**Caso estudio: ALOHANDES**

**Integrantes**

**-Cristian Caro: c.caro**

**-Sebastián Pedraza: js.pedrazap1**

Modelo conceptual trabajado

**Diagrama, Esquemático

Descripción generada automáticamente**

Modificaciones: Solo hubo una modificación que fue agregar el atributo de “disponibilidad” a la clase oferta para poder cumplir con el RF9 y 10 que deshabilitan/habilitan una oferta.

Modelo relacional en archivo de Excel

**RF1: Registrar operadores de Alohandes.** Fue durante la iteración 1 y se resolvió a través de crear nuevas instancias de Hotel, Hostal, Empresa o Propietario Inmueble.

**RF2: Registrar propuestas de alojamiento.** Se realizó a través de crear una nueva oferta asociada a un operador.

**RF3: Registrar personas habilitadas para usar los servicios.** Se basa en instanciar la clase cliente, pues se asume que hay un solo administrador.

**RF4: Registrar una reserva.** Se basa en modificar una instancia de la clase oferta de modo que el atributo “reservado” debe tener valor de 1

**RF5: Cancelar una reserva.** Se soluciona a través de modificar una instanciar de la clase oferta de modo que el atributo “reservado” tenga su valor en 0

**RF6: Retirar una oferta de alojamiento.** La solución está basada en eliminar una instancia de la clase oferta y de este modo, ningún usuario puede ver la oferta ya que está borrada

**RF7: Registrar reserva colectiva.** La solución está basada en pedir cuántas reservas se desean hacer. Una vez pasa esto, se llama ese número de veces el **RF4** que permite agregar una reserva.

**RF8: Cancelar reserva colectiva.** Se busca pedir cuantas reservas desea cancelar una reserva, al dar el número, se usa el requerimiento **RF5** que es cancelar una reserva el número de veces que ha dicho el usuario.

**RF9: Deshabilitar oferta de alojamiento.** Al haber incluido en la iteración 3 un nuevo atributo en la clase oferta llamado “disponibilidad”, este RF se soluciona por medio de modificar el valor del atributo a 0. Cuando pasa esto, dicha oferta no debe ser visible a ningún cliente y solo el administrador puede modificar su estado.

**RF10: Rehabilitar oferta de alojamiento.** Al haber incluido en la iteración 3 un nuevo atributo en la clase oferta llamado “disponibilidad”, este RF se soluciona por medio de modificar el valor del atributo a 1. De este modo, la oferta se puede visualizar normalmente.

**RFC1: Mostrar el dinero recibido por cada operador en el año de corrido.** Se realiza un cruce entre la tabla oferta y contrato. Oferta para saber qué operador fue y contrato para calcular el dinero recibido por cada operador para el cual exista al menos una oferta.

**RFC2: Mostrar las 20 ofertas más populares.** Se hace un conteo de las 20 mejores habitaciones e inmuebles a través de ver las apariciones.

**RFC3: Mostrar el índice de ocupación de cada una de las ofertas.** Se solucionó a través de un atributo a la clase oferta. Entonces, cada instancia de la clase lo tendrá. Se cada vez que se modifica el estado de “reservado”, pues indica que hubo una ocupación. La forma de calcular es a través de #ocupaciones por cada instancia / #ocupaciones totales.

**RFC4: Mostrar los alojamientos disponibles en un rango de fechas tal que tengan ciertos servicios.** Se soluciona pidiendo los servicios a proveer, estos se agregar a una lista. Luego, se busca acceder a las habitaciones y ver a través de la tabla “Brindan”, qué habitaciones brindan todos los servicios descritos.

**RFC5: Mostrar el uso de Alohandes para cada tipo de usuario.** La solución es implementada en la interfaz. Al inicio, cada persona que accede a la página debe ingresar “A” o “U” dependiendo si es User o Admin. Un admin tiene un menú que da acceso a todas las clases, puede crear, modificar y eliminar instancias de todas las clases. Por otro lado, un User solo puede hacer consultas y reservas.

**RFC6: Mostrar el uso de Alohandes dado un usuario.** Dado un ID de cliente se busca mostrarle el historial que tiene, es decir, las ofertas que ha reservado, contratos creados y demás.

**RFC7: Analizar la operación de Alohandes.** Este requerimiento se soluciona a través de que, para cada operador se busque la fecha que más aparecen en las instancias de contrato, mostrando así los días de mayor demanda. Por otro lado, las fechas de mayores ingresos son tales que la sumatoria de los precios finales de los contratos que se dieron en una misma fecha sean los máximos.

**RFC8: Encontrar clientes frecuentes.** Se realiza un conteo del número de clientes que han obtenido una misma oferta, es decir, adquieren el mismo servicio de alojamiento en diferentes fechas.

**RFC9: Encontrar ofertas de alojamiento no frecuentes.** Para este requerimiento se busca encontrar las ofertas de alojamiento de poca demanda. Esto se calcula buscando el número de ofertas asociadas que tiene cada operador. La comparación entre operadores es sobre las instancias de los mismos.

**RFC10: Consultar consumo alohandes.** Dadas dos fechas y el tipo de ordenamiento con su criterio, la fecha de inicio del contrato debe estar entre las fechas dadas. Luego, para los contratos que comenzaron en esas fechas se hace un cruce con la tabla cliente para conocer la información de los clientes que aparecieron al menos una vez. (Interfaz cliente consulta sobre cliente e interfaz administrador consulta sobre todos los clientes)

**RFC11: Consultar consumo alohandes v-2.** Es similar al **RFC10** Dadas dos fechas y el tipo de ordenamiento con su criterio, la fecha de inicio del contrato debe estar entre las fechas dadas. Luego, se buscan los clientes que no hicieron ninguna reserva en esas fechas. (Interfaz cliente consulta sobre cliente e interfaz administrador consulta sobre todos los clientes)

**RFC12: Consultar funcionamiento.** Se divide en varias partes. Por cada semana se debe: consultar oferta de alojamiento con menos ocupación. Operadores que hayan recibido más reservas y menos reservas.

**RFC13: Consultar buenos clientes.** Se buscan los clientes que hacen reservas al menos una vez al mes, entonces deben reservar una oferta al mes como mínimo. También, clientes cuyo costo de alojamiento sea mayor a 150 USD por noche. Finalmente, los clientes que reservan habitaciones de tipo suite.

**Reglas de negocio**

1. Los hostales deben indicar en su oferta los horarios de apertura y salida
2. El servicio ofrecido por personas naturales genera un contrato con mínima duración de un mes y puede establecer precios especiales por contratos de varios meses prepagados
3. Un propietario de inmueble debe ser un miembro de la comunidad (profesores, empleados, estudiantes, egresados o padres de estudiantes)
4. En un año no se puede utilizar la vivienda por más de 30 días para el servicio de ofrecer la vivienda propia
5. Para el servicio de vivienda propia no es necesario tener registro ante la cámara de comercio ni la superintendencia
6. Una vivienda universitaria debe ofrecer habitaciones por mínimo una noche
7. El operador debe brindar servicios externos e indicar cuales de ellos tienen costo adicional

**Boyce Cood Normal Form:** La razón por la cual el modelo esta en Boyce Cood es que todas las dependencias funcionales no triviales que involucran atributos no clave se mantienen en tablas separadas. Esto ayuda a evitar problemas de actualización, inserción y eliminación de datos que pueden surgir cuando las tablas no están diseñadas adecuadamente.

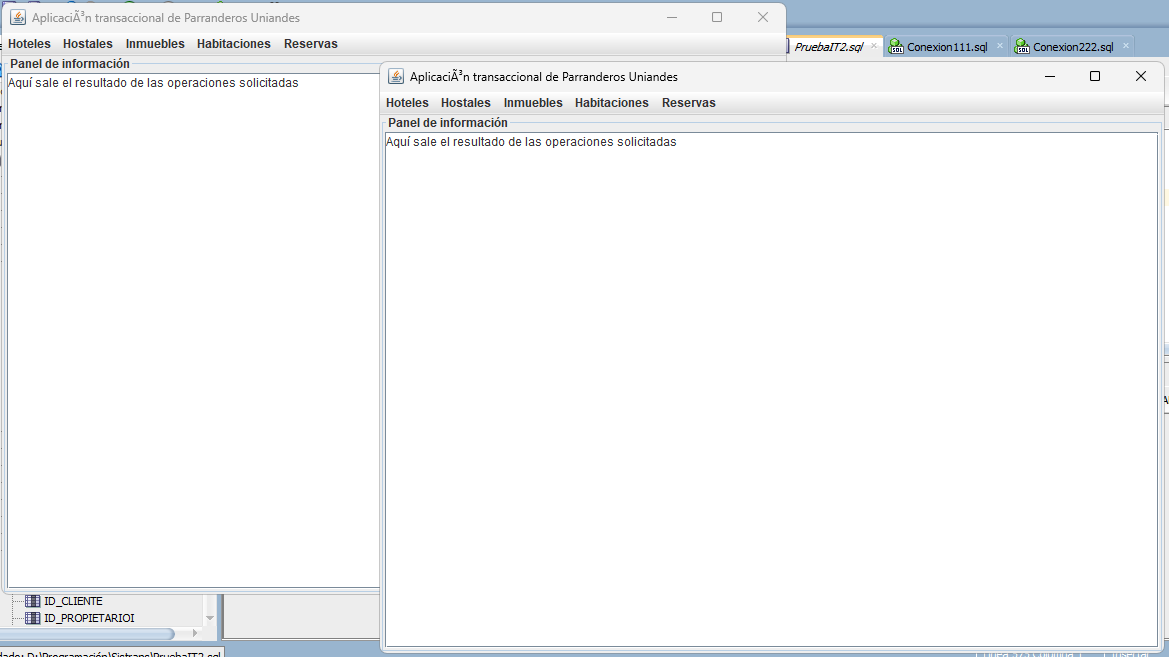
**Índices:** Para cada tabla se crearon índices primarios asociados al ID de cada una de las clases. La razón es debido a que siempre hacer consultas sobre los IDs y estos son únicos en las tablas entonces no pueden apuntar a dos registros. No se utiliza hash o árbol B+.

“CREATE UNIQUE INDEX PK\_HOTEL ON HOTEL (ID\_HT);”

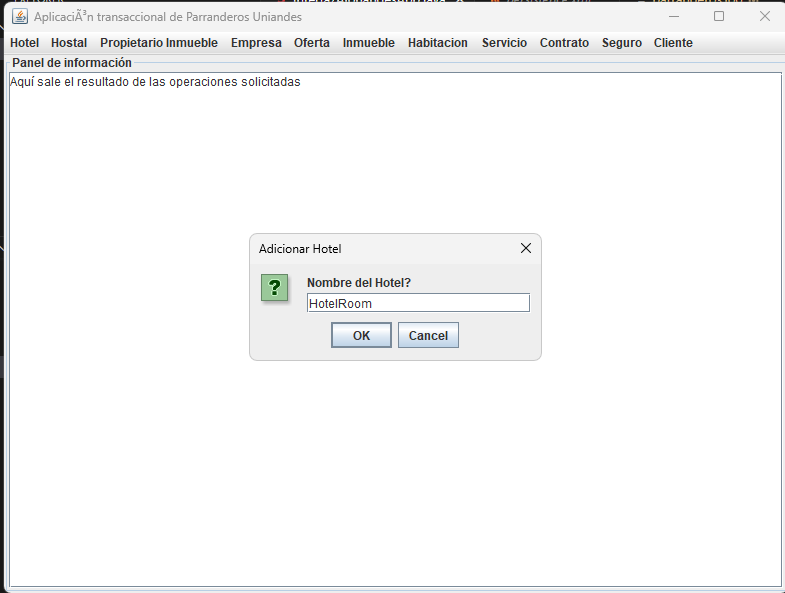
Escenarios y pruebas de FKS Y CKS (Archivo .sql añadido en el proyecto)

1. Inserción sobre hostal tal que la fecha de cierre sea menor a la fecha de apertura.
2. Insertar sobre propietario inmueble tal que el “vinculo” no exista dentro de las posibilidades
3. Insertar sobre propietario inmueble tal que el “tipo” no exista dentro de las posibilidades planteadas
4. Insertar un seguro tal que el id del inmueble no exista
5. Insertar sobre oferta tal que el id del cliente no exista
6. Insertar sobre oferta tal que el id del hotel no exista
7. Insertar sobre inmueble tal que el id de oferta no exista
8. Insertar sobre inmueble tal que “tipoI” no exista dentro de las posibilidades
9. Insertar sobre habitación tal que el id del inmueble no exista
10. Insertar sobre habitación tal que el “tipoH” no exista dentro de las posibilidades
11. Insertar sobre brindan tal que el id de la habitación no exista
12. Insertar sobre brindan tal que el id del servicio no exista

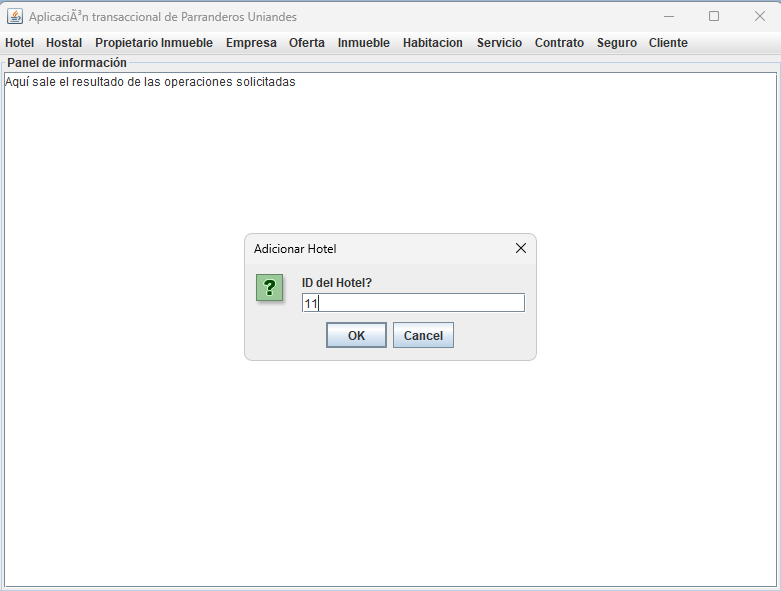
Seguimiento transaccional



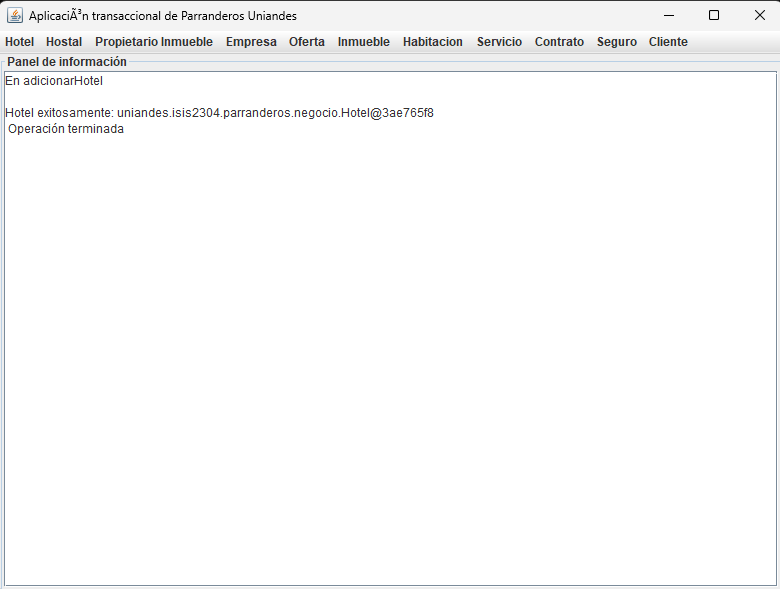
Img 1. Inician dos sesiones sobre la aplicación.



Img 2. Inicia una transacción desde la sesión 1.



Img 3. Asociar ID a una instancia de hotel desde la sesión 1.



Img 4. Inicia una transacción desde la sesión 1.

Interfaz de usuario gráfica

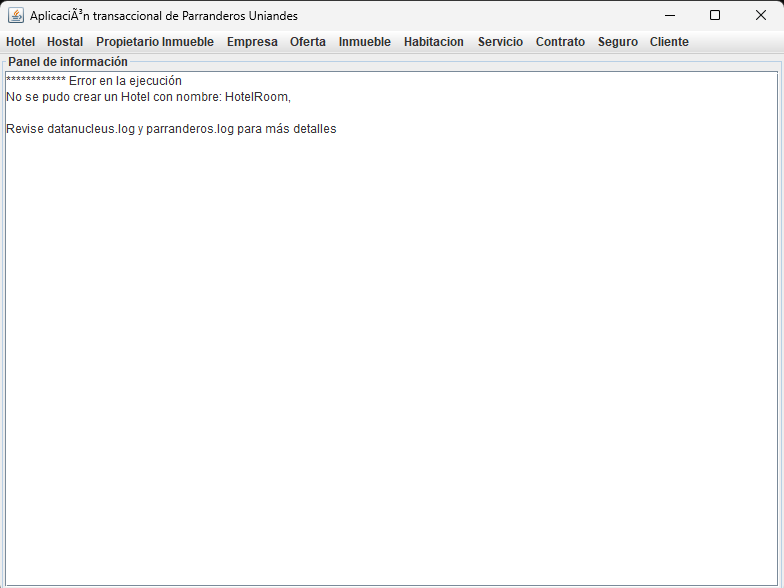
Descripción generada automáticamente

Img 5. Inicia una transacción desde la sesión 2.

Interfaz de usuario gráfica

Descripción generada automáticamente

Img 5. Crear una instancia de hotel desde la sesión 2.



Img 7. Falla la transacción desde la sesión 2.