# RETO TECNICO FINAL – BASES DE DATOS


Realizado por

**LAURA VANESSA ALBA GONZALEZ**


Dirigido a

**JUAN PINEDA**


**SOFKA U**

**TRAINING QA**

**FUSAGASUGA - CUNDINAMARCA**

**2023**

# Contenido

## Requisitos de la actividad

- Indicar que ejercicio fue asignado
- Realizar el modelo E-R
- Realizar el modelo relacional
- Normalizar correctamente
- Escribir con sentencias SQL toda la definición de la base de datos.
- Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).
- Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).
- Generar al menos 4 procedimientos almacenados.
- Generar al menos 4 triggers
- Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.
- Al terminar el ejercicio responda ¿Está conforme con el resultado obtenido según el contexto o cree que hubiera obtenido un mejor resultado con una base de datos no relacional?
- documente muy bien su proceso (paso a paso) en un archivo PDF escriba todas las aclaraciones o especificaciones necesarias para realizar el ejercicio.

## Ejercicio asignado

**Tienda Virtual Don pepe (Ejercicio C)**

Don pepe quiere que sus clientes puedan realizar compras desde sus casas. El junto a su esposa tienen una cantidad domiciliarios conocidos que se encargan de llevar los pedidos a los clientes.

A continuación, se muestra la conversación que se tuvo con don pepe:

- ¡Veee mijo! yo quiero que mas gente me compre los productos, cuando llegá un vecino nuevo a la cuadra yo lo apunto en un cuadernito. Entiendo don pepe, y no le gustaría que le comprarán por internet?
- Ehhh mijo pues no es mala idea y que hago con mi clientela?
- Pues don pepe hacemos un video tutorial para usar la aplicación, y le pedimos una información a sus clientes indicando sus datos personales (ID, cedula, Nombre, Dirección, Teléfono, email y password) a través de un formulario de registro. Una vez registrado podrá acceder a la realización de pedidos con su email y su password.
- ¡ eeeee yo no te creo! Asi de fácil? Como motilando calvos?
- Don pepe ojala fuera así de sencillo dejeme le cuento mejor, Los productos que oferta el supermercado deben estar divididos en diversas categorías. Los datos necesarios para cada categoría son: nombre de la categoría, condiciones de almacenamiento (frío, congelado, seco) y observaciones. Tambien debemos detallar la información de

los productos (nombre, marca, origen, dimensiones (volumen y peso), una fotografía, la categoría y unidades disponibles).¡ no mijo eso me va salir muy caro con tanto detalle!

- don pepe todo lo contrario va aumentar mucho sus ganancias espereme le cuento algo mas, la aplicación permitirá visualizar un listado de productos ordenado por categoría, permitiendo seleccionar los productos que desee comprar mediante una caja de texto donde se indicará el número de unidades seleccionadas. La aplicación llevará la cuenta (cesta de la compra) de los productos que el cliente ha ido seleccionando. La aplicación permitirá también efectuar un pedido con todos los productos que lleve almacenados en su cesta de la compra. Los datos del pedido son: código del pedido, fecha del pedido, cliente, dirección de entrega, productos pedidos, importe total del pedido y datos de pago (número de tarjeta y fecha de caducidad)®.

Para poder generar un pedido se deberán dar dos situaciones:

- El cliente deberá pertenecer a una zona (Código Postal) donde existan domiciliarios. Un domiciliario se identifica mediante un nombre, número de matrícula de la furgoneta y zona donde reparte.
- Debe haber unidades suficientes por cada producto para satisfacer las demandas de cada pedido.

Una vez generado el pedido se mostrará al usuario una página con los datos de su pedido, se restarán del stock las unidades pedidas y se emitirá una nota de entrega a los responsables de almacén para que sirvan ese pedido.

# Modelo entidad relación (MER)

## Entidades

Después de haber leído y analizado el problema planteado se extraen las entidades que se lograron evidenciar

- Domiciliario
- Cliente
- Producto
- Categoría
- Carrito de compra
- Zona
- Pedido
- Factura
- Proveedor
- Inventario

## Atributos

Luego de haber sacado las entidades continuamos a definirle los atributos a cada entidad acompañado de su clave primaria

- Domiciliario: **cedula** , nombre, numero de matricula.
- Cliente: **cedula**, nombre, dirección, teléfono, correo, contraseña.

- Producto: **nombre,** marca, dimensión, fotos, precio.
- Categoría: **id**, nombre, almacenamiento, observaciones.
- Carrito de compra: **id,** cantidad**.**
- Zona : **código postal**, nombre.
- Pedido: **código**, fecha, datos de pago.
- Factura: **código**, total.
- Proveedor: **razón social**, teléfono, correo
- Inventario: **id**, cantidad disponible

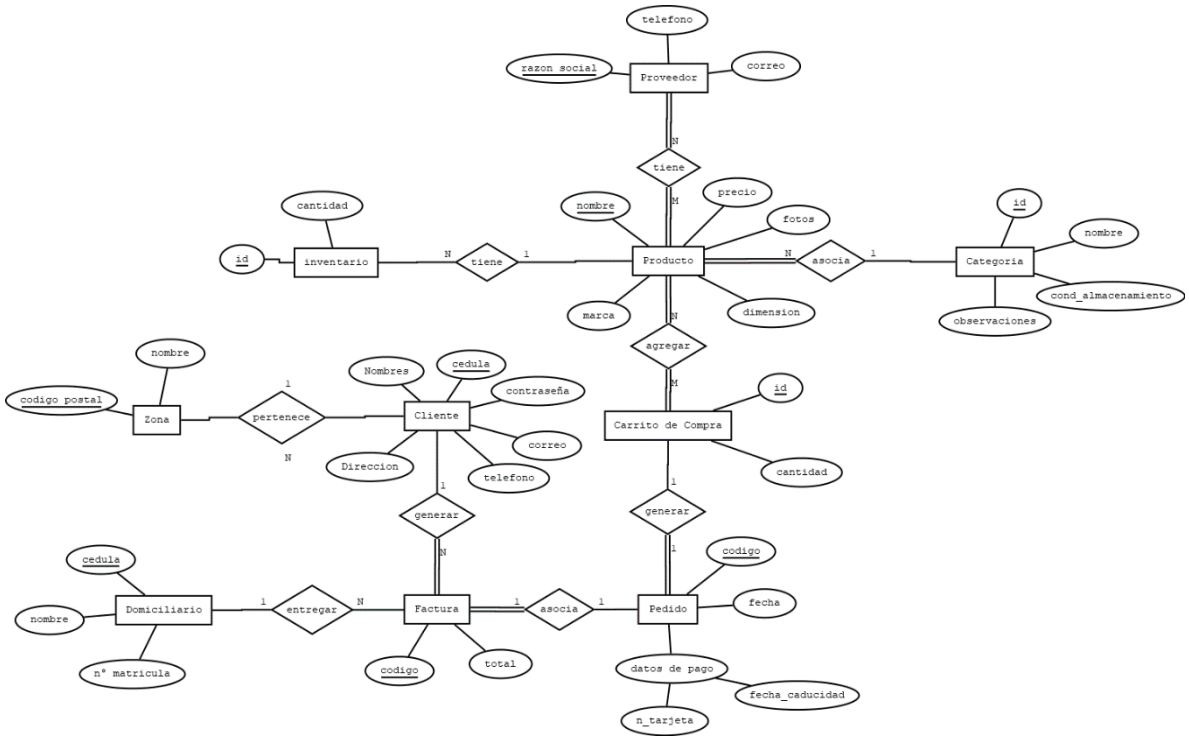
## Relaciones

Ya con la anterior información se empiezan a sacar las relaciones entre entidades

- Un domiciliario puede entregar una o muchas facturas, una factura puede ser entregada por solo un domiciliario (1:N)
- Un cliente puede generar una o muchas facturas, una factura puede ser generada solo por un cliente (1:N)
- En una zona pueden pertenecer uno o muchos clientes, un cliente puede pertenecer a una sola zona (1:N)
- Una factura puede estar asociada a un solo pedido, un pedido puede estar asociado a solo una factura (1:1)
- Un carrito de compra puede generar un pedido, un pedido puede ser generado solo por ese carrito de compra (1:1)
- Un producto puede ser agregado a uno o muchos carritos de compra, un carrito de compra puede agregar uno o muchos productos (N:M)
- Un producto puede estar asociada a una categoría, una categoría se le pueden asociar uno o muchos productos (1:N)
- Un producto puede tener uno o muchos proveedores, un proveedor puede tener uno o muchos productos (N:M)
- Un producto puede tener una o muchas unidades en el inventario, cada unidad del inventario solo corresponde a un producto (N:1)

## Diagrama Modelo Entidad Relación (MER)

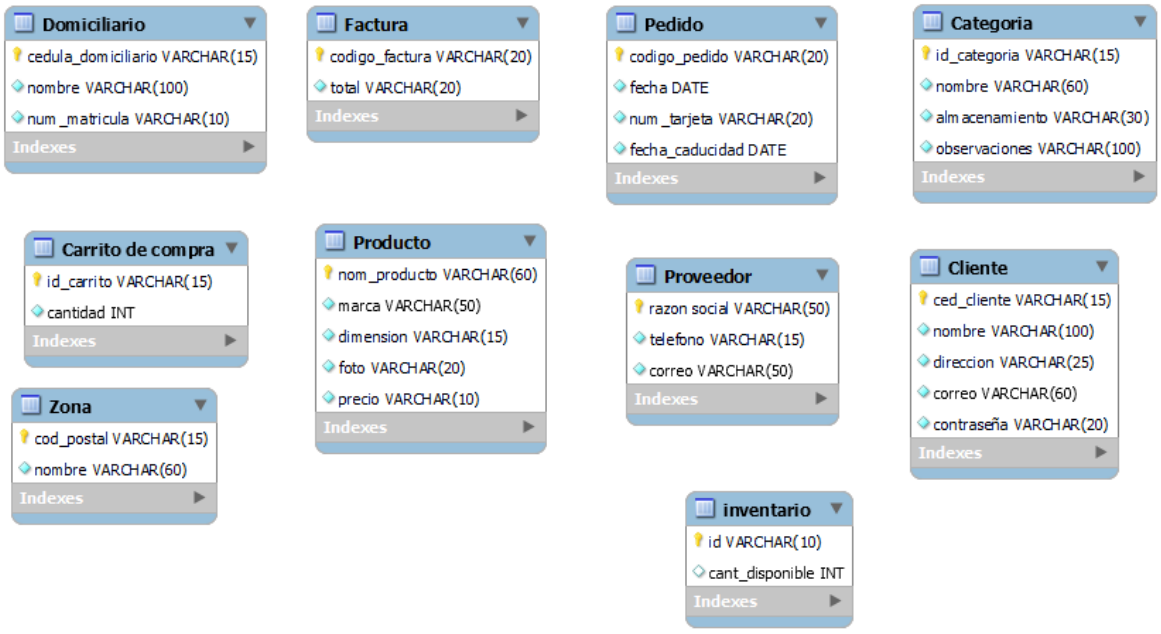
Después de recolectar toda la información se empieza a crear ya el modelo uniendo todas las entidades con sus atributos y relacionándolas entre ellas.



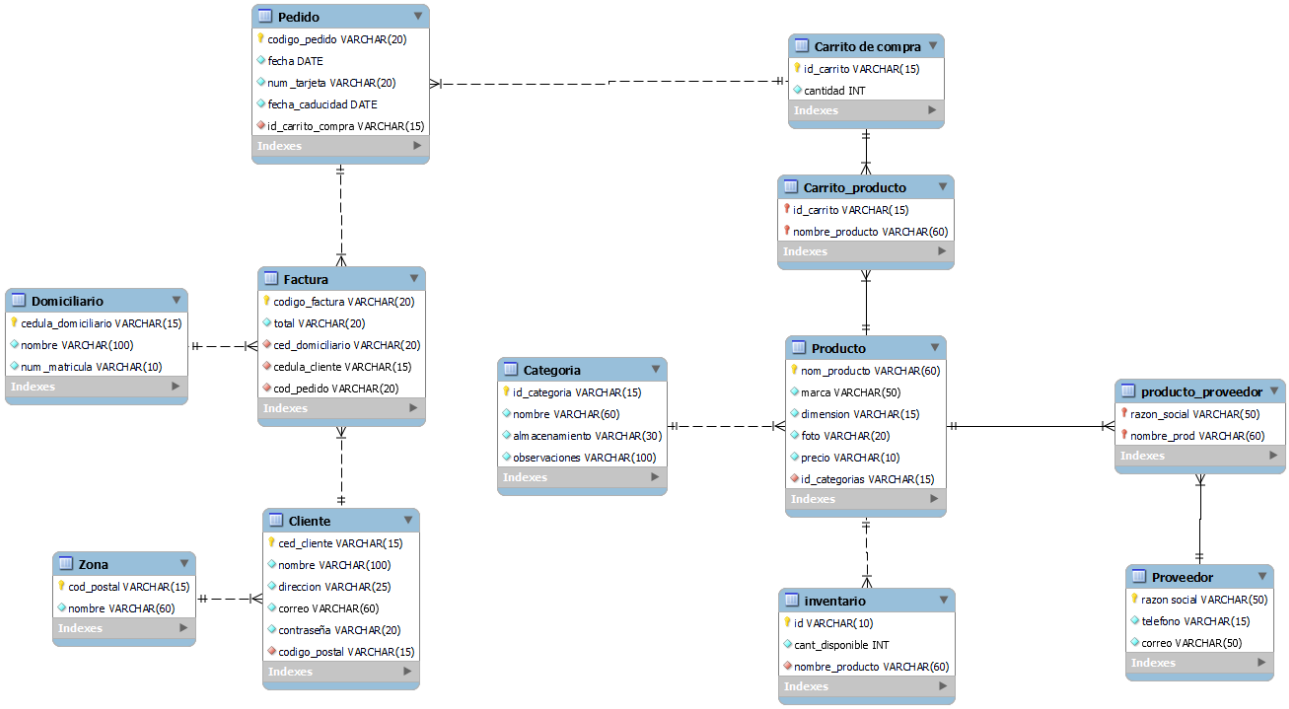

En el modelo se puede evidenciar la relación entre proveedor y productos ya que el proveedor es el que suministra los productos de la tienda don pepe cada producto tiene su categoría, también se le relaciona un inventario para poder controlar la cantidad de productos vendidos y así mismo los productos se van agregando al carrito de compra luego de terminar de escoger los productos y sus cantidades, se confirma la compra mediante un pedido esto me genera una factura que esta relacionado con el cliente y se muestra el total de la compra, después de generada la factura un domiciliario es el encargado de realizar la entrega de la compra donde la factura contiene todos los datos del cliente en que zona se encuentran y productos seleccionados.

## Modelo Relacional (MR)

Primero vamos a transformar cada entidad en una tabla, cada una con sus correspondientes atributos que se convierten en columnas en la tabla.




Después ya se puede empezar agregar las claves foráneas de cada tabla para empezarlas a relacionar entre si como se especifico en modelo entidad relación, también se crean las nuevas tablas que sean necesarias en este caso tenemos dos relaciones N:M (proveedor – productos) y (productos – carrito de compra) cada una se le crea su tabla correspondiente con las llaves primarias de cada una



# Normalización

A continuación, se empieza a aplicar la normalización en toda la base de datos organizando los datos y eliminar redundancia si se encuentran y así mismo poder mejorar la integridad de los datos.

## Primera forma normal 1FN:

| Característica | Condición |
|---|---|
| Cada atributo debe contener valores atómicos | Cumple |
| Cada fila de la tabla debe ser única | Cumple |
| Debe prevalecer un crecimiento vertical de los datos y no horizontal | Cumple |
| no deben existir grupos repetitivos entre los datos | Cumple |

## Segunda forma normal 2FN:

| Característica | Condición |
|---|---|
| Estar en 1FN | Cumple |
| Las relaciones deben tener una clave principal de preferencia simple | Cumple |
| Cada atributo de la tabla debe depender totalmente del atributo clave | Cumple |

## Tercera forma normal 3FN:

| Característica | Condición |
|---|---|
| Estar en 2FN | Cumple |
| No deben existir atributos no principales que dependan del atributo clave | Cumple |

# Definición Base de Datos

Para definir la base de datos se utilizaron sentencias SQL en el SGBD MySQL Workbench

- Creación de la base de datos

```
create database tiendaDonPepe;
use tiendaDonPepe;
```

- Creación de cada tabla

```
create table domiciliario (
  cedula_domiciliario VARCHAR(15) primary key,
  nombre VARCHAR(100),
  num_matricula VARCHAR(10)
);
```

```
create table zona (
   cod_postal VARCHAR(15) primary key,
   nombre VARCHAR(60)
);
```

```sql
create table cliente(
ced_cliente VARCHAR(15) primary key,
nombre VARCHAR(100),
direccion VARCHAR(25),
correo VARCHAR(60),
contraseña VARCHAR(20),
codigo_postal VARCHAR(15),
foreign key (codigo_postal) references zona(cod_postal)
);
```

```sql
create table producto(
nom_producto VARCHAR(60) primary key,
marca VARCHAR(50),
dimension VARCHAR(15),
foto VARCHAR(20),
precio VARCHAR(10),
id_categorias VARCHAR(15),
foreign key (id_categorias) references categoria(id_categoria)
);
```

```sql
create table categoria(
id_categoria VARCHAR(15) primary key,
nombre VARCHAR(60),
almacenamiento VARCHAR(30),
observaciones VARCHAR(100)
);
```

```sql
create table carrito_de_compra(
id_carrito VARCHAR(15) primary key
);
```

```sql
create table pedido(
codigo_pedido VARCHAR(20) primary key,
fecha DATE,
num_tarjeta VARCHAR(20),
fecha_caducidad DATE,
id_carrito_compra VARCHAR(15),
foreign key(id_carrito_compra) references carrito_de_compra(id_carrito)
);
```

```sql
create table proveedor(
razon_social VARCHAR(50) primary key,
telefono VARCHAR(15),
correo VARCHAR(50)
);
```

```sql
create table factura(
codigo_factura VARCHAR(20) primary key,
total VARCHAR(20),
ced_domiciliario VARCHAR(20),
cedula_cliente VARCHAR(15),
cod_pedido VARCHAR(20),
foreign key(ced_domiciliario) references domiciliario(cedula_domiciliario),
foreign key(cedula_cliente) references cliente(ced_cliente),
foreign key(cod_pedido) references pedido(codigo_pedido)
);
```

```sql
create table inventario(
id_inventario VARCHAR(10) primary key,
cant_disponible INT,
nombre_producto VARCHAR(60),
foreign key(nombre_producto) references producto(nom_producto)
);
```

- Creación de cada tabla que son generadas por la relación N:M

```sql
create table producto_proveedor(
razon_social VARCHAR(50),
nombre_prod VARCHAR(60),
primary key(razon_social,nombre_prod),
foreign key(razon_social) references proveedor(razon_social),
foreign key(nombre_prod) references producto(nom_producto)
);
```

Se realiza una modificación en esta tabla ya que al principio la cantidad estaba en la tabla carrito de compra, paso este atributo a la tabla carrito compra producto para llevar un mejor registro de los productos que son agregados al carrito de compra y la cantidad de cada producto.

```
create table carrito_compra_producto(
id_carrito_compra VARCHAR(15),
nombre_producto VARCHAR(60),
cantidad INT,
primary key(id_carrito_compra,nombre_producto),
foreign key(id_carrito_compra) references carrito_de_compra(id_carrito),
foreign key(nombre_producto) references producto(nom_producto)
);
```

## Consultas

- Conocer la categoría y sus productos

```
select categoria.nombre as Nombre_Categoria,
group_concat(producto.nom_producto separator ', ') as Nombre_Productos
from producto inner join  categoria
on producto.id_categorias = categoria.id_categoria
group by categoria.nombre;
```

Esta consulta nos permitirá saber las categorías y que productos se encuentran en cada una para tener un mejor manejo de cada producto en los estantes o lugares de almacenamientos que pueden ser vitrinas, nevera, congelador o bodegas

- Conocer cuantas unidades hay de cada producto

```
select producto.nom_producto as Nombre_Producto, sum(inventario.cant_disponible) as Unidades_Disponibles
from producto inner join inventario
on producto.nom_producto= inventario.nombre_producto
group by producto.nom_producto;
```

Es importante llevar un registro de la cantidad de productos que nos quedan en el inventario para así mismo poder hacer un pedido al proveedor a tiempo y no quedarnos sin unidades disponibles

- Conocer que proveedor ha suministrado que productos

```
select proveedor.razon_social as Nombre_Proveedor,
 group_concat(producto.nom_producto separator ', ')
as Nombre_Productos
from producto inner join producto_proveedor
on producto.nom_producto= producto_proveedor.nombre_prod
inner join proveedor
on proveedor.razon_social= producto_proveedor.razon_social
group by proveedor.razon_social;
```

Esta consulta es de gran importancia ya que los proveedores son los encargados de suministrarme los productos, también estos productos los traen varios proveedores entonces se podría hacer un análisis con que proveedor tengo mejores precios a la hora de hacer un pedido y así mismo generar mas ganancia a la tienda

- Conocer la zona y sus clientes

```
select zona.nombre as Barrio, group_concat(cliente.nombre separator ', ') as Clientes
from zona inner join cliente
on zona.cod_postal=cliente.codigo_postal
group by zona.nombre;
```

Es bueno saber en que barrio se encuentra cada cliente para determinar un tiempo de espera en lo que el pedido es entregado

- Conocer que productos se han agregado al carrito de compra

```
select carrito_compra_producto.id_carrito_compra as Carrito_Compra,
carrito_compra_producto.nombre_producto as Nombre_Producto,
carrito_compra_producto.cantidad as Cantidad
from carrito_compra_producto
where carrito_compra_producto.id_carrito_compra='1';
```

Este tipo de consulta va mas dirigida al cliente ya que en el momento que entra a la aplicación empieza agregar los productos y verifica en el carrito cuales son los productos que lleva y la cantidad de cada uno para llevar un control sobre lo que necesita

- Conocer cuantos pedidos han sido confirmados

```
select pedido.codigo_pedido as Codigo,
pedido.id_carrito_compra as Pedido_Confirmado
from pedido;
```

Esta consulta se relaciona mas cuando un cliente confirma su pedido, pero no genera la factura entonces se envía un mensaje por medio de la aplicación al cliente que tiene algunos productos en el carrito de compra y si desea continuar para poder terminar la compra

- Conocer cuantos pedidos ha repartido cada domiciliario

```
select domiciliario.nombre as Nombre_Domiciliario, count(factura.cedula_cliente) as Numeros_Pedidos
from domiciliario inner join factura
on domiciliario.cedula_domiciliario= factura.ced_domiciliario
group by domiciliario.nombre;
```

Se puede llevar un historial del domiciliario que reparta mas pedidos en un determinado tiempo y generar un incentivo al domiciliario

- Cuantos pedidos ha generado cada cliente

```
select cliente.nombre as Nombre_Cliente, count(factura.codigo_factura) as Pedidos_Realizados
from cliente inner join factura
on cliente.ced_cliente= factura.cedula_cliente
group by cliente.nombre;
```

Esta consulta nos puede ayudar para establecer un tipo de fidelidad por parte de cada cliente y así mismo en pedidos futuros generar un tipo de descuento en algunos productos o en el total de la factura, por el lado otros clientes que no realicen muchos pedidos incentivarlos a usar más la aplicación ya que se manejan buenos productos de excelente calidad con los mejores precios.

- Conocer el numero de pedidos repartidos por cada domiciliario en una zona especifica

```
select domiciliario.nombre as Nombre_Domiciliario,
zona.nombre as Zona,
count(factura.cod_pedido) as Numero_Pedidos
from domiciliario inner join factura
on domiciliario.cedula_domiciliario = factura.ced_domiciliario
inner join cliente  on factura.cedula_cliente = cliente.ced_cliente
inner join zona on cliente.codigo_postal= zona.cod_postal
where zona.nombre = 'Emilio sierra'
group by domiciliario.nombre;
```

se llevaría un registro de los domiciliarios que se encuentran mas cerca de cada zona para así mismo reducir el tiempo de espera por parte del cliente

- Conocer el número de productos en cada categoría suministrado por cada proveedor

```
select proveedor.razon_social as Nombre_Proveedor, categoria.nombre as Nombre_Categoria,
count(producto.nom_producto) as Numero_de_Productos
from proveedor inner join producto_proveedor
on proveedor.razon_social = producto_proveedor.razon_social
inner join producto on producto_proveedor.nombre_prod = producto.nom_producto
inner join categoria on producto.id_categorias = categoria.id_categoria
group by proveedor.razon_social, categoria.nombre;
```

Es una forma mas organizada de saber que proveedor me trae que producto y a cual categoría corresponde para que cuando llegue el pedido sea más fácil de organizar los productos en sus diferentes almacenamientos y condiciones

## Procedimientos

- Actualizar la tabla inventario

```
DELIMITER //
create procedure actualizar_inventario(in Nom_producto varchar(60), Cantidad int )
begin
    update inventario
    set cant_disponible = cant_disponible -  Cantidad
    where nombre_producto = Nom_producto;
end;

call actualizar_inventario('jamon',3);
```

Este procedimiento lo vamos a utilizar para que se actualice el inventario cada vez que un producto se agregado al carrito de compras y generado ya la factura, entonces se va disminuir la cantidad de productos

- Agregar un nuevo producto

```
DELIMITER //
create procedure agregar_producto_tienda(in Nom_producto varchar(60), Marca_prod varchar(50),
Dimen_prod varchar(15), foto_prod varchar(20), precio_prod varchar(10), Id_categoria varchar(15))
begin
    insert into producto (nom_producto,marca,dimension,foto,precio,id_categorias)
    values (Nom_producto, Marca_prod,Dimen_prod,foto_prod,precio_prod,Id_categoria);
end;
call agregar_producto_tienda('Lenteja','diana','500 gr','letej.png','3200','104')
```

Es importante saber que en una tienda constantemente lleguen nuevos productos innovadores, por lo tanto este procedimiento nos va permitir agregar un numero producto con todas sus características y también a una determinada categoría

- Actualizar precio de un producto

```
DELIMITER //
create procedure modificar_precio(in Nom_producto varchar(60), Precio_prod varchar(10))
begin
    update producto set precio=Precio_prod where nom_producto= Nom_producto;
end;

call modificar_precio('jamon','4800')
```

Todos somos consientes que cada día las cosas van aumentando de precio, por eso se crea este procedimiento con la finalidad que si en un pedido que me llegue de un proveedor aumenta o disminuye su precio me permita hacer esa modificación en la lista de productos

- Crear zona

```
DELIMITER //
create procedure crear_zona(in Codigo_postal varchar(15), Nombre_postal varchar(60))
begin
    insert into zona (cod_postal,nombre)
    values (Codigo_postal,Nombre_postal);
end;

call crear_zona('B-06','Fusacatan')
```

Crear una nueva zona es para que me permita definir cuantas zonas tengo disponibilidad para poder repartir los pedidos generados por la factura

## Trigger

Para usar la funcionalidad de los triggers se debe crear una tabla donde se lleve el control de los registros que se realicen

- Crear tabla control de cambios de la tienda don pepe

```
create table control_de_cambios_tienda_pepe(
    usuario varchar(45),
    accion varchar(45),
    fecha datetime default current_timestamp
);
```

En esta tabla se definen el nombre del usuario y la acción que realiza en la tabla con una fecha para llevar mejor el registros de los cambios ocasionados en la tabla.

- Generar un registro cuando se ingrese una categoría

```
delimiter //
create trigger agregar_categoria after insert on categoria
    for each row
    begin
        insert into control_de_cambios_tienda_pepe (usuario,accion,fecha)
        values ('Luisa mendez','Agregar Categoria', now());
    end;
    //

insert into categoria (id_categoria,nombre,almacenamiento,observaciones)
values ('106','cremas','seco','dejar en estantes');
```

Este trigger genera automáticamente un registro en la tabla control de cambios cada vez que le agreguemos una nueva categoría.

- Generar un registro cuando se actualice una categoría

```
delimiter //
create trigger actualizar_categoria after update on categoria
    for each row
    begin
        insert into control_de_cambios_tienda_pepe (usuario,accion,fecha)
        values ('Pedro perez','Actualizar', now());
    end;
    //

update categoria set observaciones= 'dejar en estantes y alejado de los jabones'
where id_categoria = '106';
```

Este trigger genera automáticamente un registro en la tabla control de cambios cada vez que se le actualice algún campo a una categoría.

- Generar un registro cuando se elimine una categoría

```
delimiter //
create trigger eliminar_categoria after delete on categoria
    for each row
    begin
        insert into control_de_cambios_tienda_pepe (usuario,accion,fecha)
        values ('Vivian Gonzalez','eliminar categoria', now());
    end;
    //

delete from categoria where id_categoria='106';
```

Este trigger genera automáticamente un registro en la tabla control de cambios cada vez que se elimine algún registro en la tabla categoría.

- Generar un registro cuando se ingrese un proveedor

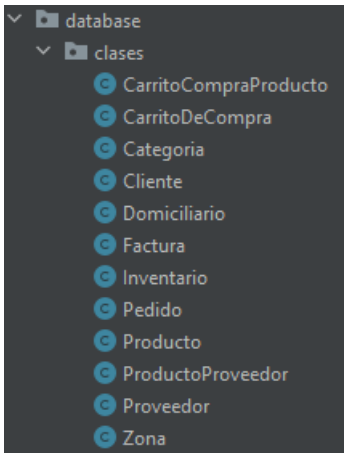
```
delimiter //
create trigger agregar_proveedor after insert on proveedor
    for each row
    begin
        insert into control_de_cambios_tienda_pepe (usuario,accion,fecha)
        values ('Luisa mendez','Agregar Proveedor', now());
    end;
    //

insert into proveedor (razon_social,telefono,correo)
values ('zoto y zuluaga','345345','zoto@gmail.com')
```

Este trigger genera automáticamente un registro en la tabla control de cambios cada vez que le agreguemos una nueva categoría.

## Registro en la Base de datos

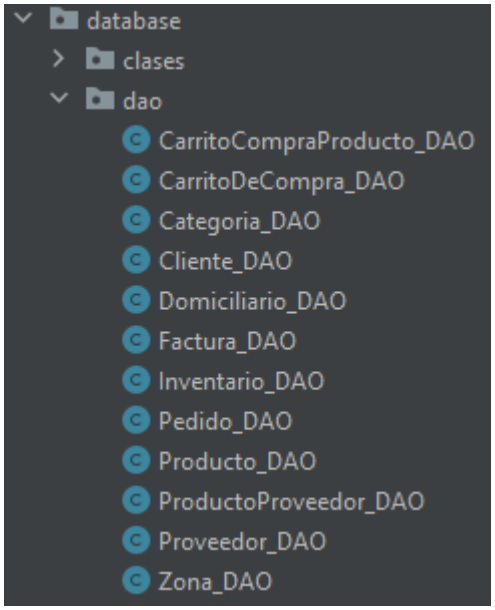
Para el registro de los datos se utilizo el IDE Intellij con la librería Faker, para la organización de las clases se uso el patron de diseño DAO que me permite acceder a la base de datos de forma independiente, las carpetas se dividieron de la siguiente forma.

- Clases



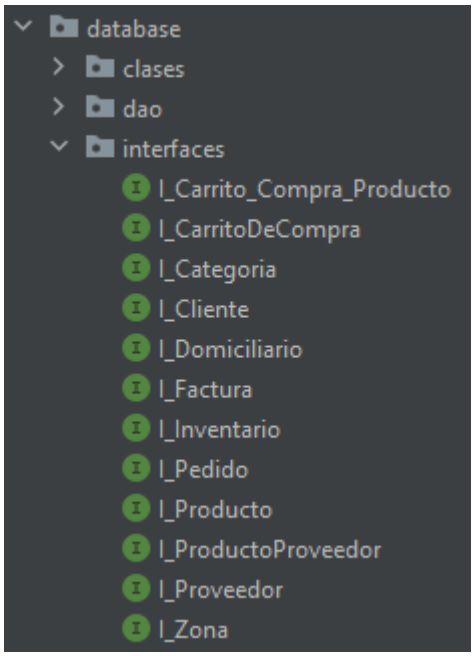

En cada clase se definieron los mismos atributos que en tablas en la base de datos, cada clase contene sus atributos, constructor y métodos setter y getter.

- DAO




Cada clase contiene el desarrollo de los métodos asignados por su interface

- Interfaces




Se usa como plantilla para definir los métodos necesarios para cada clase