Iteración 3

Sebastián García 201630047, Nicolas Sotelo 201623026

Grupo D-05

Sistemas Transaccionales

Ingenieria de Sistemas y Computación

{js.garcial1, n.sotelo[}@uniandes.edu.co](mailto:%7d@uniandes.edu.co)

Abril 24 de 2018

Tabla de contenido

[1 Introducción 1](#_Toc512333083)

[2 Diseño y Construcción de la Aplicación 1](#_Toc512333084)

[2.1 Modificaciones al modelo conceptual 1](#_Toc512333085)

[2.2 Lógica de los requerimientos funcionales 6](#_Toc512333086)

[3 Balance de pruebas 7](#_Toc512333087)

[4 Consideraciones 11](#_Toc512333088)

[5 Resultados 11](#_Toc512333089)

# 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.

# Diseño y Construcción de la Aplicación

## 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. El 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.

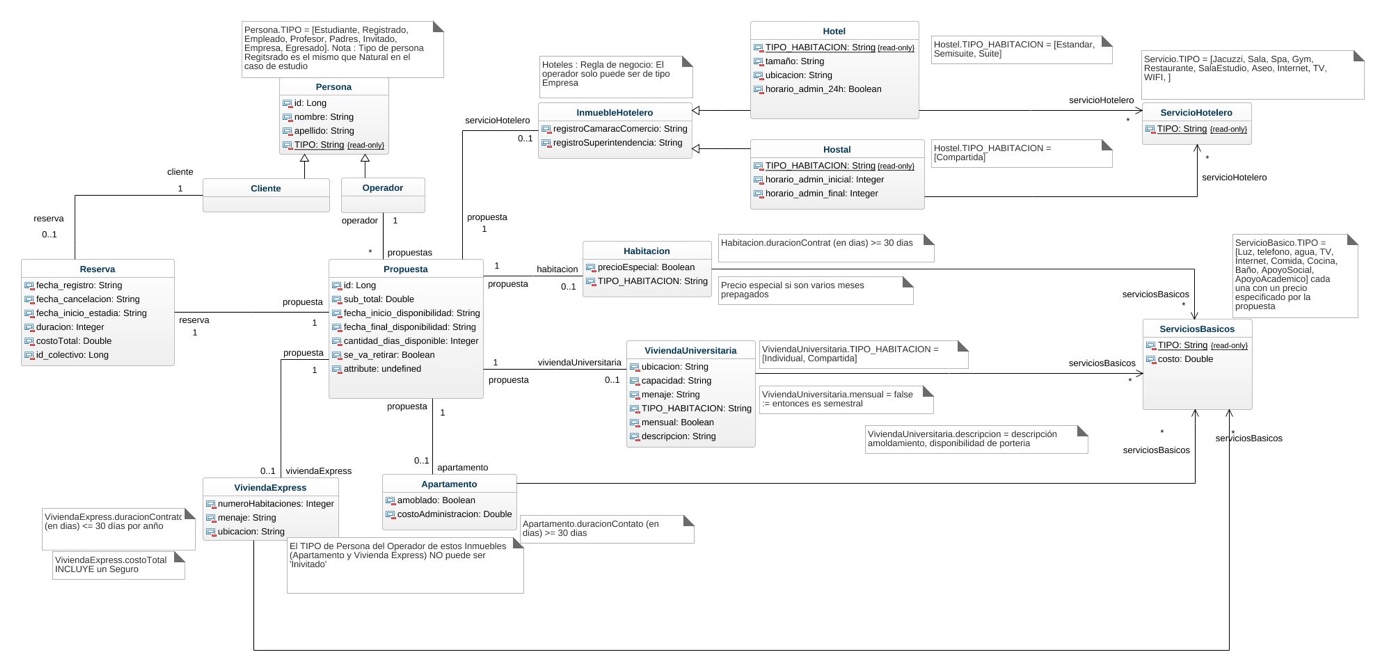


Figura 1. Modelo de clases UML actualizado luego de los resultados obtenidos en la iteración 1. (El modelo relacional se encuentra en docs > Iteracion 2 > modelos)

En el caso de la relación Propuestas (entendida como oferta) se crearon varios atributos que contienen la información de la fecha desde la cual una propuesta comienza a estar disponible, la fecha en la cual la propuesta deja de estar disponible, la cantidad de días de disponibilidad de una propuesta, el estado actual de la propuesta (disponible o no) y el estado a futuro (si se va a retirar la propuesta, teniendo reservas vigentes).

Cada vez que se registra una reserva, el atributo de FECHA\_FINAL\_DISPONIBILIDAD cambia de valor. Similarmente, cuando se da por culminada una reserva, el valor del atributo FECHA\_INICIO\_DISPONIBILIDAD se actualiza. Por medio de las funciones de SQL Developer: to\_char(), to\_number() y to\_date(), se obtiene la cantidad de días disponibles luego de realizar la diferencia en días entre el atributo FECHA\_FINAL\_DISPONIBILIDAD y el atributo FECHA\_FINAL\_DISPONIBILIDAD. Más adelante esta información será útil al momento se hacer el análisis sobre las propuestas.

Como se mencionó anteriormente, se agregaron los atributos SE\_VA\_RETIRAR y DISPONIBLE que se actualizan cada vez que se registra una reserva o se deshabilita un inmueble.

Para la siguiente iteración no se crearon nuevas relaciones.

|  |  |  |
| --- | --- | --- |
| Descripción General del Caso de Uso | | |
| **CU07: Realizar reserva colectiva**  Esta operación permite a un usuario realizar múltiples reservas  Respecto a un alojamiento deseado | | |
| Entidades Involucradas | | |
| PROPUESTAS  RESERVAS  ALOJAMIENTO  CLIENTE  OPERADOR | | |
| Precondiciones | | |
| 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 | | |
| 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 | | |

|  |  |  |
| --- | --- | --- |
| **CU09:Deshabilitar oferta de alojamiento**  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 | | |
| Entidades Involucradas | | |
| PROPUESTAS  RESERVAS  ALOJAMIENTO  CLIENTE  OPERADOR | | |
| Precondiciones | | |
| El usuario debe estar registrado como operador.  La oferta de alojamiento debe estar registrada en la base de datos  La base datos esta inicializada | | |
| 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 definidomuestra 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 | | |
|  | | |
| Caminos 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 | | |
|  | | |
| Caminos 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.

## 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\_PROPUESTA 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.

# Balance de pruebas

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

1. Estado inicial de la base de datos.

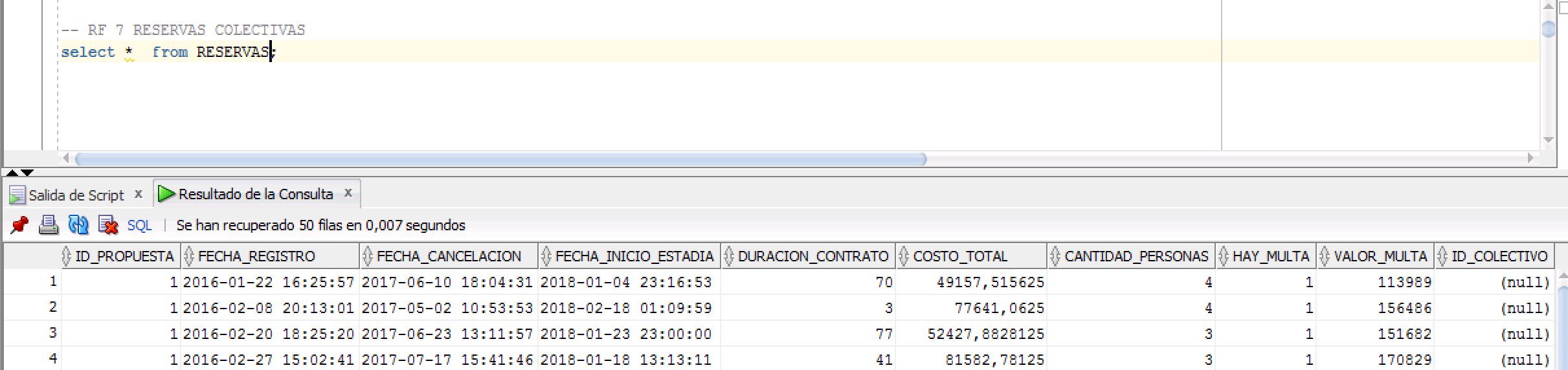


Figura 2. Estado inicial de la base de datos para los requerimientos 7 y 8. Es importante observar el valor de la columna ID\_COLECTIVO.

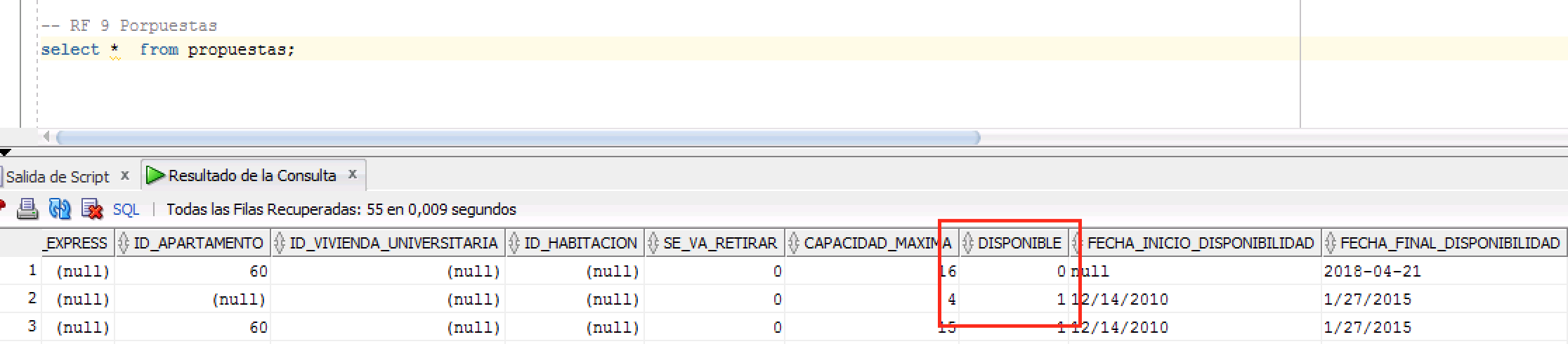


Figura 3. Estado Inicial de la base de datos para los requerimientos 9 y 10.

1. Los datos involucrados en la operación transaccional solicitada.

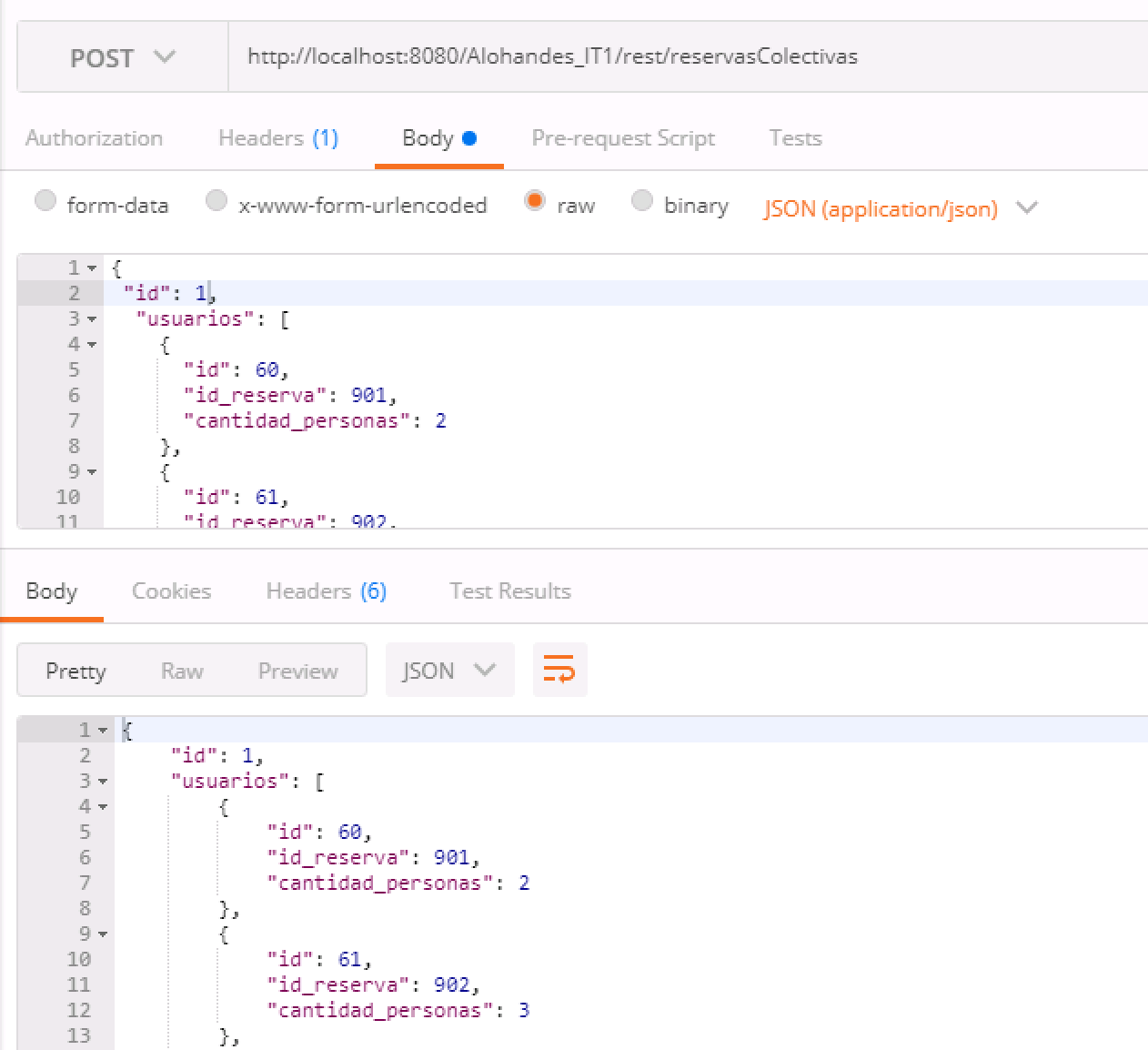


Figura 4. Datos involucrados para realizar el requerimiento 7 de registrar una reserva colectiva. Se ingresan los datos necesarios en formato JSON y se retorna la reserva realizada.

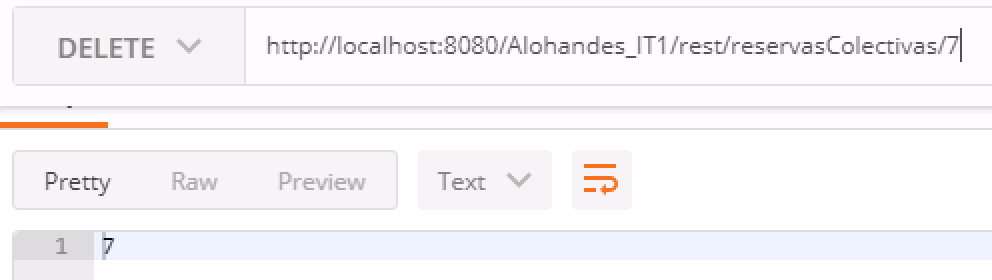


Figura 5. Datos involucrados en el requerimiento funcional 8. Se cancela una reserva colectiva por el identificador de la misma.

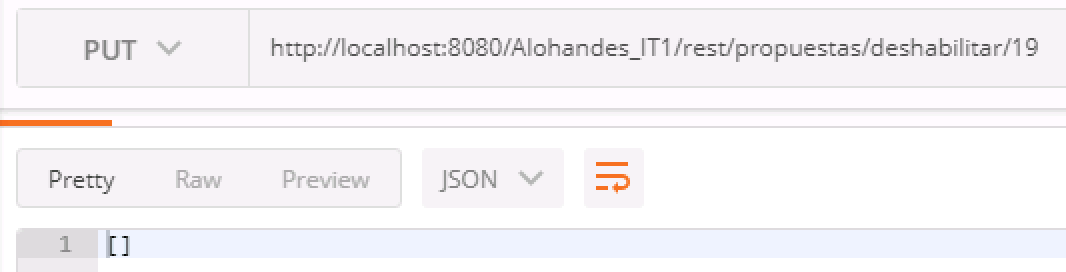


Figura 6. Datos involucrados en el requerimiento funcional 9 de deshabiliatr una porpuesta (oferta).

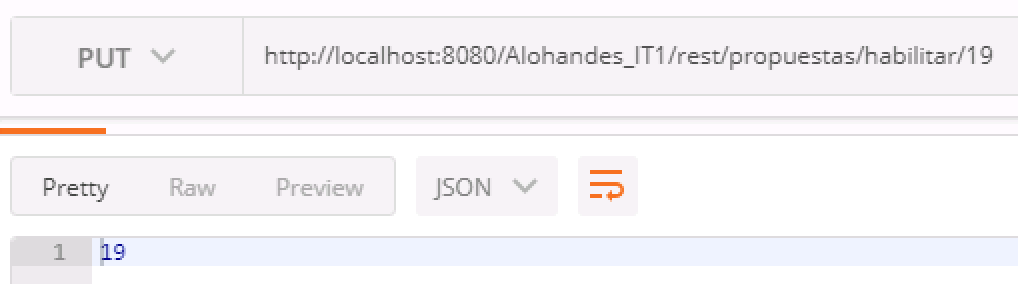
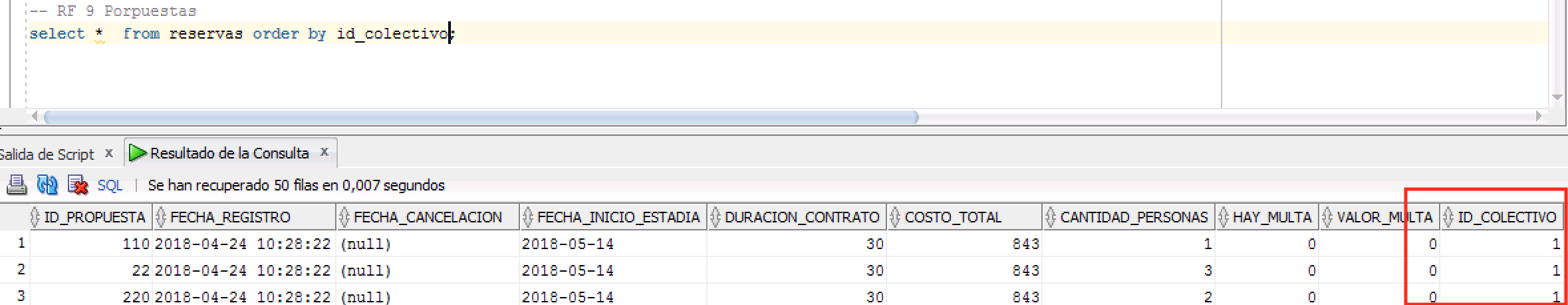


Figura 7. Datos involucrados en el requerimiento funcional 10 de rehabiliatr una porpuesta (oferta).

1. El estado final de la base de datos.

Figura 7. Estado final de la base de datos para el reqeurimiento funcional 7, se observa el cmabio de valor en el atributo ID\_COLECTIVO.

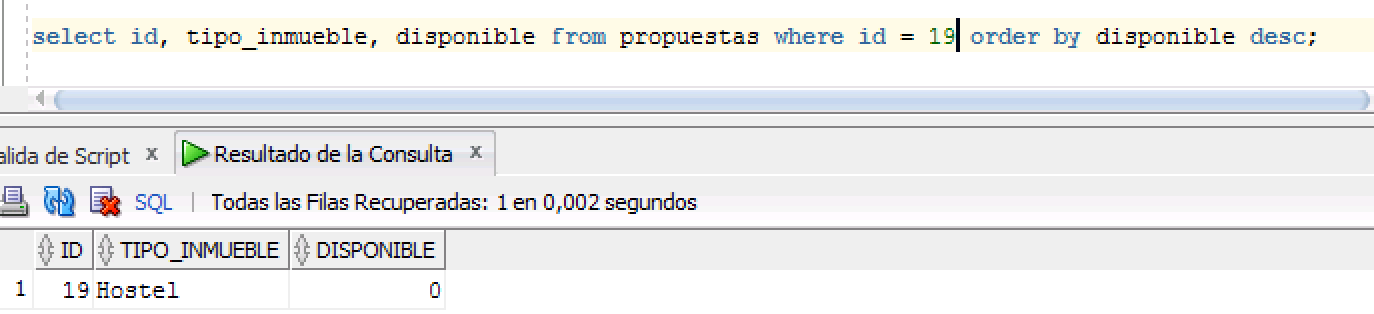


Figura 8. Estado final luego de deshabilitar una propuesta (requerimiento funcional 9) por identificador.

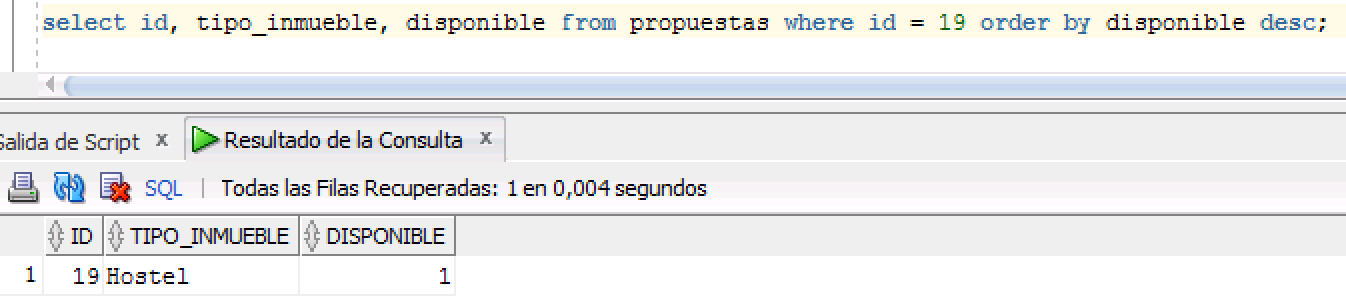


Figura 9. Estado final luego de rehabilitar (requerimiento funcional 10) la propuesta con identificador 9.

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



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

Macintosh HD:Users:Nicolas:Desktop:Fallo del  rf7 consola.png

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

Macintosh HD:Users:Nicolas:Desktop:condiciones fallo rf8.png

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

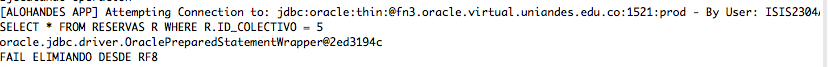


Figura. 11.1 finalización de la transacción puesto ya que no se cumplió alguna regla de negocio al momento de realizar la transacción .La transacción hace rollback y no se modifica la base de datos

Macintosh HD:Users:Nicolas:Desktop:condiciones fallo rf9.png

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

Macintosh HD:Users:Nicolas:Desktop:Fallo rf9 cosnola.png

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

Macintosh HD:Users:Nicolas:Desktop:condiciones fallo rf10 postman.png

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

Macintosh HD:Users:Nicolas:Desktop:Fallo consola rf10.png

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

# 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 encentran 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.

# 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 escenario 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).