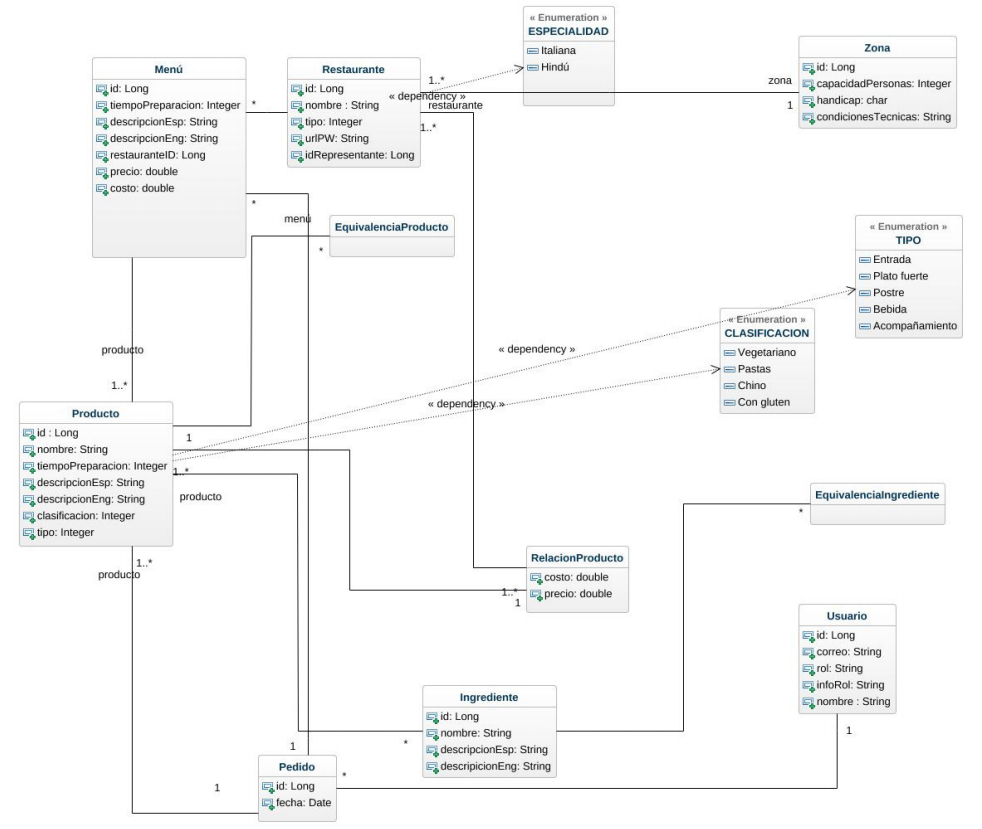
**Documentación Iteración 3**

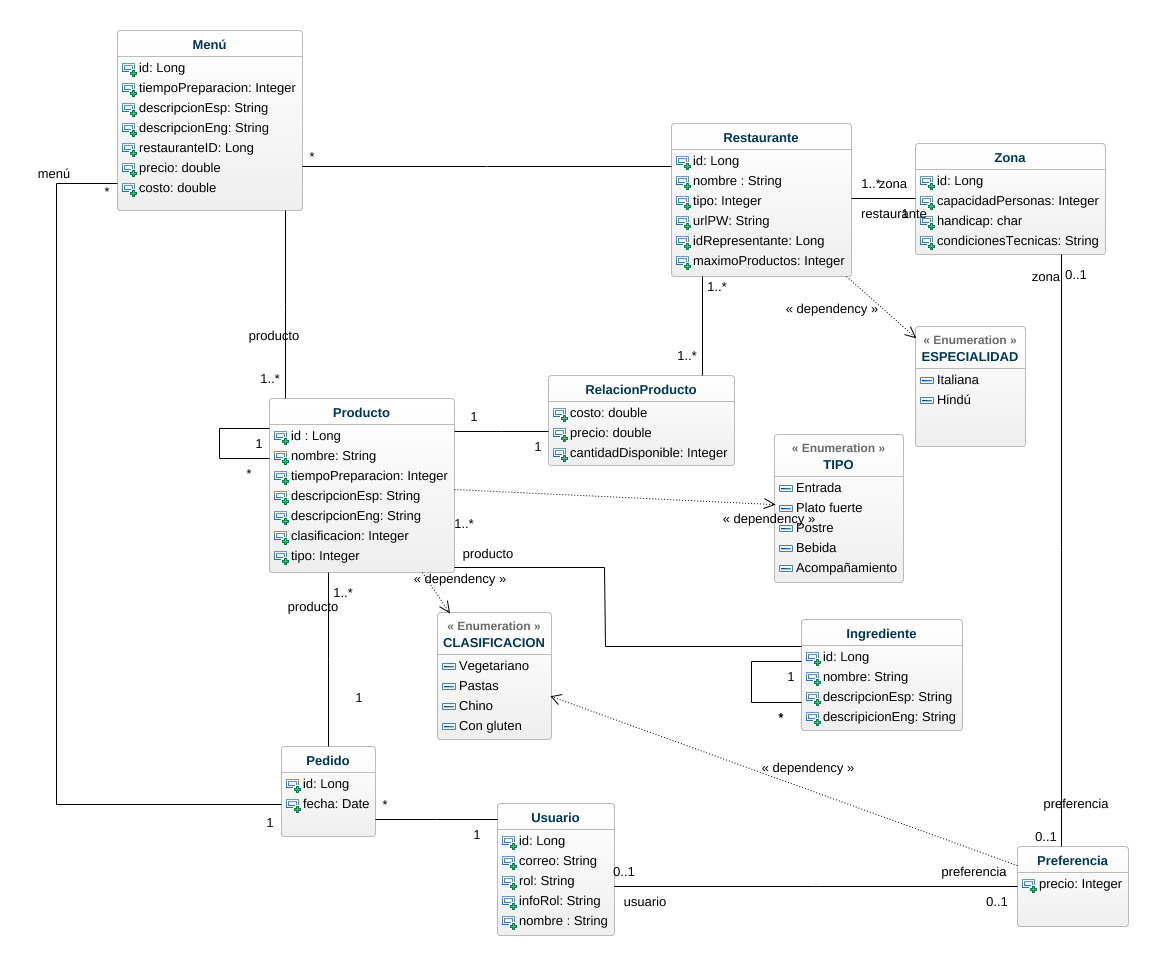
**Modelo anterior:**



**Cambios:**

* Se agrega un nuevo atributo a la clase de restaurante que modela el máximo de productos que tiene disponibles para la venta.
* Se eliminan las clases de equivalenciaProducto y equivalenciaIngrediente para introducir en las clases de Producto e Ingrediente relaciones hacia sí mismas.
* Se añade un nuevo atributo a la clase de RelacionProducto que modela la cantidad de productos disponibles de ese restaurante específico.
* Se añade la clase Preferencia para modelar la preferencia de un usuario respecto a algunos criterios estándar.

**Nuevo modelo:**



**Punto 2:**

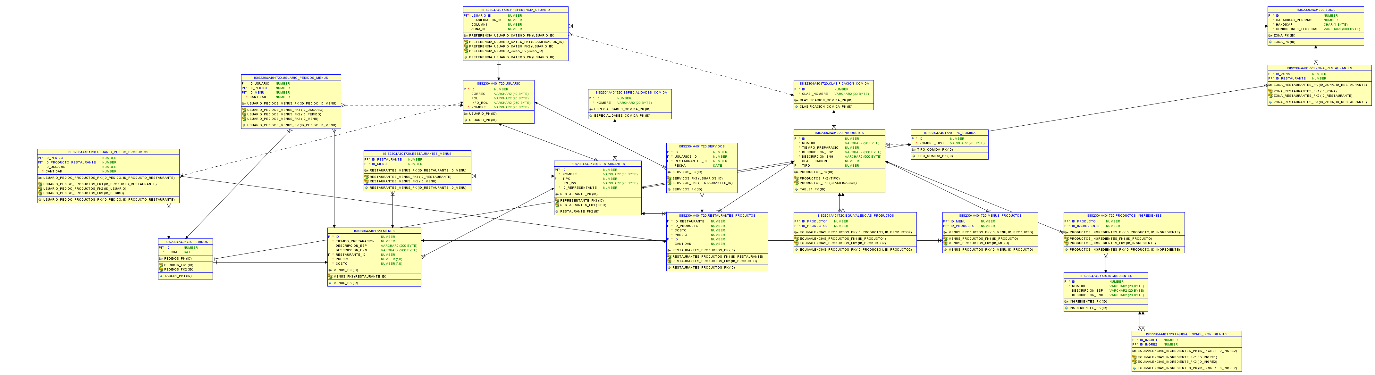
**Modelo anterior:**

****

**Cambios:**

* Ahora las preferencias se manejan en una sola tabla.
* Se borra la tabla servidos y se maneja como un estado de pedido
* Además, se organizó un poco el modelo porque en el anterior teníamos un problema con las relaciones y toco crear el modelo desde sql developer para manejar los criterios acordados por las reglas de entregas de trabajos.
* Se valido que la relación de producto restaurante tuviera la cantidad agregando esta columna.

**Nuevo modelo:**



El modelo relacional que presentamos sobre la posible solución del caso de estudio cumple con todas las normas de formalización:

**Análisis de formalidad**

**FN1:** el modelo relacional claramente se encuentra en forma normal 1, y esto se puede probar porque todos los valores de los atributos de las tabas corresponden a valores atómicos, es decir, que solo poseen un valor y no son una tupla o algo por el estilo.

**FN2:** como el modelo respeta la FN1 y todos los atributos que no son llaves primarias dependen de las llaves primarias y no existen relaciones parciales.

**FN3:** como el modelo respeta la FN2 y claramente se muestra que no existen dependencias entre os atributos que no son llaves, por ejemplo, el tipo de producto no depende de su nombre.

**Forma normal de Boyce-Codd (FNBC):** como el modelo respeta la FN3 y no existen relaciones entre primos, llaves que dependan de otras, las tablas generadas están correctamente formadas. Esto significa que un atributo que determina completamente a otro es clave candidata.

**Nota:** el listado de tablas se encuentra en un archivo Excel en la carpeta Docs, la sentencia usada para obtener toda la información fue la siguiente:

**SELECT \* FROM USER\_TAB\_COLS INNER JOIN ALL\_CONSTRAINTS**

**ON USER\_TAB\_COLS.TABLE\_NAME = ALL\_CONSTRAINTS.TABLE\_NAME;**

**Parte C: Explicación de los requerimientos.**

**RF11:**

Para este requerimiento se hace un insert normalmente de los ids de los ingredientes sin problema pues no tiene restricción de transaccionalidad. antes de ejecutarlo se evalúa que sea un restaurante e que lo hace.

**RF12:**

Para este requerimiento se hace un insert normalmente de los ids de los productos validando previamente con un if si son de la misma clasificación y si lo cumple lo registra sin problema pues no tiene restricción de transaccionalidad. antes de ejecutarlo se evalúa que sea un restaurante e que lo hace.

**RF13:**

En este método se hace un update de la información de la cantidad en la tabla relación haciendo la condición que la relación cumpla con el id de producto e id de restaurante para que se haga la calidad y antes de ejecutarlo se evalúa que sea un restaurante e que lo hace.

**RF15:**

En este requerimiento lo que se hace es crear los pedidos si no existe antes, se registra la información se recorre un ciclo para registrar los pedidos de todos los comensales de una mesa. Si no hay la cantidad de alguno ese pedido no se regista y se hace rollback para no crear ese.

**RF16:**

En este requerimiento se recorre el arreglo de pedidos que entra como parámetro apra cambiar su estado a servido.

**RF17:**

En este requerimiento se recorre la lista de productos que entra como parámetro se cambia la cantidad de unidades que había antes.