**Iteración 4:**

María Paula Franco Guzmán, Camilo Andrés Sánchez Salamanca

Iteración

Universidad de los Andes, Bogotá, Colombia

{mp.franco10, ca.sanchez38}@uniandes.edu.co

Fecha de presentación: Noviembre 21 de 2017

**Tabla de contenido**

[1 Introducción 1](#_Toc497062574)

[2 Etapa 1: Análisis 1](#_Toc497062575)

[3 Etapa 2: Diseño de la aplicación 2](#_Toc497062576)

[4 Etapa 3: Construcción de la aplicación 6](#_Toc497062577)

[5 Bibliografía 6](#_Toc497062578)

# Introducción

En el ámbito de los restaurantes en grandes centros comerciales, los sistemas transaccionales son componentes importantes para realizar las operaciones más importantes de la organización e incrementar la productividad de estas. Estos permiten controlar y administrar múltiples transacciones manteniendo seguridad, persistencia y consistencia de los datos manejados. En el caso de “RotondAndes”, empresa que actúa como intermediaria entre restaurantes y usuarios finales, los sistemas transaccionales permiten a los usuarios consultar, comprar y pagar servicios de alimentación de acuerdo a sus intereses y a las restricciones de los proveedores.

Para realizar una implementación correcta de un sistema transaccional es necesario realizar procesos de diseño y análisis del sistema. El proceso inicial de diseño consiste en identificar los requerimientos funcionales del sistema, conocer las restricciones o condiciones de este y realizar modelos conceptuales que permitan representar el sistema e identificar entidades y relaciones entre estas. Posteriormente, se deben realizar modelos de datos y relaciones a partir de los modelos conceptuales. Se debe realizar la implementación del diseño realizado en un sistema transaccional y las características de este que le permiten tener las propiedades A.C.I.D. En el presente documento se proponen las modificaciones a los modelos conceptuales de la aplicación utilizada por la organización “RotondAndes” así como a los modelos de datos y relacionales de la aplicación.

Para finalizar es necesario asegurarse de haber realizado una implementación optima, no únicamente optima respecto a las operaciones en la sentencia, si no también con el aprovechamiento de índices en las tablas consultadas. Finalmente se presentan los resultados de la implementación de la aplicación transaccional y se documentan los requerimientos pedidos.

# Etapa 1: Análisis

A nivel de modelo conceptual no se realzo ninguna modificación ya que los objetos que representan las diferentes cosas que pertenecen a la rotonda no tienen ningún cambio, así mismo no se agrega ni se elimina ninguna funcionalidad. Por tanto, el UML no cambia en ningún momento y se mantiene con respecto a la iteración pasada.

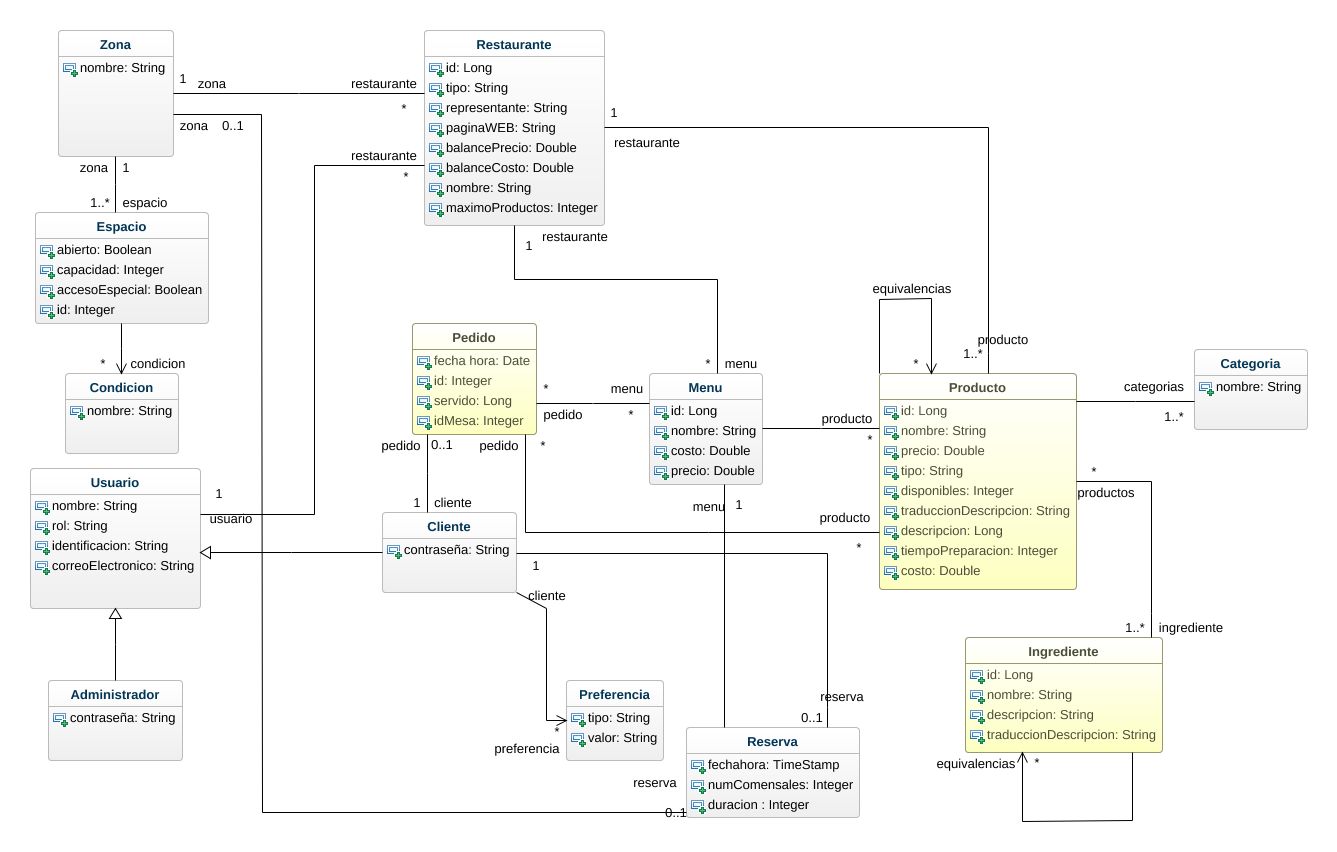


Figura Modelo del mundo modificado

A nivel del modelo relacional, tampoco se realiza ningún cambio esto debido a que las relaciones se mantienen iguales, y al no haber cambio en el modelo conceptual tampoco se hace necesario agregar ninguna tabla nueva. Lo único que podría decirse que cambia respecto a este modelo es el uso de índices en las tablas para hacer que las consultas sean mucho más optimas, sin embargo, es un tema que se explicara más adelante y aunque se relaciona con el modelo relacional no hace parte de este.

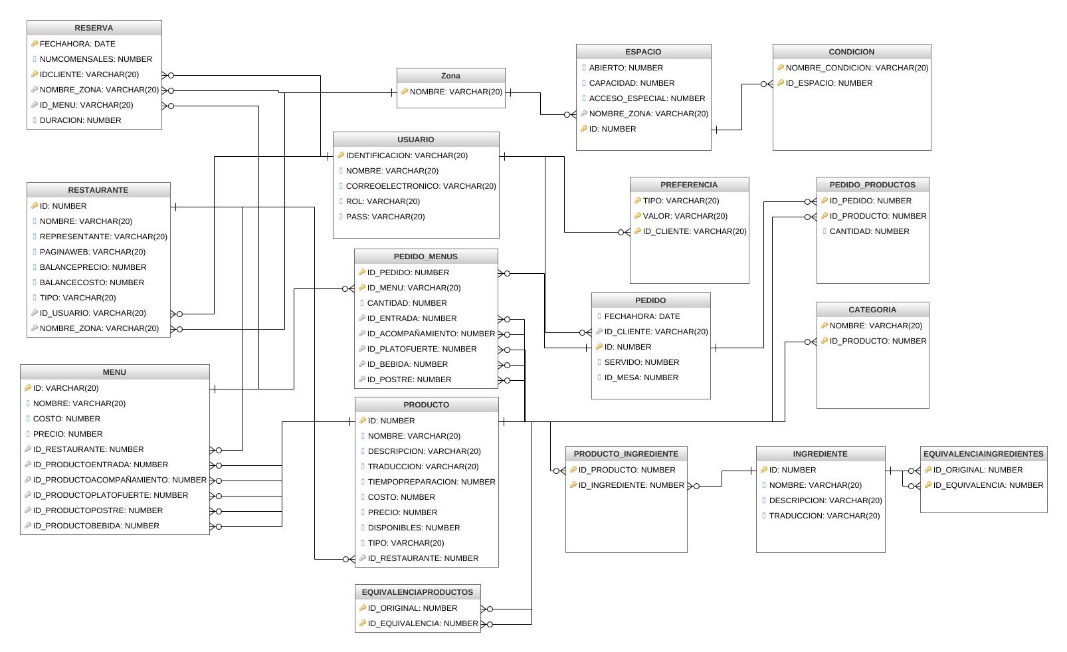


Figura Modelo relacional

En la iteración anterior las tablas se habían diseñado en BCNF, por lo que no se realizaron cambios de calidad. La anormalidad que hay es la de contener los productos del menú en el pedido, pero esto es debido a que el restaurante necesita conocer qué es lo que el cliente quiere pedir. Mediante transacciones se manejan estas columnas.

# Etapa 1: Diseño de la aplicación

Para esta iteración como ya se mencionó antes no se realizó ninguna modificación en las tablas de la base de datos, ya que la idea principal del trabajo era optimizar el trabajo sobre estas tablas, mas no agregar cosas nuevas como tal, sin embargo, hubiese sido útil crear algunas columnas nuevas con datos específicos, para ahorrar el uso de algunas funciones, como por ejemplo para fecha ya que se hace inútil crear un índex de fecha ya que se ejecuta una función sobre este valor para poder hacer el filtro.

Se crearon diferentes en las tablas para optimizar cada consulta.

**Consulta 1:**

**Se colocaron índices sobre las tablas de relación de pedidos y productos, tanto sobre su id de producto y su id de restaurante, los cuales no funcionaron debido a que el plan de trabajo ya usaba las llaves primarias de estas tablas para realizar las búsquedas, por tanto, ya se está usando un índex en estas. A su vez se realizó un índice compuesto sobre la tabla pedido con su fecha y el id del cliente, sin embargo, al igual que los índices nombrados anteriormente no fueron de utilidad, ya que el análisis de la fecha se hace con una función por tanto el índex no puede ser utilizado ya que terminan revisándose todas las tuplas de la tabla.**

**Consulta 2:**

**Ya que esta consulta es la negación de la anterior, esta se basa por completo en la anterior, solo que buscando aquellos usuarios que no pertenecen a esta tabla. Por tanto se hace imposible el uso de un índice para optimizarla, o al menos debido al plan de trabajo que se usa el cual ya esta usando llaves primarias en todas sus búsquedas, es completamente inútil crear mas índices, y el único índice compuesto es inútil también debido a la forma en la que se analiza el rango de fecha.**

**Consulta 3:**

**Consulta 4:**



Tabla Información del modelo relacional

En la tabla se ven todas las columnas de cada tabla, el tipo de dato de las columnas, el nombre de la PK a la que pertenece una columna. Si no pertenece a ninguna aparece vacío (en sqldeveloper aparece null). También se puede ver el nombre del check y el nombre de la llave foránea.

Para garantizar las propiedades ACID de cada requerimiento se toman por defecto los valores de aislamiento y lectura de Oracle, ya que se tiene el “READ COMMITED” por defecto, que solo lee cambios que se han confirmado, lo cual brinda datos consistentes, pues ninguna transacción hace cambios sin el autocommit. Adicionalmente, al realizarse validaciones en el mundo del problema se evitan inconsistencias. Además como las tablas están en BCNF no se presentan muchos problemas al realizar una utilización concurrente.

# Etapa 2: Construcción de la aplicación

Para las pruebas los datos se generaron manualmente. No hay una gran cantidad de datos, pero hay suficientes datos para revisar los requerimientos y son consistentes entre ellos. En la carpeta docs está el archivo sql DB.sql que se ejecuta para reiniciar la base de datos. No se ejecuta directamente desde eclipse sino que se copia y se pega a sqldeveloper, y luego este se ejecuta como script.

A continuación, se presentan las pruebas (escenarios de prueba), especificaciones y demás datos sobre cada requerimiento. Se presentan los datos requeridos en este documento y no en un Excel para tener más orden.

**RF1 – Registrar Usuario**

En este requerimiento simplemente se agrega un usuario a RotondAndes con una sentencia de insert.

URL: localhost:8080/RotondAndes/rest/usuarios

* Registrar nuevo usuario

JSON:

{

"rol": "No registrado",

"name": "Andres",

"identificacion": "40",

"contraseña": null,

"correoEletronico": "andresca2@hotmail.com"

}

* Registrar un usuario existente

JSON:

{

"rol": "No registrado",

"name": "Andres",

"identificacion": "40",

"contraseña": null,

"correoEletronico": "andresca2@hotmail.com"

}

Con el GET puede obtener todos los usuarios en RotondAndes.

# Bibliografía

|  |  |
| --- | --- |
| [1] | Oracle, «Oracle Docs,» Oracle Help Center , 2017. [En línea]. Available: https://docs.oracle.com/en/. |
| [2] | W3Schools, «SQL Tutorial,» W3Schools, 2017. [En línea]. Available: https://www.w3schools.com/sql/default.asp |