“Manejo transaccional de información”

Juan Sebastián Díaz Serrano, Sergio Guzmán Mayorga

Iteración 3

Universidad de los Andes, Bogotá, Colombia

{js.diaz, s.guzmanm[}@uniandes.edu.co](mailto:%7d@uniandes.edu.co)

Fecha de presentación: Octubre 29 de 2017

Tabla de contenido

[1 Introducción 1](#_Toc497069829)

[2 Análisis 2](#_Toc497069830)

[3 Diseño de la aplicación 3](#_Toc497069831)

[3.1 Impacto de nuevos requerimientos 3](#_Toc497069832)

[3.2 Modelo relacional 3](#_Toc497069833)

[3.3 Documentación de cada tabla seguida por de su forma normal 4](#_Toc497069834)

[3.4 Cambios sustanciales de la Iteración 2 a la Iteración 3 18](#_Toc497069835)

[3.5 Documentación de la lógica de requerimientos nuevos y transaccionales 18](#_Toc497069836)

[4 Construcción de la aplicación 22](#_Toc497069837)

[4.1 RF9 22](#_Toc497069838)

[4.2 RF10 22](#_Toc497069839)

[4.3 RF11 22](#_Toc497069840)

[4.4 RF12 22](#_Toc497069841)

[4.5 RF13 22](#_Toc497069842)

[4.6 RF14 22](#_Toc497069843)

[4.7 RF15 22](#_Toc497069844)

[4.8 RF16 23](#_Toc497069845)

[4.9 RF17 23](#_Toc497069846)

[4.10 RFC7 23](#_Toc497069847)

[4.11 RFC8 23](#_Toc497069848)

[4.12 RNF3 23](#_Toc497069849)

[5 Consideraciones adicionales 24](#_Toc497069850)

[6 Análisis de resultados 24](#_Toc497069851)

[6.1 Aprendizajes y logros 24](#_Toc497069852)

[6.2 Conclusiones 24](#_Toc497069853)

[7 Bibliografía 24](#_Toc497069854)

# Introducción

En el presente documento se presentan los puntos clave para el desarrollo de las actividades planteadas en [1]. Para cumplir con esto, cada sección del presente texto hará referencia a las tareas a cumplir en el documento referenciado, a algunos puntos que pensamos que vale la pena aclarar respecto al uso de requerimientos, o a conclusiones y observaciones que se hayan presentado a lo largo de la iteración.

# Análisis

Para el modelo conceptual no se vio ningún cambio con respecto al presentado en la iteración 3. A saberse, se tiene la siguiente imagen:

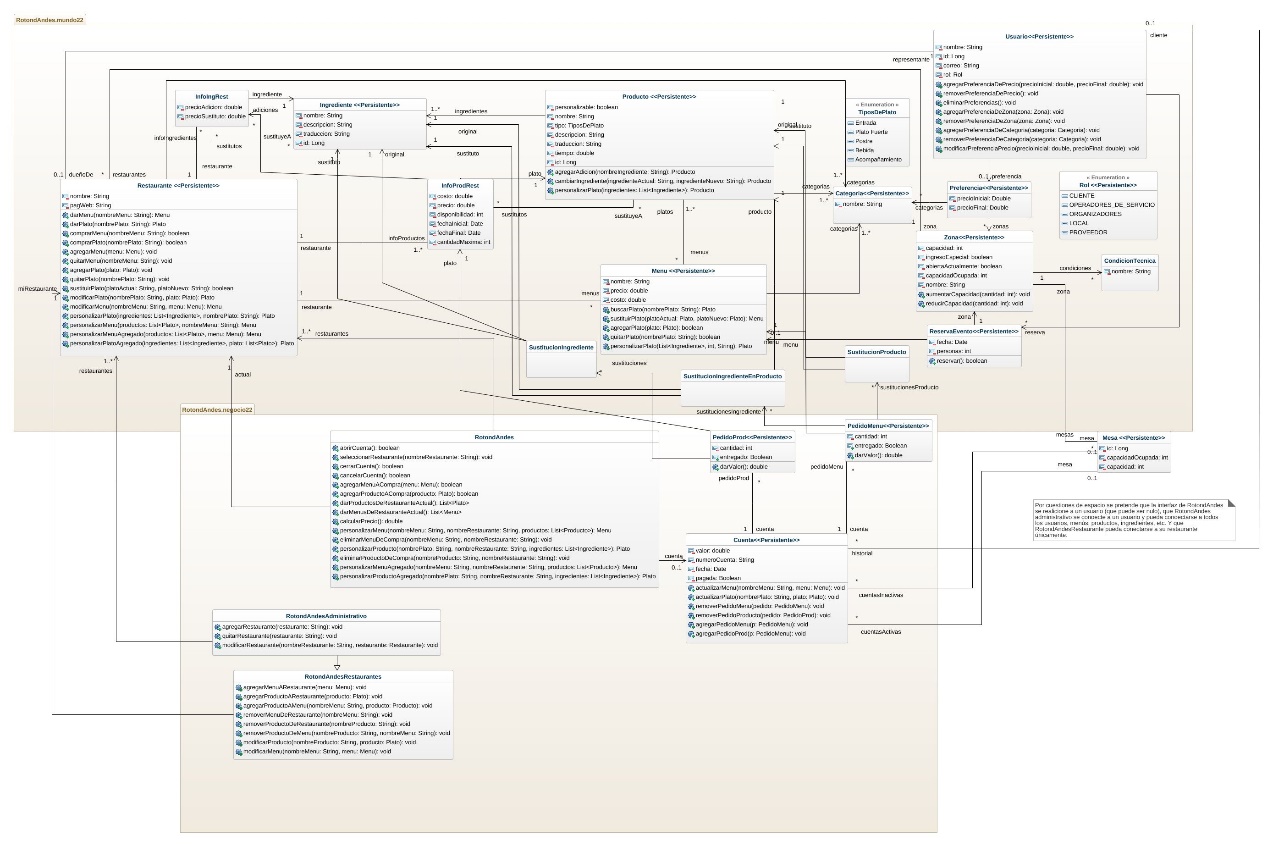


Figura 1. Modelo conceptual de la iteración 4

# Diseño de la aplicación

## Impacto de nuevos requerimientos

Como tal no se ven impactos de los nuevos requerimientos sobre la aplicación más allá de la creación de índices para responder a las consultas establecidas de forma eficiente. Sin embargo, este tema se trata más adelante en la subsección de diseño físico de la aplicación.

## Modelo relacional

Por lo dicho anteriormente, no se realizaron cambios sobre el modelo relacional establecido, es decir, se mantuvo el siguiente diagrama:

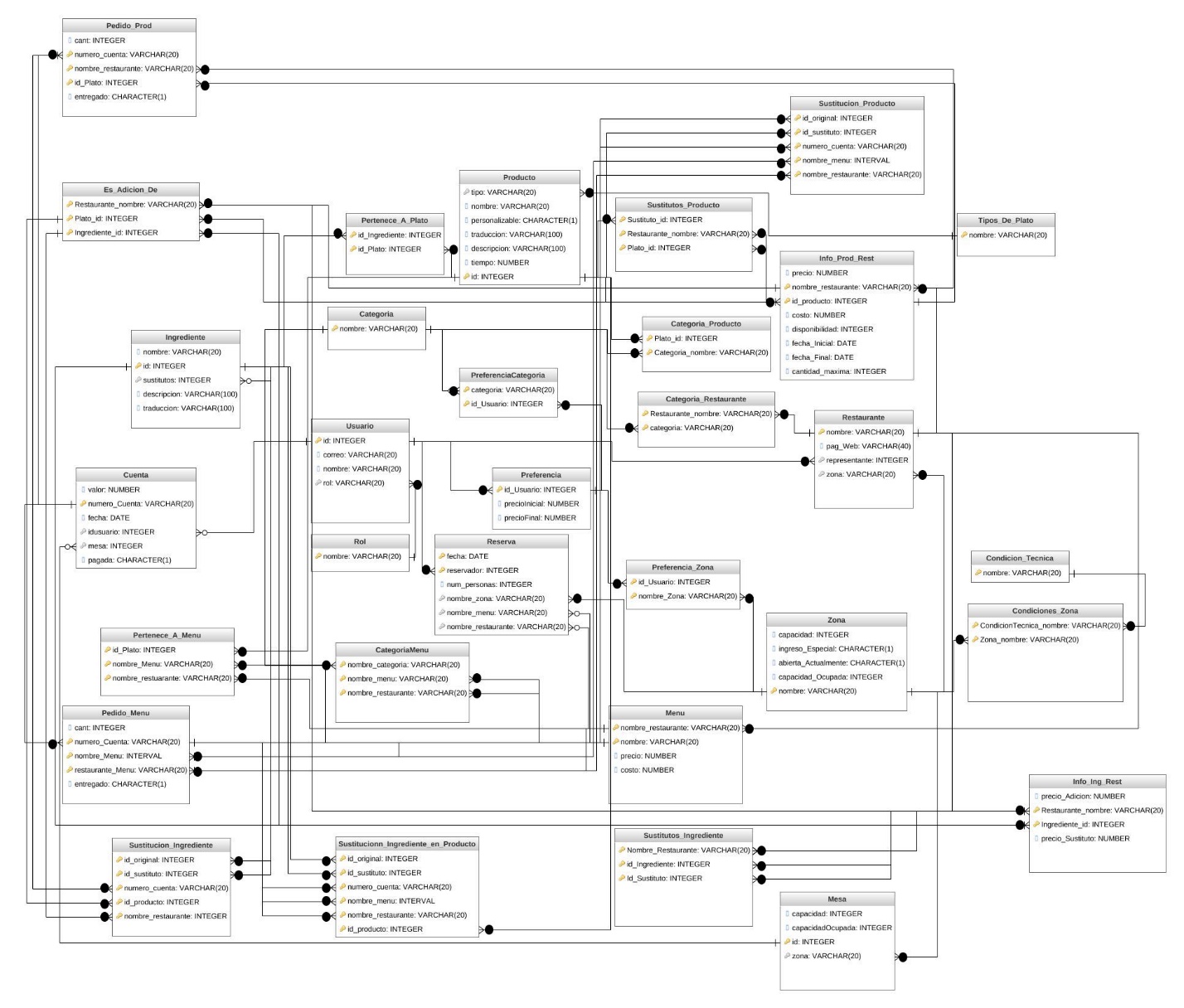


Figura 2. Modelo lógico de la iteración 4.

## Diseño físico: Justificación de índices

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NOMBRE | TABLA | COLUMNA | TIPO | JUSTIFICACION |
| IDX\_ZONA\_RESTAURANTE | RESTAURANTE | NOMBRE\_ZONA | BITMAP | Son pocas zonas, permitiendo una mayor facilidad de espacio y una búsqueda más eficiente a través de operaciones lógicas. |
|
|
|
| IDX\_USUARIO\_CUENTA | CUENTA | IDUSUARIO | B+ | Se permite una consulta más eficiente sobre los ids de usuario de las cuentas. Sirve perfecto para identificar valores diferentes de NULL. |
|
|
|
| IDX\_TIPO\_PRODUCTO | PRODUCTO | TIPO | BITMAP | Al solo haber 5 tipos fijos es facil realizar búsquedas y operaciones lógicas con el manejo de bits. |
|
|
|
| IDX\_PRECIO\_INFOPR | INFO\_PROD\_REST | PRECIO | B+ | Se permite una consulta más eficiente por rangos pequeño de precios. |
|
|
|
| IDX\_PRECIOI\_PREFERENCIA | PREFERENCIA | PRECIOINICIAL | B+ | Se permite una consulta más eficiente por rangos pequeño de precios. |
|
|
|
| IDX\_PRECIOF\_PREFERENCIA | PREFERENCIA | PRECIOFINAL | B+ | Se permite una consulta más eficiente por rangos pequeño de precios. |
|
|
|
| IDX\_PERS\_PRODUCTO | PRODUCTO | PERSONALIZABLE | BITMAP | Al haber solo dos valores, se permite una búsqueda más eficiente con búsuqedas y operaciones lógicas. |
|
|
|
| IDX\_NOMBRE\_USUARIO | USUARIO | NOMBRE | B+ | Al haber repetidos nos parece mejor tener un árbol B+ con overhead |
|
|
|
| IDX\_NOMBRE\_PRODUCTO | PRODUCTO | NOMBRE | B+ | Al haber repetidos nos parece mejor tener un árbol B+ con overhead |
|
|
|
| IDX\_MESA\_CUENTA | CUENTA | MESA | B+ | Se permite una consulta más eficiente sobre los ids de mesas de las cuentas. Sirve perfecto para identificar valores diferentes de NULL. |
|
|
|
| IDX\_FECHA\_CUENTA | CUENTA | FECHA | B+ | Se permite una consulta más eficiente por rangos pequeño de fechas para saber cuando se pagaron |
|
|
|
| IDX\_FECHAIN\_INFOPR | INFO\_PROD\_REST | FECHA\_INICIO | B+ | Se permite una consulta más eficiente por rangos pequeño de precios. |
|
|
|
| IDX\_FECHAFIN\_INFOPR | INFO\_PROD\_REST | FECHA\_FIN | B+ | Se permite una consulta más eficiente por rangos pequeño de precios. |
|
|
|

Figura 3. Tabla de índices con su justificación de selección

## Diseño físico: Índices antes de la inserción de los nuevos identificados

A continuación, se adjunta una tabla donde están los índices que se encontraban en la base de datos antes de iniciar con el proceso de creación. La versión completa de la tabla se encuentra en la carpeta docs bajo el nombre de Pks Oracle.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INDEX\_NAME | TABLE\_NAME | COLUMN\_NAME | COLUMN\_POSITION | DESCEND |
| PK\_CATEGORIA | CATEGORIA | NOMBRE | 1 | ASC |
| PK\_CATEGORIAMENU | CATEGORIA\_MENU | NOMBRE\_CATEGORIA | 1 | ASC |
| PK\_CATEGORIAMENU | CATEGORIA\_MENU | NOMBRE\_MENU | 2 | ASC |
| PK\_CATEGORIAMENU | CATEGORIA\_MENU | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_CATEGORIA\_PRODUCTO | CATEGORIA\_PRODUCTO | NOMBRE\_CATEGORIA | 1 | ASC |
| PK\_CATEGORIA\_PRODUCTO | CATEGORIA\_PRODUCTO | ID\_PRODUCTO | 2 | ASC |
| PK\_CATEGORIA\_RESTAURANTE | CATEGORIA\_RESTAURANTE | NOMBRE\_CATEGORIA | 1 | ASC |
| PK\_CATEGORIA\_RESTAURANTE | CATEGORIA\_RESTAURANTE | NOMBRE\_RESTAURANTE | 2 | ASC |
| PK\_CONDICIONTECNICA | CONDICIONTECNICA | NOMBRE | 1 | ASC |
| PK\_CONDICIONZONA | CONDICIONZONA | CONDICIONTECNICANOMBRE | 1 | ASC |
| PK\_CONDICIONZONA | CONDICIONZONA | ZONANOMBRE | 2 | ASC |
| PK\_CUENTA | CUENTA | NUMEROCUENTA | 1 | ASC |
| PK\_INFOINGREST | INFO\_ING\_REST | ID\_INGREDIENTE | 1 | ASC |
| PK\_INFOINGREST | INFO\_ING\_REST | NOMBRE\_RESTAURANTE | 2 | ASC |
| PK\_INFO\_PROD\_REST | INFO\_PROD\_REST | ID\_PRODUCTO | 1 | ASC |
| PK\_INFO\_PROD\_REST | INFO\_PROD\_REST | NOMBRE\_RESTAURANTE | 2 | ASC |
| PK\_ING | INGREDIENTE | ID | 1 | ASC |
| PK\_MENU | MENU | NOMBRE | 1 | ASC |
| PK\_MENU | MENU | NOMBRE\_RESTAURANTE | 2 | ASC |
| PK\_MESA | MESA | ID | 1 | ASC |
| PK\_PEDIDO\_MENU | PEDIDO\_MENU | NUMERO\_CUENTA | 1 | ASC |
| PK\_PEDIDO\_MENU | PEDIDO\_MENU | NOMBRE\_MENU | 2 | ASC |
| PK\_PEDIDO\_MENU | PEDIDO\_MENU | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_PEDIDO\_PROD | PEDIDO\_PROD | NUMERO\_CUENTA | 1 | ASC |
| PK\_PEDIDO\_PROD | PEDIDO\_PROD | ID\_PRODUCTO | 2 | ASC |
| PK\_PEDIDO\_PROD | PEDIDO\_PROD | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_PERTENECE\_A\_MENU | PERTENECE\_A\_MENU | NOMBRE\_MENU | 1 | ASC |
| PK\_PERTENECE\_A\_MENU | PERTENECE\_A\_MENU | ID\_PLATO | 2 | ASC |
| PK\_PERTENECE\_A\_MENU | PERTENECE\_A\_MENU | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_PERTENECE\_A\_PLATO | PERTENECE\_A\_PLATO | ID\_PLATO | 1 | ASC |
| PK\_PERTENECE\_A\_PLATO | PERTENECE\_A\_PLATO | ID\_INGREDIENTE | 2 | ASC |
| PK\_PREFERENCIA | PREFERENCIA | IDUSUARIO | 1 | ASC |
| PK\_PREFERENCIACATEGORIA | PREFERENCIACATEGORIA | IDUSUARIO | 1 | ASC |
| PK\_PREFERENCIACATEGORIA | PREFERENCIACATEGORIA | NOMBRECATEGORIA | 2 | ASC |
| PK\_PREFERENCIAZONA | PREFERENCIAZONA | IDUSUARIO | 1 | ASC |
| PK\_PREFERENCIAZONA | PREFERENCIAZONA | NOMBREZONA | 2 | ASC |
| PK\_PRODUCTO | PRODUCTO | ID | 1 | ASC |
| PK\_RESERVA | RESERVA | FECHA | 1 | ASC |
| PK\_RESERVA | RESERVA | ID\_RESERVADOR | 2 | ASC |
| PK\_RESTAURANTE | RESTAURANTE | NOMBRE | 1 | ASC |
| PK\_ROL | ROL | NOMBRE | 1 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | ID\_ORIGINAL | 1 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | ID\_SUSTITUTO | 2 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | ID\_PRODUCTO | 3 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | NUMERO\_CUENTA | 4 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | NOMBRE\_MENU | 5 | ASC |
| PK\_SUSN\_ING\_EN\_PRODUCTO | SUSN\_INGREDIENTE\_EN\_PRODUCTO | NOMBRE\_RESTAURANTE | 6 | ASC |
| PK\_SUSTITUCION\_INGREDIENTE | SUSTITUCION\_INGREDIENTE | ID\_ORIGINAL | 1 | ASC |
| PK\_SUSTITUCION\_INGREDIENTE | SUSTITUCION\_INGREDIENTE | ID\_SUSTITUTO | 2 | ASC |
| PK\_SUSTITUCION\_INGREDIENTE | SUSTITUCION\_INGREDIENTE | NUMERO\_CUENTA | 3 | ASC |
| PK\_SUSTITUCION\_INGREDIENTE | SUSTITUCION\_INGREDIENTE | ID\_PRODUCTO | 4 | ASC |
| PK\_SUSTITUCION\_INGREDIENTE | SUSTITUCION\_INGREDIENTE | NOMBRE\_RESTAURANTE | 5 | ASC |
| PK\_SUSTITUCION\_PRODUCTO | SUSTITUCION\_PRODUCTO | ID\_ORIGINAL | 1 | ASC |
| PK\_SUSTITUCION\_PRODUCTO | SUSTITUCION\_PRODUCTO | ID\_SUSTITUTO | 2 | ASC |
| PK\_SUSTITUCION\_PRODUCTO | SUSTITUCION\_PRODUCTO | NUMERO\_CUENTA | 3 | ASC |
| PK\_SUSTITUCION\_PRODUCTO | SUSTITUCION\_PRODUCTO | NOMBRE\_MENU | 4 | ASC |
| PK\_SUSTITUCION\_PRODUCTO | SUSTITUCION\_PRODUCTO | NOMBRE\_RESTAURANTE | 5 | ASC |
| PK\_SUSTITUTOS\_INGREDIENTE | SUSTITUTOS\_INGREDIENTE | ID\_INGREDIENTE | 1 | ASC |
| PK\_SUSTITUTOS\_INGREDIENTE | SUSTITUTOS\_INGREDIENTE | ID\_SUSTITUTO | 2 | ASC |
| PK\_SUSTITUTOS\_INGREDIENTE | SUSTITUTOS\_INGREDIENTE | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_SUSTITUTOS\_PRODUCTO | SUSTITUTOS\_PRODUCTO | ID\_PRODUCTO | 1 | ASC |
| PK\_SUSTITUTOS\_PRODUCTO | SUSTITUTOS\_PRODUCTO | ID\_SUSTITUTO | 2 | ASC |
| PK\_SUSTITUTOS\_PRODUCTO | SUSTITUTOS\_PRODUCTO | NOMBRE\_RESTAURANTE | 3 | ASC |
| PK\_TIPOSP | TIPOSDEPLATO | NOMBRE | 1 | ASC |
| PK\_USUARIO | USUARIO | ID | 1 | ASC |
| PK\_ZONA | ZONA | NOMBRE | 1 | ASC |

Figura 4. Tabla del estado de los índices de la base de datos antes de la creación de los nuevos identificados

Por lo que se puede evidenciar, *Oracle* creó estos índices primarios (o *clustered*) debido a que son llaves primarias de las tablas a las que pertenecen, permitiendo tener una búsqueda más eficiente en los requerimientos funcionales de la aplicación sobre un conjunto ordenado de datos que se identifican con un valor único en toda la tabla. Partiendo de acá, y después de analizar los requerimientos funcionales de consulta que se han creado a lo largo de estas 4 iteraciones, se afirma que los índices más útiles de los ya creados son:

* PK\_USUARIO: Permite la búsqueda rápida de un Usuario.
* PK\_ZONA: Permite la búsqueda rápida de una Zona.
* PK\_RESTAURANTE: Permite la búsqueda rápida del nombre de un Restaurante.
* PK\_PERTENECE\_A\_MENU: Permite saber si un Producto pertenece a un Menú de forma rápida.
* PK\_PEDIDO\_PROD: Interesa saber el número de la cuenta que ya se encuentra en el orden necesario. A su vez se reuiere que sus otras dos llaves estén en el orden de las llaves primarias de la tabla INFO\_PROD\_REST, lo cual ya está.
* PK\_PEDIDO\_MENU: Interesa saber el número de la cuenta que ya se encuentra en el orden necesario. A su vez se reuiere que sus otras dos llaves estén en el orden de las llaves primarias de la tabla MENU, lo cual ya está.
* PK\_PRODUCTO: Interesa saber el id del producto para unirlo rápidamente con cualquier tabla que lo tenga involucrado.
* PK\_PREFERENCIA: Interesa saber la identificación del Usuario en la preferencia para poder acceder a esta rápidamente.
* PK\_PREFERENCIAZONA: Interesa saber la identificación del Usuario en la preferencia para poder acceder a esta rápidamente.
* PK\_PREFERENCIACATEGORIA: Interesa saber la identificación del Usuario en la preferencia para poder acceder a esta rápidamente.
* PK\_MESA: Interesa saber la identificación de las mesas para vincularlas rápidamente a las consultas involucradas.
* PK\_MENU: Interesa que esté presentando primero el nombre del menú y luego el del restaurante (para que se conecte directamente con el índice planteado en PK\_PEDIDO\_MENU).
* PK\_INFO\_PROD\_REST: Interesa que esté presentando primero el nombre del menú y luego el del restaurante (para que se conecte directamente con el índice planteado en PK\_PEDIDO\_PROD).

## Documentación de la lógica de requerimientos nuevos y transaccionales

# Construcción de la aplicación

En esta sección se presenta una breve descripción de cómo se realizaron los requerimientos introducidos en esta nueva iteración. Las pruebas de los mismos se encuentran en colecciones de *Postman* como se sugiere en [1], junto con casos de éxito y fallo en un archivo *Excel*.

## RF9

Para este requerimiento, cada vez que se agregaba un producto o un menú al pedido se verificaba la disponibilidad y se actualizaba el precio de la cuenta dependiendo del valor y cantidad dados.

## RF10

Para este requerimiento se tuvo la decisión de coger cada producto y menú encontrado en la cuenta y verificar si era posible pagarlo o no con la disponibilidad actual del restaurante; si lo anterior fallaba de alguna manera se guardaba en una lista todos los artículos comprados que no se tuvieran en el momento y se le mostraba al usuario el listado respectivo. Con este listado se puede decidir actualizar el pedido, dejar todo cómo está o , cuando se ofrezca la posibilidad de utilizar sustitutos, sustituir el producto dado por alguna de sus posibles opciones.

## RF11

Para este requerimiento se permitió añadir las equivalencias de dos maneras: o mediante un PUT a la tabla de InfoIngRest el cual permiten modificar la lista de sustituciones que está guardada en ella, o mediante un POST en el cual se pueden asociar dos ingredientes, conociendo sus identificadores. Borrar la sustitución solo se permitió a través del PUT.

## RF12

Para este requerimiento se permitió añadir las equivalencias de dos maneras: o mediante un PUT a la tabla de InfoProdRest el cual permiten modificar la lista de sustituciones que está guardada en ella, o mediante un POST en el cual se pueden asociar dos productos, conociendo sus identificadores. Borrar la sustitución solo se permitió a través del PUT.

## RF13

Se pretende que cuando se actualice un restaurante se indique si se quiere surtir de nuevo los productos o no. En caso tal de que se desee hacer esto, se validad que solo lo pueda hacer un usuario dueño del restaurante objetivo. Al final, cuando se obtiene la información del restaurante se puede ver la reacomodación de datos de forma clara.

## RF14

Este requerimiento se dividió en dos servicios REST diferentes, uno para ordenar productos, y otro para ordenar menús. Al ordenar el producto toca añadir en una lista las sustituciones deseadas, al ordenar el menú se puede sustituir un producto de él por otro, o sustituir un ingrediente dentro de alguno de los productos que componen el menú. Ambas cosas pueden hacerse añadiendo la lista de sustituciones al cuerpo de la petición.

## RF15

Este requerimiento hace uso del requerimiento RF9 para ordenar varios pedidos. Todos deben pertenecer a la misma mesa, aunque pueden ser de diferentes cuentas. El ordenar los pedidos se hace de manera atómica, es decir, o se hacen todos los pedidos o ninguno. Cabe aclarar que esto también se hace a partir de dos servicios, uno para ordenar muchos productos, y otro para ordenar muchos menús.

## RF16

En este requerimiento hace uso del requerimiento RF10 para tomar el servicio de todas las cuentas de una mesa, lo cual se hace de manera atómica, si no hay disponibilidad de algún pedido no se lleva a cabo el servicio de ninguno.

## RF17

En general, para este requerimiento se usó la opción de borrar para las clases PedidoProd y PedidoMenu por lo que si uno es el restaurante, solo puede borrarlos del sistema antes de haberlos entregado. En el caso de una mesa se permite que el usuario de tipo Restaurante intente eliminar todas los pedidos que pertenecen únicamente a su local dentro de la misma mesa.

## RFC7

Para este requerimiento se sigue la ruta "[http://localhost:8080/VideoAndes/rest/usuarios /productos"](http://localhost:8080/VideoAndes/rest/usuarios%20/productos%22) , además se crearon las siguientes clases en el paquete rfc:

* ContenedoraClienteProducto: Muestra la información de un cliente (usando su identificación), dicha información se encuentra en la clase ContenedoraMesaMenuCliente.
* ContenedoraMesaMenuCliente: Representa a través de un booleano si el menú tiene o no una mesa asignada, y después muestra un listado de información expresado en la clase ContenedoraMenuCliente.
* ContenedoraMenuCliente: Tiene el nombre del restaurante, el nombre del menú y la información relevante del producto. En caso tal de que se haya consumido el producto por fuera del menú, el nombre del menú no existe. La información del producto se expresa en la clase ProductoInformativo.
* ProductoInformativo: Solo tiene un nombre, una descripción, tipo, traducción e identificación.

En esencia, dependiendo del usuario que lo solicite, se muestra la información de un cliente(el que realiza la consulta) o de todos los clientes del sistema (si el usuario que lo solicita es un administrador).

## RFC8

Para este requerimiento se sigue la ruta "[http://localhost:8080/VideoAndes/rest/ restaurantes/informacion-financiera"](http://localhost:8080/VideoAndes/rest/%20restaurantes/informacion-financiera%22), además se crearon las siguientes clases en el paquete rfc:

* ContenedoraRestauranteInfoFinanciera: Contiene la información financiera de un restaurante particular.
* InformacionFinanciera: Es la información financiera de un restaurante. Contiene el producto al que hace referencia (Id de un plato o nombre del menú), la cantidad que se ha vendido a cuentas con o sin cliente, y el total de ventas en ambos casos.

Dependiendo de la identificación del usuario que llega como parámetro se escoge si mostrar toda la información de todos los restaurantes, o si la de uno solo (el restaurante del que el usuario sería dueño). Esta información se almacena en las clases ya descritas.

## RNF3

Para probar el mantenimiento de la concurrencia, además de tener el uso de commit y rollback, y de usar un nivel de aislamiento READ COMMITTED, se realizó una carpeta de pruebas en el archivo *Postman* llamado “PruebasConcurrencia”. En este se encuentran tres carpetas:

* Misma sesión: Se intenta probar la concurrencia de dos pedidos simulados en una sola sesión donde se agota la cantidad de uno de los productos, y cuando se va a hacer otro pedido sobre el mismo ítem se genera un error.
* Sesión 1: Carpeta que se corre antes de la carpeta “Sesión 2”, y es usada para surtir el restaurante llamado “Skinte”. Se supone que después de haberse surtido el restaurante, se inicia la prueba de la carpeta “Sesión 2”, y uno de los registros de pedido con valor “3000” en alguna de las dos sesiones debe fallar debido a la falta de disponibilidad.
* Sesión 2: Pruebas que se corren después de haberse surtido el restaurante “Skinte” en la carpeta “Sesión 1”.

# Consideraciones adicionales

Dentro de la carpeta data se tienen las siguientes consideraciones:

* Se tienen los archivos de pruebas de *Postman* en la carpeta pruebas. Antes de ejecutarlas se debe correr el archivo PrePostman.sql.
* En el archivo BorradoEInserciones.sql se tienen los comandos para borrar todos los datos existentes en la base de datos y volver a poblarla en caso tal de que se corrompan de laguna manera los datos del sistema.
* En el archivo DBSETUPFULL.sql se tienen los esquemas de las tablas para reconstruir la base de datos dado el caso en que haya la necesidad de hacerlo.
* En la carpeta inicializacion están todas las inserciones en tablas utilizadas en la población de la base de datos. Además, las tablas de equivalencias se encuentran en el archivo Prepostman.sql.
* La ruta se sigue llamando “VideoAndes” de acuerdo con el ejemplo de clase, dado que no se encontró la forma de cambiar el nombre a “RotondAndes” como se pretendía.

# Análisis de resultados

## Aprendizajes y logros

A lo largo del desarrollo de la iteración 3 se aprendió a refinar un sistema transaccional básico con el uso de las propiedades ACID.

## Conclusiones

* Se debe saber determinar las operaciones transaccionales del sistema porque hay varias que requieren ser o todo o nada.
* SQL es una herramienta poderosa que permite realizar consultas complejas sobre modelos de datos gigantes.
* La interconexión entre clases permite la funcionalidad completa de un sistema transaccional dado.
* Toda decisión de diseño se debe reflejar en la base de datos planteada.
* Las propiedades ACID ayudan a determinar el manejo sobre la concurrencia de la aplicación, permitiendo definir diferentes niveles de aislamiento, dependiendo de las necesidades del programador. Para esta iteración se usó el modo READ COMMITED al ver que solo se debía garantizar que las transacciones pudieran ver quien había realizado o no un commit durante las diferentes transacciones de la aplicación.

# Bibliografía

1. Universidad de los Andes. *Caso de estudio - v3.* [En línea] Universidad de los Andes. [Citado el: 4 de Octubre de 2017.] https://sicuaplus.uniandes.edu.co/bbcswebdav/pid-1918605-dt-content-rid-20597579\_1/courses/UN\_201720\_ISIS2304/isis2304-172-CasoEstudio-RotondAndes-v3%281%29.pdf