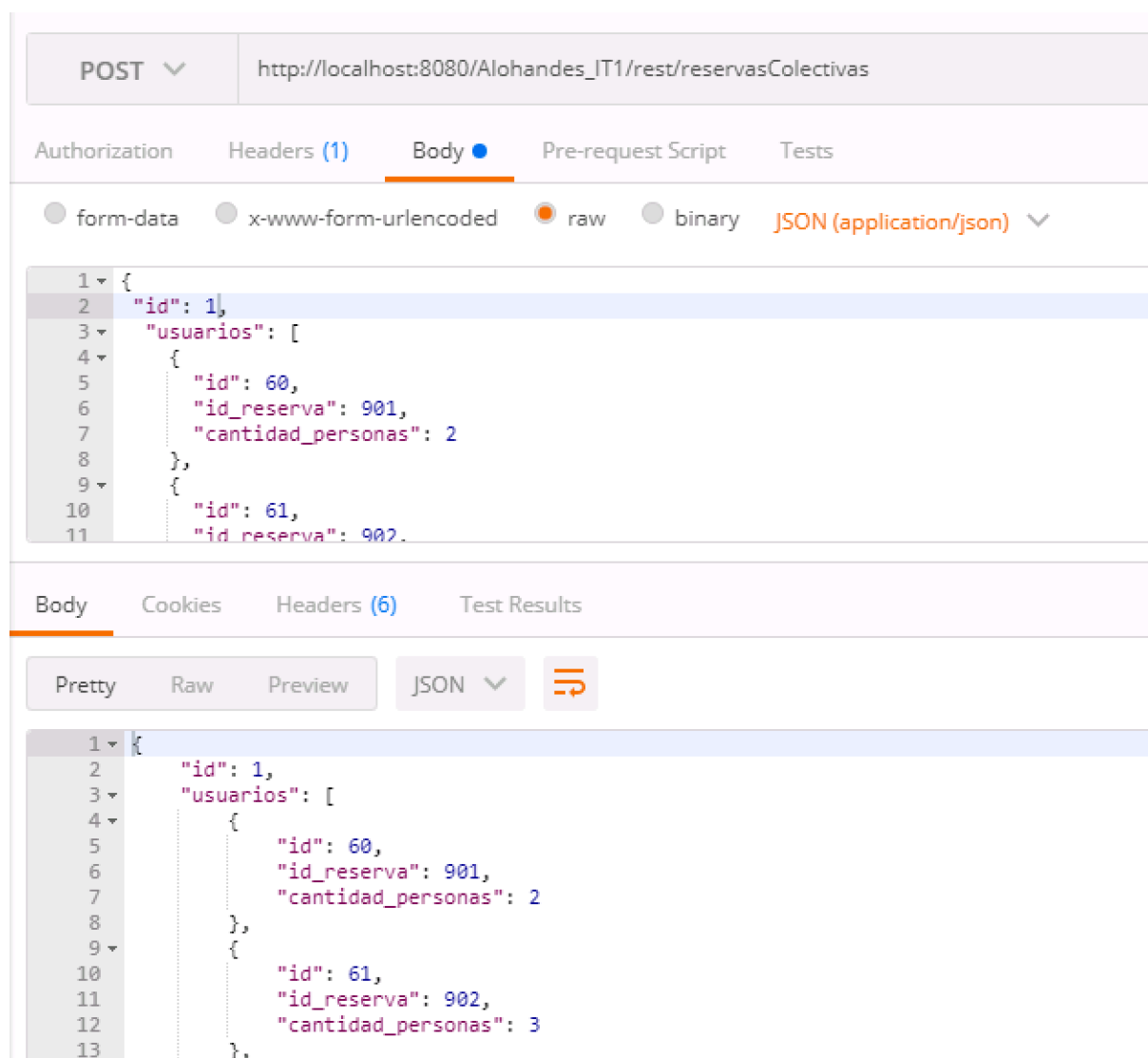


Figura 4. Datos involucrados para realizar el requerimiento 7 de registrar una reserva colectiva. Se ingresan los datos necesarios en formato JSON y se retorna la reserva realizada.

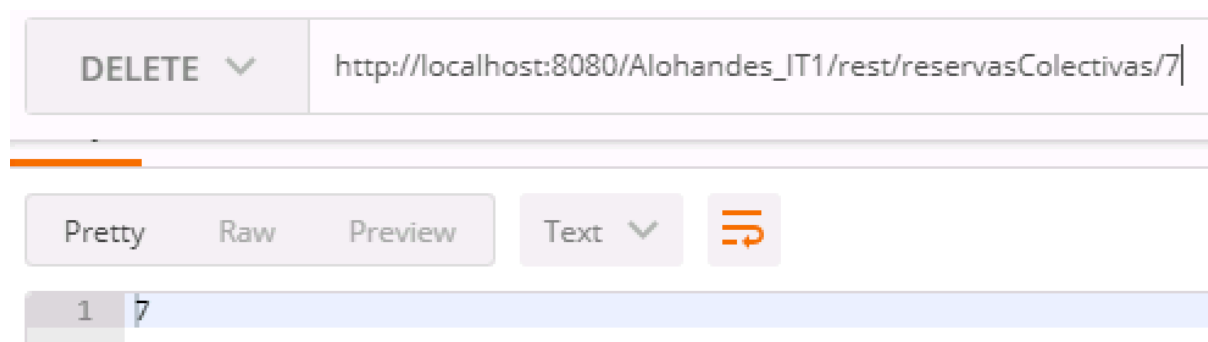


Figura 5. Datos involucrados en el requerimiento funcional 8. Se cancela una reserva colectiva por el identificador de la misma.

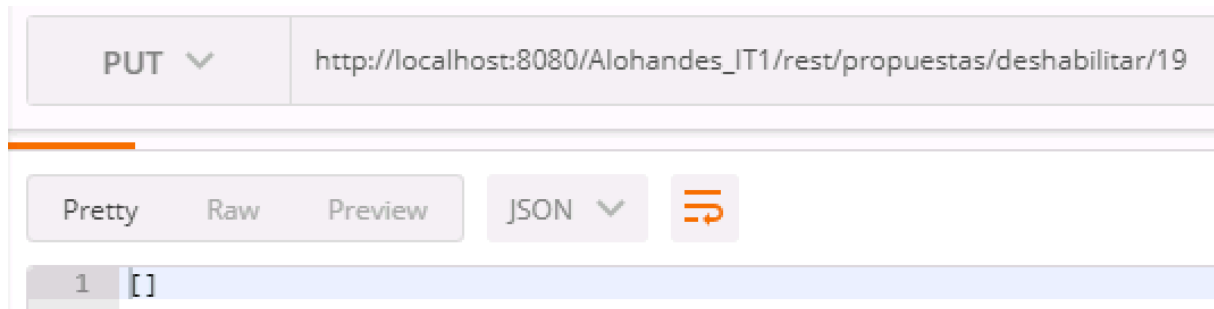


Figura 6. Datos involucrados en el requerimiento funcional 9 de deshabilitar una propuesta (oferta).

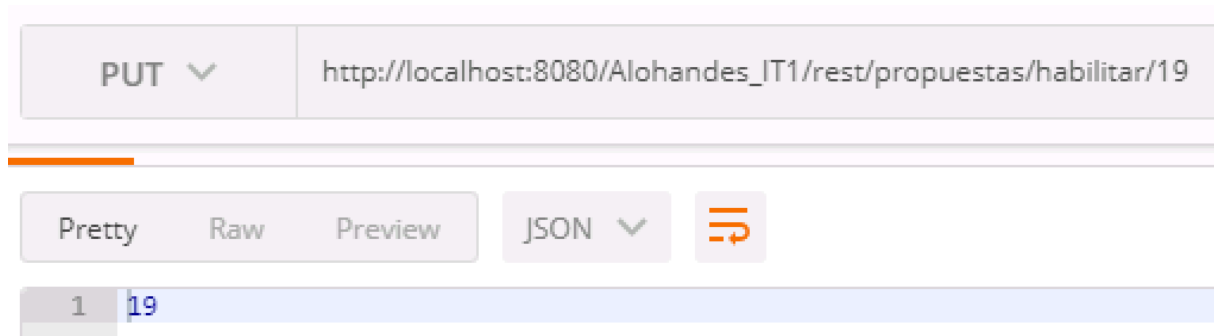


Figura 7. Datos involucrados en el requerimiento funcional 10 de rehabilitar una propuesta (oferta).

3. El estado final de la base de datos.

```
-- RF 9 Propuestas
select * from reservas order by id_colectivo;
```

Salida de Script x Resultado de la Consulta x

SQL | Se han recuperado 50 filas en 0,007 segundos

ID_PROPOSTA	FECHA_REGISTRO	FECHA_CANCELACION	FECHA_INICIO_ESTADIA	DURACION_CONTRATO	COSTO_TOTAL	CANTIDAD_PERSONAS	HAY_MULTA	VALOR_MULTA	ID_COLECTIVO
1	110 2018-04-24 10:28:22	(null)	2018-05-14	30	843	1	0	0	1
2	22 2018-04-24 10:28:22	(null)	2018-05-14	30	843	3	0	0	1
3	220 2018-04-24 10:28:22	(null)	2018-05-14	30	843	2	0	0	1

Figura 7. Estado final de la base de datos para el requerimiento funcional 7, se observa el cambio de valor en el atributo ID_COLECTIVO.

```
select id, tipo_inmueble, disponible from propuestas where id = 19 order by disponible desc;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,002 segundos

ID	TIPO_INMUEBLE	DISPONIBLE
1	19 Hostel	0

Figura 8. Estado final luego de deshabilitar una propuesta (requerimiento funcional 9) por identificador.

```
select id, tipo_inmueble, disponible from propuestas where id = 19 order by disponible desc;
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,004 segundos

ID	TIPO_INMUEBLE	DISPONIBLE
1	19 Hostel	1

Figura 9. Estado final luego de rehabilitar (requerimiento funcional 10) la propuesta con identificador 9.

Para realizar pruebas de transacción no exitosa se realizaron los siguientes casos.

```
{
  "id": 5,
  "usuarios": [
    { },
    { },
    { },
    { },
    {
      "id": 64,
      "id_reserva": 905,
      "cantidad_personas": 2
    },
    {
      "id": 65,
      "id_reserva": 906,
      "cantidad_personas": 2
    }
  ],
  "duracion": 30,
  "fecha_inicio_estadia": "2018-05-14",
  "tipo_inmueble": "Apartamento",
  "privacidad": "sencilla",
  "cantidad_inmuebles": 10,
  "servicios_deseados": [
    "luz",
    "tv",
    "internet"
  ]
}
```

Figura. 10 condiciones iniciales para que el requerimiento funcional 7 no se cumpla

```
[ALOHANDES APP] Attempting Connection to: jdbc:oracle:thin:@fn3.oracle.virtual.uniandes.edu.co:1521:prod
SELECT P.ID FROM PROPUUESTAS P WHERE UPPER(TIPO_INMUEBLE) = UPPER('Apartamento') AND P.ID_Apartamento IN
El sistema no cuenta con los suficientes inmuebles que se requieren. # Apartamentos = 4
```

Figura. 10.1 finalización de la transacción puesto ya que no se cumplió alguna regla de negocio al momento de realizar la transacción .La transacción hace rollback y no se modifica la base de datos

http://localhost:8080/Alohandes_IT1/rest/reservasColectivas/5

Figura. 11 condiciones iniciales para que el requerimiento funcional 8 no se cumpla

```
[ALOHANDES APP] Attempting Connection to: jdbc:oracle:thin:@fn3.oracle.virtual.uniandes.edu.co:1521:prod - By User: ISIS2304
SELECT * FROM RESERVAS R WHERE R.ID_COLECTIVO = 5
oracle.jdbc.driver.OraclePreparedStatementWrapper@2ed3194c
FAIL ELIMIANDO DESDE RF8
```

Figura. 11.1 finalización de la transacción puesto ya que no se cumplió alguna regla de negocio al momento de realizar la transacción .La transacción hace rollback y no se modifica la base de datos

```
http://localhost:8080/Alohandes_IT1/rest/propuestas/deshabilitar/199999999
```

Figura. 12 condiciones iniciales para que el requerimiento funcional 9 no se cumpla

```
Ejecutando operacion  
[ALOHANDES APP] Attempting Connection to: jdbc:oracle:thin:@fn3.oracle.virtual.uniandes.edu.co:1521:prod - By User: ISIS  
[EXCEPTION] General Exception:La propuesta que quiere rehabilitar no existe o ya se encuentra disponible en el sistema  
tm.BusinessLogicException: La propuesta que quiere rehabilitar no existe o ya se encuentra disponible en el sistema
```

Figura 12.1 finalización de la transacción puesto ya que no se cumplió alguna regla de negocio al momento de realizar la transacción .La transacción hace rollback y no se modifica la base de datos

```
http://localhost:8080/Alohandes_IT1/rest/propuestas/habilitar/199999999
```

Figura. 13 condiciones iniciales para que el requerimiento funcional 10 no se cumpla

```
[ALOHANDES APP] Attempting Connection to: jdbc:oracle:thin:@fn3.oracle.virtual.uniandes.edu.co:1521:prod - By User:  
[EXCEPTION] General Exception:La propuesta que quiere rehabilitar no existe  
tm.BusinessLogicException: La propuesta que quiere rehabilitar no existe
```

Figura 13.1 finalización de la transacción puesto ya que no se cumplió alguna regla de negocio al momento de realizar la transacción .La transacción hace rollback y no se modifica la base de datos

4 Consideraciones

En los documentos anexos se encuentran la colección de pruebas Postman (`docs > Iteracion 2 > Pruebas`). De la misma forma, en la misma carpeta se encuentra un archivo PDF que explica los casos de éxito y los casos de fallo para cada requerimiento funcional.

En la carpeta de SQL (`docs > Iteracion 2 > SQL`) se encuentran las sentencias SQL implementadas para: crear las tablas y sus atributos, poblar las tablas, realizar los requerimientos funcionales, y se encuentran las sentencias utilizadas para completar los requerimientos funcionales de consulta.

5 Resultados

Luego de realizar los anteriores requerimientos, se obtuvieron varios resultados. Con respecto a los requerimientos funcionales de modificación y de consulta, se logró que la base de datos hiciera las respectivas actualizaciones y modificaciones a ésta. Para conseguir la transacción, se utilizaron los comandos de `COMMIT` `ROLLBACK` para guardar o abortar los cambios que se generaban en la base datos dependiendo de los distintos escenarios que pueden surgir en un requerimiento. Uno de estos escenarios es de falla, en donde se especifica un identificador de propuesta (oferta) o de reserva que no son válidos, o la un error generado por el no cumplimiento de las reglas de negocio (como por ejemplo el que imposibilita a un cliente hacer más de una reserva por día).

Con respecto a los resultados no logrados, en la parte *front* de la aplicación se lograron desplegar toda la información necesaria, sin embargo, existen objetos que no se despliegan como clientes o propuestas faltantes. De la misma forma, al momento de realizar una reserva colectiva o una propuesta, en el *front* no se visualiza el objeto creado, sin embargo, en la base de datos es evidente la creación de los anteriores mencionados (por medio de SQL Developer).